MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PROGRAMA DE PÓS-GRADUAÇÂO EM ENGENHARIA MECÂNICA

ORGANIZAÇÃO DE CONHECIMENTO E INFORMAÇÕES PARA INTEGRAÇÃO DE COMPONENTES EM UM ARCABOUÇO DE PROJETO ORIENTADO PARA A MANUFATURA (DFM)

por

André Luiz Tietböhl Ramos

Tese para obtenção do Título de Doutor em Engenharia

Porto Alegre, Novembro, 2015

ORGANIZAÇÃO DE CONHECIMENTO E INFORMAÇÕES PARA INTEGRAÇÃO DE COMPONENTES EM UM ARCABOUÇO DE PROJETO ORIENTADO PARA A MANUFATURA (DFM)

por

André Luiz Tietböhl Ramos
Mestre em Engenharia de Produção

Tese submetida ao Programa de Pós-Graduação em Engenharia Mecânica, da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos necessários para obtenção do Título de

Doutor em Engenharia

Área de Concentração: Processos de Fabricação

Orientador: Flávio José Lorini

Aprovada por:

Prof Dr. João Carlos Espíndola Ferreira, GRIMA/UFSC

Prof Dr. José Leomar Todesco, EGC/UFSC

Prof Dr. José Antônio Esmério Mazzaferro, PROMEC/UFRGS

Prof. Dr. Luiz Alberto Oliveira Rocha
Coordenador PROMEC

Porto Alegre, 06, Novembro, 2015

DEDICATÓRIA

Eu gostaria de dedicar este trabalho a meus pais Ivan Ferraz Ramos e Marisa Tietböhl Ramos por seu ***infinito amor, dedicação e compreensão****, sempre mostrando o* ***verdadeiro significado*** *destas palavras. Adicionalmente, claro, quero dedicar este trabalho também a minhas irmãs Karin Tietböhl Ramos, uma pessoa* ***única e muito especial****, e Aline Tietböhl Ramos, a caçula maravilhosa.*

Integralmente, este trabalho é dedicado a meus queridos filhos Nicolas Tietböhl Ramos e Michelle Tietböhl Ramos; agradeço muito por sua existência. Deus me abençoou com eles.

E, obviamente, acima de tudo, a Deus.

AGRADECIMENTOS

Inicialmente, gostaria de agradecer a duas pessoas muito importantes para conclusão desta pesquisa: Dr. Flávio José Lorini pela orientação correta e opiniões consistentes, por vezes sobrepesando sua admirável fundamentação teórica, e, especialmente, ao Dr. Joyson Luiz Pacheco pela sugestão em dar prosseguimento à pesquisa pelo Programa de Pós-Graduação em Engenharia Mecânica (PROMEC). A cada um dos membros da banca, Dr. João Carlos Espíndola Ferreira, Dr. José Antônio Esmério Mazzaferro e, especialmente, Dr. José Leomar Todesco que, além de competente membro da banca que muito me auxiliou, é um terno amigo pessoal, expresso meus sinceros agradecimentos por suas consistentes sugestões e questionamento.

Tenho a honra de ter a amizade destas queridas pessoas, alguns também colegas profissionais, que sempre estarão comigo: Prof. Dr. Vinícius Medina Kern, Dra. Luciana Saraiva Kern, Prof. Dr. Rubem Dutra Ribeiro Fagundes, Dra. Regina Fagundes, Prof. Dr. Jorge Hugo Silvestrini, Prof. Dr. João Carlos Pinheiro Beck, Prof. Nilson Valega Fernandes, estes dois chefes na verdadeira acepção da palavra, Profa. Dra. Berenice Anina Dedavid, Profa. Dra. Rosina Weber, Profa. Dra. Maria Cleci Martins, Dr. Leonardo Rocha de Oliveira e ao Prof. Dr. Roberto Carlos dos Santos Pacheco. A meus eternos amigos, Álvaro Ribeiro Fazio, José Luis Nieto, João Norberto Bratkowski, Dr. Alexandre Reus Baroni, Prof. Tiago Leonardo Broilo, Dr. Paulo Ernandorena, Prof. Janio Alves e Filipi Damasceno Vianna fica neste meu muito obrigado, sem esquecer o valiosíssimo auxílio do Rafael de Moura Sperroni.

Da mesma forma, eu nunca esquecerei o suporte do Conselho Nacional de Pesquisa (CNPq), especialmente o Dr. Carlos Pittaluga.

Finalmente, eu gostaria de agradecer muito a algumas pessoas extremamente importantes na e para minha vida, destacando algumas em particular: Dr. José de Jesus Peixoto Camargo, bem como a toda sua equipe, especificamente Dr. Spencer Marcantônio Camargo e Dra. Beatriz Gehm Moraes; adicionalmente, ao Dr. Renato Moraes Lucas e ao Dr. Nédio Steffen.

ACKNOWLEDGMENTS

The author of this work would like to thank from his heart and soul this work's previous main advisor: Dr. Michael P. Deisenroth for his excellent guidance, always showing understanding and comprehension along with technical expertise, as well as its co–advisor Dr. Patrick C. Koelling. I also thank very much each member of my previous committee for their valuable support and questions throughout the previous research. Namely, I outline and thank Dr. Janis Terpenny, a very kind person, Dr. Arvid Myklebust, and Dr. Jesus M. de la Garza for their collaboration as well. Last but but not least, I'll never forget and thank my friends from India: Dr. Khrishna Kumar Khrishnan and Dr. Malay Dalal.

RESUMO

A constante evolução de métodos, tecnologias e ferramentas associadas na área de projeto fornece maior capacidade para o projetista. Entretanto, ela também aumenta os requisitos de interfaces e controle do conjunto de componentes de projeto consideravelmente. Tipicamente, este aspecto está presente na área de Projeto Orientado para a Manufatura (DFM) onde existem diversos distintos componentes. Cada um dos componentes existentes, ou futuros, pode ter foco diferente, consequentemente com requisitos de informação, utilização e execução distintos. Este trabalho propõe a utilização de padrões conceituais flexíveis de informação e controle de forma abrangente em uma arquitetura de Projeto Orientado para a Manufatura (DFM). O objetivo principal é auxiliar a análise e resolução de DFM, bem como dar suporte à atividade de projeto estruturando e propondo uma solução em relevantes aspectos em DFM: *estruturação do contexto das informações (ou conhecimento) em DFM*. A arquitetura utiliza as seguintes atividades de projeto em processos de usinagem: Tolerância, Custo, Acessibilidade da ferramenta, Disponibilidade de máquinas e ferramentas e Análise de materiais para demonstrar a relevância da correta contextualização e utilização da informação no domínio DFM . Sob forma geral, concomitantemente, as amplas necessidades de compreensão dos distintos tipos e formas da informação em DFM demandam que uma arquitetura de projeto tenha capacidade de gerenciar/administrar diferentes *contextos de informações de projeto*. Este é um tópico relevante tendo em vista que existem diferentes atividades DFM que eventualmente devem ser incluídas no ato de projetar. Tipicamente, cada uma delas tem requisitos distintos em termos de dados e conhecimento, ou contextualização do projeto, que idealmente poderiam ser gerenciados através da arquitetura de informação atual – STEP. A arquitetura proposta gerencia *contextos de informações de projeto* através de `ontologias` direcionadas no domínio DFM. Através dela, será possível compreender e utilizar melhor as intrínsecas interfaces existentes nas informações deste domínio, além de, através disto, aumentar a flexibilidade e eficácia de sistemas DFM.

Palavras-chave: Projeto Orientado para Manufatura (DFM); Ontologias; Representação do Conhecimento; Organização do Conhecimento; Arquitetura (Arcabouço) de Sistemas; Integração da Informação.

ABSTRACT

This work proposes the use of industry standards to support the utilization of Design for Manufacturing (DFM) techniques in a comprehensive scale in the design field. The specific aspect being considered in an architecture is the definition and structure of *DFM information context*. In order to demonstrate the research concepts, some design activities are implemented the framework (which is focused in machining processes): Tolerancing model, Cost model based on material remove processes, Tool Accessibility model taking into consideration the part being designed, Availability of Machines and Tools model, and Material analysis. The broad needs of design–based frameworks, in general, require that its architecture must have the capabilities to handle different framework design information utilization contexts, or *information context concepts*. This is a relevant aspect since there are several DFM components/activities that preferably should be included in the design process. Traditionally, each one of them might have distinct data & knowledge requirements, which can be handled by the current information architecture – STEP – only in part. Additionally, each one of them might have, or need, different forms of understanding DFM information (information context). The framework handles *information context concepts* through the use of the `ontologies` targeted to the DFM field. It is expected that a better comprehension and usage of the intrinsic information interfaces existent in its domain be achieved. Through it, more flexible and effective DFM systems information-wise can be obtained.

Keywords: Design for Manufacturing (DFM); Ontologies; Knowledge Representation; Knowledge Organization; Systems Framework; Infortation Integration.

ÍNDICE

1 INTRODUÇÃO 1
1.1 Aspectos iniciais 1
1.2 Apresentação do Problema 3
1.3 Objetivos da Pesquisa 9
1.4 Organização do Documento 9

2 DOMÍNIO DE PROJETO, PRODUTO E DFM 11
2.1 Arquiteturas de Projeto 11
2.1.1 Perspectiva Histórica 13
2.1.2 Implementação de Arquiteturas de Projeto 13
2.1.2.1 Propósito Geral 14
2.1.2.2 Propósito Específico 15
2.1.2.3 Arquiteturas Comerciais de Projeto 17
2.2 Sistemas Integrados 17
2.2.1 Sistemas Integrados Comerciais de Projeto 18
2.2.2 Sistemas de Projeto baseados em Agentes 22
2.3 Arquiteturas de Projeto para a Manufatura (DFM) 25

3 INFORMAÇÕES NO DOMÍNIO DE PROJETO, PRODUTO E DFM . . . 29
3.1 Integração e Intercâmbio de Informação 29
3.1.1 Contexto Histórico 29
3.1.2 Intercâmbio Estático de Informações 31
3.1.3 Intercâmbio de Informações em Tempo de Execução 40
3.2 Utilização de Informações Contextuais 41
3.3 Estrutura e Representação Computacional de Informações Contextuais de Projeto 74

4 ARCABOUÇO DE INFORMAÇÕES E CONHECIMENTO PARA INTEGRAÇÃO DE COMPONENTES DE PROJETO ORIENTADO PARA MANUFATURA (DFM) 85
4.1 Arquitetura DFM de Informação 86
4.1.1 Modelo Cognitivo 88
4.2 Funcionalidade 89
4.3 Arquitetura sistêmica contextual de informação 93
4.3.1 Ontologias 93
4.3.2 Modelagem da informação 96
4.3.3 Modelagem da Informação em Projeto para Manufatura 98

4.3.3.1 Definição taxonômica e seu detalhamento . . . 98
4.3.3.2 Definição do domínio e escopo da ontologia . . . 101
4.3.3.3 Caracterização dos detalhes do domínio da informação . . . 102
4.3.3.3.1 Taxonomia . . . 102
4.3.3.3.2 Hierarquia Taxonômica (Classes) . . . 107
4.3.3.3.3 Detalhes, ou Propriedades, (*Slots*) das Classes . . . 123
4.4 Informação de suporte DFM . . . 131
4.4.1 Materiais . . . 133
4.4.2 Dimensão . . . 134
4.5 Metainformação no domínio DFM . . . 135

5 ANÁLISE DE CONSISTÊNCIA E COMPLETEZA DO ARCABOUÇO DE INFORMAÇÕES E CONHECIMENTO INTEGRADOS DE PROJETO ORIENTADO PARA MANUFATURA (DFM) . . . 139
5.1 Princípios/regras DFM do domínio e seu escopo . . . 141
5.1.1 Flexibilidade Dimensional inferida . . . 141
5.1.2 Usinabilidade inferida . . . 143
5.1.3 Padronização inferida . . . 145
5.1.4 Simplicidade inferida . . . 146
5.2 Análise da metainformação . . . 147
5.3 Completeza da estrutura ontológica no escopo selecionado do domínio DFM . . . 149
5.3.1 Asseveração taxonômica da ontologia no domínio e escopo da pesquisa 150
5.3.2 Materialização da ontologia no domínio e seu escopo . . . 150
5.3.2.1 Materialização da Análise DFM . . . 152
5.3.2.2 Materialização de Princípios/Regras DFM . . . 156
5.3.3 Interface com a máquina de inferência OWL . . . 158
5.3.3.1 Exemplo de análise DFM em uma peça . . . 166

6 CONCLUSÕES . . . 171

REFERÊNCIAS BIBLIOGRÁFICAS . . . 175

APÊNDICE A – INTRODUÇÃO À LINGUAGEM OWL . . . 191

APÊNDICE B – DESCRIÇÃO TEXTUAL DETALHADA DAS CLASSES, PROPRIEDADES E INDIVÍDUOS . . . 193
B.1 Taxonomia anotada - introdução . . . 194
B.2 Taxonomia hierárquica anotada . . . 196
B.3 Taxonomia operacional anotada . . . 237

B.4 Taxonomia anotada de indivíduos 252
B.5 Taxonomia anotada de dados 259

APÊNDICE A – LÓGICA MODAL E TAXONOMIA 263
A.1 Aspectos Iniciais 263
A.2 Opinião 264
A.3 Semântica Kriptke 265
A.4 Relacionamentos 265
A.5 Lógica Doxástica 266
A.6 Lógica Epistemológica 266
A.7 Lógica Deôntica 267
A.8 Necessidade 267
A.9 Lógica Descritiva 268

LISTA DE FIGURAS

2.1 Arquitetura Quadro–negro. 12
2.2 Arquitetura baseada em Agentes, adaptada de Bradshaw. 13
2.3 Árvore de dependência entre parâmetros. 14
2.4 Arquitetura de Agentes baseada em JADE. 22
2.5 Arquitetura de Agentes baseada em Arquitetura JADE integrada baseada em Agentes no domínio DFM. 23
2.6 Arquitetura DFM baseada em Agentes. 24

3.7 Arquitetura STEP 34
3.8 Arquitetura de alto nível de implementação no modelo STEP. 35
3.9 Modelo de implementação STEP e interface SDAI. 36
3.10 Arquitetura da linguagem KQML. 39
3.11 Relações entre conhecimento e informação em um sistema de manufatura baseado em conhecimento segundo Guerra-Zubiaga e Young. 42
3.12 Formalismo da atividade de projeto. 47
3.13 Desenvolvimento de ontologias genéricas de projeto em engenharia. 49
3.14 Taxonomia genérica de atividades de gerenciamento de projeto. 50
3.15 Linguagens possíveis de modelagem de produtos. 51
3.16 Arquitetura ONTO–PDM de interoperabilidade de sistemas de manufatura. 54
3.17 Utilização de bibliotecas de peças como dados na arquitetura contextual–semântica de informação de engenharia ESW. 59
3.18 Sistema de engenharia estrutural. 60
3.19 Utilização de ontologias em camadas no desenvolvimento colaborativo de produtos. 64
3.20 Meta–ontologia de desenvolvimento colaborativo de produto. 64
3.21 Sistema de conhecimento baseado em ontologias para projeto de reatores químicos. 65
3.22 Ontologia para configuração de produtos. 67
3.23 Atributos do modelo conceitual por ontologias. 70
3.24 Integração entre processos industriais e ontologias. 71
3.25 Metodologia de desenvolvimento integrado de taxonomia de ontologias de projeto. 72
3.26 Taxonomia da atividade de projeto. 73
3.27 Taxonomia do ciclo de vida integrada com manufatura em projeto. 73
3.28 Gerenciamento de alterações ontológicas. 75
3.29 Teoremas básicos da ontologia PSL. 80
3.30 Arquitetura OWL/SWRL sobre Web Semântica. 82

4.31 Hierarquia combinada da informação entre processos e geometria. 86
4.32 Arquitetura de conhecimento, informação e dados. 91
4.33 Arquitetura Orientada a Modelos (MOF). 97
4.34 Arquitetura de desenvolvimento de uma ontologia e sua utilização no domínio DFM desta pesquisa. 99
4.35 Taxonomia definida relacionada ao contexto DFM. 109
4.36 Taxonomia definida relacionada ao contexto DFM (parte). 110
4.37 Regra DFM: Flexibilidade Dimensional definida. 113
4.38 Regra DFM: Usinabilidade definida. 113
4.39 Regra DFM: Padronização definida. 115
4.40 Regra DFM: Simplicidade definida. 116
4.41 GeometryFeature, ManufaturingFeature e sua integração com aspectos tradicionalmente díspares através de ontologias. 133
4.42 Representação meta da informação e conhecimento através, no caso, do vocabulário. 136
4.43 Representação meta da informação e conhecimento no arcabouço utilizando agentes. 137

5.44 Arquitetura definida de informação fundamentada em agentes (complementação da figura 4.35 e como tal, devido ao ‘ *plug-in*” utilizado, não impõe equalidade). 140
5.45 Regra DFM: Flexibilidade Dimensional inferida. 142
5.46 Regra DFM: Manufaturabilidade inferida. 144
5.47 Regra DFM: Padronização inferida. 145
5.48 Regra DFM: Simplicidade inferida. 146
5.49 Princípios/Regras definidas DFM. 147
5.50 Princípios/Regras inferidas DFM. 148
5.51 Mapeamento classe/dados do Laboratório CIM da PUCRS. 155
5.52 Materialização e inferência SPARQL. 159
5.53 Peça-exemplo: bloco com “*pocket*”. 167

LISTA DE TABELAS

1.1 Custo de produção relacionado com acabamento superficial. 7
1.2 Aspectos informacionais *reais*, conexões intrínsecas embutidas, no domínio DFM que não são representadas inequivocamente - exemplo. 7
1.3 Aspectos intrínsecos, porém não expressos inequivocamente, em materiais sujeitos ao processo de usinagem. 8

3.4 Conjuntos–base para ontologia de configuração de produto. 67
3.5 Mapeamento entre EXPRESS e OWL. 83

4.6 Relações lógico-abstratas das atividades DFM-específicas do domínio desta pesquisa com classes definidas em **negrito**. 111
4.7 Relacionamentos disjuntos das relações lógicas abstratas das atividades DFM no domínio da pesquisa com classes definidas em **negrito**. 111
4.8 Relações lógicas abstratas dos membros, ou tipos dos dados, DFM no domínio da pesquisa. 112
4.9 Taxonomia operacional DFM com classes definidas em **negrito**. Todas classes são subclasses da classe Conceito, ou Concept. 112
4.10a Classes de princípio, ou regra, DFM–específico Flexibilidade Dimensional. . 113
4.10b Membros do princípio/regra DFM Flexibilidade Dimensional. 113
4.11 Classes de princípio, ou regra, DFM–específico: Usinabilidade. 113
4.12a Classes e propriedades de informação herdada de Usinabilidade. 114
4.12b Membros do princípio/regra DFM Usinabilidade. 114
4.13a Classes de princípio, ou regra, DFM–específico: Padronização - foco em Materiais. 115
4.14 Classes de princípio, ou regra, DFM–específico Simplicidade. 116
4.15 Relações lógicas abstratas do conceito DFM do Processo. 116
4.16a Relações lógicas abstratas do conceito DFM de Máquina e classes DFM–específicas: Máquina. 117
4.16b Membros do conceito Máquina. 117
4.17a Classes DFM–específicas: Operação. 118
4.17b Membros da atividade Operação. 118
4.18 Classes DFM–específicas: Produto. 119
4.19 Membros do conceito Produto. 119
4.20a Classes DFM–específicas: Dimensão. 120
4.20b Membros da atividade Dimensão. 120
4.21a Classes DFM–específicas: Material. 121
4.21b Membros da atividade Material. 121

4.22a Classes DFM–específicas: Pessoal. . . . 121
4.22b Membros do conceito Pessoal. . . . 121
4.23a Custo. . . . 122
4.23b Custo inicial, ou de investimento. . . . 122
4.23c Custo de fabricação. . . . 122
4.23d Custo de fabricação: propriedades de informação herdada. . . . 123
4.23e Custo de mão de obra. . . . 123
4.23f Membros do Custo. . . . 123
4.24a Assembly. . . . 124
4.24b Part. . . . 124
4.25a Propriedades associadas com o conceito de o que, efetivamente, perfaz DFM, ou seja isDFM, ou sua operacionalização: hasActivity e needsComponent. . . 125
4.25b needsComponent. . . . 125
4.26a hasActivity (isDFMOf). . . . 126
4.26b isNeededComponentOf. . . . 126
4.26c isCostDefinedBy. . . . 126
4.27a isTolerancingFrom. . . . 127
4.27b isToolAccessibilityFrom. . . . 127
4.27c Propriedade Tolerância definida. . . . 127
4.28 Propriedade Acessibilidade da ferramenta. . . . 127
4.29 Relações lógicas abstratas do princípio DFM global. . . . 128
4.30 Definição dos indivíduos. . . . 129
4.31 Indivíduos DFM e seus relacionamentos. . . . 129
4.32 Indivíduos da taxonomia operacional DFM com classes definidas em **negrito**. Todas classes são subclasses da classe Conceito, ou Concept. . . . 130
4.33 Classes de princípio, ou regra, DFM–específico Flexibilidade Dimensional. . 130
4.34 Membros do princípio/regra DFM Flexibilidade Dimensional. . . . 130
4.35 Classes, propriedades, indivíduos e seus relacionamentos (componentes herdados em *itálico*). . . . 131
4.36 Classes, propriedades, indivíduos e seus relacionamentos. (continuação da tabela 4.35: apenas subclasses de AgentAction, que perfazem a segunda parte, e são atividades específicas de DFM são expressas; componentes herdados em *itálico*). . . . 132
4.37 Hierarquia de Propriedades de Dados – Materiais . . . 134
4.38 Hierarquia de Propriedades de Dados – Dimensão . . . 134
4.39 Propriedades de Anotações DFM Genéricas. . . . 138

5.40a Classes de princípio, ou regra, DFM–específico Flexibilidade Dimensional. . 143
5.40b Membros do princípio/regra DFM Flexibilidade Dimensional. . . . 143
5.41 Classes de princípio, ou regra, DFM–específico: Manufaturabilidade. . . . 143

5.42 Classes de princípio, ou regra, DFM–específico: Padronização. 146
5.43 Classes de princípio, ou regra, DFM–específico Simplicidade. 146
5.44 Princípios/Regras definidas DFM. 147
5.45 Atividades (componentes) definidas DFM. 147
5.46 Meta-vocabulário DFM. 149
5.47a Componentes físicos do Laboratório CIM da PUCRS. 153
5.47b Significado dos componentes. 153
5.48 Mapeamento dos dados ao conceito Flexibilidade Dimensional. 157
5.49 Componentes básicos de um projeto de produto e sua manufatura com o maquinário, especificamente a fresa considerando somente contexto genérico de operações de usinagem, e ferramental disponível no LabCIM. . 161
5.50 O que está relacionado com um determinado objeto intrínseco à DFM? . . 162
5.51 Dados da peça-protótipo (as nomenclaturas de "*features*" e faces são customizadas) 167
5.52 Dados de "*features*" da peça-protótipo no banco de dados. 168
5.53 Dados das faces de "*feature*" PPPartData de peça-protótipo. 168
5.54 Tipos de "features" que podem ser manufaturadas pela operação do escopo da pesquisa. 169
5.55 URI do ***dado*** da "feature" que não pode ser manufaturada pela operação do escopo da pesquisa. 169

A.1 Primitivas construtoras de classes OWL e suas diferentes sintaxes. 192

LISTA DE SIGLAS E ABREVIATURAS

AEC	*Architecture Engineering Construction*
API	*Application Programming Interface*
AP	*Application Protocols*
AV	*Approach Vector*
BIM	*Building Information Model*
BOM	*Bill of Materials*
B–rep	*Boundary Representation*
CAD/CAM	*Computer Aided Design/Computer Aided Manufacturing*
CAD	*Computer Aided Design*
CAM	*Computer Aided Manufacturing*
CAPP	*Computer Aided Process Planning*
CE	*Concurrent Engineering*
CORA	*Core Ontology for R&A*
CLOS	*Common Lisp Object System*
CNC	*Computer Numerical Control*
CORBA	*Common Object Request Broker Architecture*
CREO	Nome fantasia da plataforma de *software* da *Parametric Technology Corporation*.
CSG	*Constructive Solid Geometry*
CSP	*Constraint Satisfaction Problem*
DAG	*Directed Acyclic-f Graph*
DCMI	*Dublin Core Metadata Initiative*
DFA	*Design for Assembly*
DFM	*Design for Manufacturing*

DFSS	*Design for Six Sigma*
DFx	*Design for X*
DKAP	*Design Knowledge Acquisition Process*
DL	*Description Logics*
DOE	*Design of Experiments*
DXF	*Drawing Exchange Format*
EBD	*Electronic Business Documents*
EBDO	*Electronic Business Documents Ontology*
EDIT	*Engineering Design Integrated Taxonomy*
EPR	*Electronic Product Realization*
ER	*Entity–Relationship*
ESW	*Engineering Semantic Web*
FCA	*Formal Concept Analysis*
FEM/FEA	*Finite Element Method/Finite Element Analysis*
FIL	*Facilitator Interface Libraries*
FOL	*First Order Logic*
FOM	*Federation Object Model*
GC	*Generic Component*
GIS	*Geographic Information Systems*
GSM	*Generic System Models*
HLA	*High Level Architecture*
HRI	*Human-Robot-Interaction*
HTTP	*Hyper Text Transfer Protocol*
IDEF	*Integrated DEFinition*
IGES	*Initial Graphics Exchange Standard*
IR	*Information Resources*

ISOS	*In Search Of Semantics*
ISO	*International Organization for Standardization*
JADE	*Java Agent DEvelopment framework*
KIF	*Knowledge Interchange Format*
KMS	*Knowledge Management System*
KQML	*Knowledge Query Manipulation Language*
KRM	*Knowledge Retrieval Model*
LCS	*Least Common Subsumer*
MASON	*MAnufaturing's Semantics ONtology*
MEB	*Minimum Enclosing Box*
MLT	*Manufacturing Lead Time*
MOF	*Meta–Object Facility*
OMG	*Object Management Group*
OWA	*Open World Assumption*
OWL	*Ontology Web Language*
OWL-DL	*Ontology Web Language – Description Logics*
PCBA	*Printed Circuit Board Assembly*
PDES	*Product Data Exchange Specification*
PDM	*Product Data Management*
PDW	*Process Data Warehousing*
PGC	*Primitive Generic Component*
PLM	*Product Lifecycle Management*
PLM	*Product Life Cycle Management*
POS	*Positioning Ontology*
PSL	*Process Specification Language*
PTC	*Parametric Technology Corporation*

RDBMS	*Relational Data Base Management System*
R&A	*Robotics and Automation*
RDF	*Resource Description Framework*
RDFS	*Resource Description Framework Schema*
RPC	*Remote Procedure Calls*
RP	*Rapid Prototyping*
SDAI	*Standard Data Access Interface*
SDRC	*Structural Dynamics Research Corporationbo*
SKTP	*Simple Knowledge Transfer Protocol*
SOM	*Simulation Object Model*
SPARQL	*SPARQL Protocol and RDF Query Language*
SQWRL	*Semantic Query-Enhanced Rule Language*
STEP	***S**tandard for **T**he **E**xchange of **P**roduct model data*
SWRL	*Semantic Web Rule Language*
TL	*Technological Layers*
TSP	*Traveling Salesmen Problem*
UML	*Unified Modelling Language*
UOB	*Unit of Behaviour*
URI	*Uniform Resource Identifier*
IRI	*Internationalized Resource Identifier*
VLSI	*Very Large Scale Integration*
XMI	XML *Metadata Interchange*
XML	*eXtensible Markup Language*

1 INTRODUÇÃO

1.1 Aspectos iniciais

A crescente competição pelo mercado em escala global vem mudando as características da produção industrial nas últimas décadas, que está impactando na atividade de projeto no ambiente do ciclo de vida de produtos. Historicamente, decisões de projeto podem ser responsáveis por até 70% do custo de manufatura [Stoll, 1991], 60% da qualidade e 50% do tempo de manufatura (MLT) de um produto [Daues e Meeker, 1993]. A atual conjuntura competitiva em nível mundial tornou estes aspectos muito mais volúveis, adicionando um peso considerável à dinamicidade sistêmica em manufatura. A integração destes aspectos distintos tradicionalmente é conhecida como Engenharia Simultânea – ES (ou CE) [Warman, 1990].

Vários componentes/atividades tipicamente associados com ES tem sido implementados por um período de tempo considerável. Computadores mais potentes e facilmente disponíveis permitiram o desenvolvimento e acesso a ferramentas computacionais avançadas tais como modelagem através de Projeto Auxiliado por Computador – CAD, Análise e Modelagem por Elementos Finitos – FEM/FEA e Planejamento do Processo Auxiliado por Computador – CAPP, dentre outras. Um sistema computacional integrado, ou arquitetura, tem sido proposto de forma abrangente na literatura disponível como a solução mais adequada para resolução do problema de Engenharia Simultânea [Kim, Hom e Parthasarathy, 1991, Domazet, Lu e Kalajdzic, 1992, Cutkoski *et al.*, 1993, Edmonds *et al.*, 1994, Hosaka e Kimura, 1990, Gruber, 1993b, Rosen, Dixon e Poli, 1992].

A abordagem de Engenharia Simultânea no desenvolvimento de produto analisa aspectos de manufatura através de Projeto Orientado para Manufatura – DFM [Corbett *et al.*, 1991, Lai, Gupta e Reddy, 2005, Bralla, 1999] e Projeto Orientado para Montagem – DFA. Projeto Orientado para Manufatura é o desenvolvimento de produto incorporando aspectos relativos à fabricação no processo de projeto [Rosen, Dixon e Poli, 1992]. DFM pode ser subdividido de DFM de peça e DFM de produto. O DFM de peça lida com a análise uma peça única de um produto. O DFM do produto efetua a análise de fabricação do produto completo e das inter–relações entre as operações de fabricação de forma integral. Projeto Orientado para Montagem é a análise do DFM do produto com foco nas operações de montagem e sua otimização [Boothroyd e Dewhurst, 1991].

Kuo, Huang e Zhang, 2001 definiram a relevância de DFM nos processos de projeto e manufatura e o quão importante é a integração adequada das atividades específicas necessárias, bem como sua perspectiva no futuro. A utilização efetiva de DFM facilita

a integração das seguintes atividades, tradicionalmente consideradas em estágios mais avançados de projeto.

- Padronização – possibilita aumento do poder de compra, simplificação do gerenciamento dos estoques necessários, redução do custo da peça com melhoria da margem de lucro e aumento de flexibilidade de produção, bem como redução do custo de desenvolvimento de produtos.
- Customização em massa – através desta análise é possível fabricar produtos de acordo com a demanda, os quais são mais eficientemente projetados para determinados mercados, diferentes países ou clientes específicos.
- Fabricação sob encomenda – possibilita a fabricação de qualquer quantidade de produtos customizados em massa ou produtos–padrão sob demanda, sem previsão, lote, estoque ou atraso no faturamento das ordens de compra.
- Racionalização da linha de produção – permite a otimização de linhas de produção para eliminar ou sub–contratar a fabricação de produtos antigos ou de baixo volume de produção. Através disto é possível, imediatamente, aumentar o lucro e simplificar o gerenciamento da cadeia do estoque. Adicionalmente, permite que colaboradores diversos e/ou distintos participem mais cedo de equipes multi–funcionais de desenvolvimento de produto.

A tendência crescente na disponibilidade do poder computacional, tanto em *software* quanto em *hardware*, permitiu a implementação prática de arquiteturas teóricas do ciclo de vida do produto; por exemplo Sudarsan *et al.*, 2005 desenvolveram uma arquitetura genérica para o gerenciamento do ciclo de vida do produto. A relevância da presença do processo de projetar ao longo do ciclo de vida dos produtos foi descrita por Wu *et al.*, 2006. Young, 2003 definiu, ou expressou, claramente as características e requisitos da informação necessária em sistemas heterogêneos com aspectos distintos de granularidade e abstração, tal como no domínio DFM. Dowlatshahi, 1995 desenvolveu um dos primeiros sistemas DFM, entretanto não era uma arquitetura completa e coerente com todos requisitos DFM.

Inicialmente, aspectos sistêmicos, especificamente DFM, foram dimensionamento (tolerâncias) [Ji e Lau, 1999] e raciocínio DFM [Sanchez, Priest e Souto, 1997], neste caso para compreensão da forma e estrutura do conhecimento em DFM. Conforme o conceito de uma arquitetura tornou–se mais conhecida e aceita, uma proposta de implementação *a posteriori* de arquiteturas de conhecimento e raciocínio em DFM foi feita por Bajaj *et al.*, 2003, especificamente para a área de Projeto de Produtos Eletrônicos (EPR).

Poli, 2001 descreve a relevância de estruturar DFM para que o processo efetivo de fabricação de um produto seja simplificado. A inclusão de DFM em Engenharia Simultânea foi especificada por Anderson, 2004. Tipicamente, após o processo DFM ser

estruturado adequadamente é possível que ele seja implementado computacionalmente. Em geral, tradicionalmente, a melhor forma de *implementação computacional* de uma área tão ampla quanto análise DFM é através de uma arquitetura sistêmica que englobe a parte computacional. No caso específico deste trabalho, o foco é no fundamento que permite uma arquitetura eficaz e eficiente através da análise e gerenciamento da informação deste domínio.

Este trabalho é baseado na abordagem de solução do processo de projeto como um algoritmo e assume que o projeto preliminar é o ponto inicial para análise da manufatura. Ele propõe a utilização de uma arquitetura para aplicações DFM que possam ser potencializados em um ambiente computacional integrado com a contextualidade da informação intrínsecos em projeto.

1.2 Apresentação do Problema

Uma arquitetura de projeto DFM possibilita reduções no custo de desenvolvimento de produtos através da avaliação e análise de várias alternativas de manufatura de um projeto de forma adiantada, antes que elas estejam no estágio de manufatura. Conforme expresso inicialmente, existe dinamicidade crescente na atividade de manufatura e já em 1992 a literatura disponível indicava que para cada seis meses de atraso no projeto de um produto o lucro potencial é reduzido em um terço [DEC, 1992]. Levando em consideração este aspecto, em geral, existe a possibilidade do intervalo de tempo relacionado com a obtenção de lucro ser todavia mais curto. Adicionalmente, um ciclo de desenvolvimento de produto reduzido permite que uma empresa obtenha ganho competitivo, fator fundamental para sobrevivência no atual mercado global. Finalmente, produtos de melhor qualidade podem ser desenvolvidos devido a uma etapa de projeto integrada, fator que possibilitará aumento de sua fatia de mercado.

Uma arquitetura de integração de componentes DFM que possua forma de representação de dados e conhecimento (ou informação) mais adequada e completa, componentes (ou agentes[1]) de projeto integrados [Bradshaw, 1997], bem como estrutura atualizável independente do domínio pode simplificar o desenvolvimento de sistemas integrados de projeto. Adicionalmente, o desenvolvimento e implementação eventual da arquitetura proposta poderá simplificar a efetivação de Engenharia Simultânea como paradigma de desenvolvimento de produtos.

O desenvolvimento de uma arquitetura fundamentada em informação impõe novas demandas teóricas e de implementação para sistemas integrados de projeto, especialmente em um sistema que utiliza fontes heterogêneas de informação, como DFM. Neste contexto,

[1]Nesta pesquisa as palavras agente e atividade, ou componente, são sinônimos, embora possuam conceitos e concepção arquitetônica prática diferente. As atividades/componentes da arquitetura proposta possuem características genéricas de agentes de *software*.

Young, 2003 ressalta o quão importante é modelar a informação para que a arquitetura seja efetiva (DFM). Esta efetividade, em geral, está relacionada com a dicotomia entre representação e implementação de dados de projeto pois, na verdade, a abstração da informação em projeto é maior, visto que o foco tradicional é restrito a somente dados [Rocca, 2012].

Inicialmente, a forma mais completa para definir e implementar computacionalmente uma arquitetura que atenda os requisitos descritos é assumida como através da compreensão do contexto das informações necessárias a seus componentes. A crescente disponibilidade de ferramentas individualizadas, que necessitam e utilizam dados comuns, ocasionou o desenvolvimento de formatos padronizados de dados e/ou informação, por exemplo IGES, DXF ou KIF. A grande variabilidade dos dados, juntamente sua estrutura, tornou necessária a definição de uma arquitetura mais abrangente que incluísse conteúdo da informação ao invés de somente dados.

O padrão de metadados **STandard for the Exchange of Product model data** (**STEP**) é o mais completo atualmente disponível pois considera diferentes requisitos de informação em termos de formatos distintos, além de incorporar a sintaxe e a semântica de uma informação específica (apesar de limitada). Adicionalmente, os dados são definidos a partir da linguagem EXPRESS, a qual permite foco na estrutura da informação em termos de sua estrutura representacional. Anteriormente, apenas a representação de dados era possível limitando, desta forma, a capacidade do requisito de integração de informações entre componentes de projeto.

Os requisitos de uma arquitetura de integração de sistemas genéricos são bastante conhecidos e estruturados atualmente. Um aspecto relevante nesta integração, entretanto, ainda não está completamente desenvolvido: o formalismo estruturado e contextualizado para o domínio em questão, no caso DFM. O desenvolvimento e disponibilização computacional cada vez mais emergente do Gerenciamento do Ciclo de Vida do Produto (PLM), especificamente na área de DFM, demanda que este formalismo seja definido e disponibilizado. Conforme descrito na literatura, um formalismo inequívoco flexibiliza a sistematização e posterior desenvolvimento de soluções em ambientes heterogêneos e complexos, como DFM.

A premissa fundamental deste trabalho é que um sistema integrado de projeto conforme descrito pode disponibilizar um ambiente de desenvolvimento de produto melhor e mais apurado que ferramentas isoladas de projeto, que utilizam intercâmbio de informações apenas através de mecanismos simples de troca de dados.

Os requisitos associados com DFM no processo de projeto e sua implementação historicamente sempre existiram. Entretanto a infraestrutura computacional não possuía capacidade para solucionar suas limitações implementacionais sistêmicas. Adicionalmente, a fundamentação arquitetônica da informação tampouco permitia uma solução eficaz e efi-

ciente. A evolução de arquiteturas de sistemas, associadas com novos paradigmas teóricos, permitiu o desenvolvimento, inicialmente mais abstrato, de sistemas de desenvolvimento colaborativo de produtos como, por exemplo, os desenvolvidos por Gruber, Teinembaum e Weber, 1992 e McGuire *et al*., 1993.

A disponibilização de soluções computacionais efetivas de cálculo permitiu que fosse argumentado e proposto o desenvolvimento de uma arquitetura mais eficaz e eficiente. Este tipo de solução permite uma "visualização" mais abstrata, consequentemente mais próxima, da forma humana de compreensão do processo de projeto segundo Gruber, Teinembaum e Weber, 1992 (em conhecimento) e Catalano *et al*., 2009.

O pesquisa proposta tem por objetivo a definição uma estrutura flexível de informação, ou conhecimento, que facilite a integração de ferramentas DFM em projeto mecânico preliminar integrado em Engenharia Industrial. Claramente, este ambiente é muito amplo para sua utilização, de forma completa, em uma solução baseada em pesquisa, ainda que seja abrangente. Desta forma, o escopo do domínio de projeto para teste do objetivo da pesquisa é restrito a peças poliédricas metálicas e processos fundamentais de remoção de materiais como, por exemplo, processos de usinagem por fresagem ou furação. Adicionalmente, a pesquisa é restrita a peças isoladas/únicas pois produtos com múltiplas peças podem ser analisados de forma mais adequada por técnicas DFA. Os paradigmas UML [Blaha e Rumbaugh, 2006] e Orientação ao Objeto [Booch, 1994] são utilizados na análise, desenvolvimento e implementação do sistema.

A representação da informação, ou dados e conhecimento embutidos, com seu intercâmbio associado, é desenvolvida para o domínio e escopo descrito. A representação de todos possíveis casos e combinações possíveis entre dados e conhecimento torna qualquer arquitetura computacional extremamente complexa e, muitas vezes, inviável sob o ponto de vista prático. Este trabalho utiliza uma estrutura mais flexível e completa de representação de dados e conhecimento baseada em *ontologias* [Gruber, 1993, Gruber, 1994b, Gruber, 1993b]. Esta abordagem possui flexilidade adicional devido à utilização e eventual implementação do gerenciamento de conceitos, com sua complexidade prática e teórica, associada a uma arquitetura computacional de projeto mecânico, a qual requer maior abstração na representação de conceitos associados e inerentes a este domínio.

Este requisito demanda uma descrição da informação capaz de, adequadamente, utilizar de forma efetiva, eficiente e eficaz a informação, segundo Chandrasegaran *et al*., 2013. Particularmente, os autores ressaltam a taxonomização de Pahl *et al*., 2007, a qual classifica parte do conhecimento do domínio DFM relacionado com projeto: processos de manufatura como linguístico, simbólico e virtual. Esta taxonomia, análoga em várias outras partes do domínio de projeto e DFM, torna as formas de representação da informação restritas ao dado complexas, senão inviáveis. Desta forma, portanto, torna-se um fato inequívoco que sistemas contextualmente amplos e heterogêneos, como os existentes no

domínio industrial, consequentemente DFM, necessitam de uma forma de representação que inclua logicidade mais completa que a disponível atualmente [Fortineau, Paviot e Lamouri, 2013]. Ahmed e Han, 2013, adicionalmente, ressaltam as possíveis incoerências semânticas advindas da interoperabilidade imposta na integração da informação entre Produto e Manufatura.

Analogamente ao desenvolvimento de arquiteturas sistêmicas mais completas, também ocorreu a utilização do paradigma ontológico de representação e estruturação da informação para configuração de produtos [Yang, Dong e Miao, 2008]. Colace, Santo e Napoletano, 2009 propõem a configuração e estruturação de produtos através de ontologias que relacionam os tópicos inicias desta seção e sugerem uma arquitetura integrada para isto. Especificamente, Wei, Qin-yi e Tian-yi, 2009 desenvolveram um sistema baseado em ontologias para mapear os requisitos de um projeto informados por um cliente e a base contextualizada de conhecimento de manufatura, representada através de ontologias, para projeto de produto. Neste aspecto, Schlenoff, Ivester e Knutilla, 2002 propõem uma solução e abordagem mais abrangente e completa através da definição de uma arquitetura contextual da informação de manufatura do produto e sua integração através ontologias.

Finalmente, embora seja reconhecido que a implementação de uma arquitetura computacional deva possuir boa interação com o usuário através sua interface gráfica, este aspecto não é foco deste trabalho. O foco principal é a definição de uma estrutura consistente e flexível de informação para integração de ferramentes DFM no domínio e escopo descritos. Consequentemente, uma interface gráfica elaborada não é esperada do sistema–protótipo teste.

De forma a contextualizar esta pesquisa, introduzindo-a, uma descrição semântico–funcional da heterogeneidade e inter-relacionamentos inerentes com a informação neste domínio simplificará os conceitos descritos adiante. DFM é, tipicamente, um domínio onde os conceitos envolvidos são, em geral, relacionados com processos baseados em regras e na prática de atividades de manufatura. Logo, os dados (valores) são consequência de sua utilização sob foco informacional em DFM. Outrossim, existe uma relevante heterogeneidade entre as informações requeridas, que variam desde numerais até conceitos "puros". Ademais, informações não são pontuais, ou estanques, e a característica de cada conceito, que eventualmente pode ser descrita por um valor, influencia outro(s) conceito(s), inclusive muitas vezes até o valor de outro(s); este aspecto é ampliado em DFM.

A tabela 1.1 descreve este aspecto em termos de um conceito fundamental em manufatura, o custo, e o quanto a informação torna–se consequentemente mais "rica", ou completa, conforme aspectos informacionais de fabricação são requeridos. Por exemplo, o percentual (%) aproximado descrevendo o custo relativo é fundamentalmente um intervalo difuso baseado em informações que são, por sua vez, guias descrevendo diferentes processos e suas características. Os dados numerais expressos por estas guias classificam

o custo relativo em diferentes tipos. As guias, adicionalmente, também possuem suas inter–relações informacionais com outros conceitos, conforme é possível inferir na tabela 1.2. Este conjunto, ou árvore, de informações inter-relacionadas, se utilizado corretamente, possibilita resolver um determinado problema, além de conhecer efetivamente um domínio e seu escopo.

Tabela 1.1 – Custo de produção relacionado com acabamento superficial [Bralla, 1999].

Tipo de operação	Acabamento μ polegada	Custo relativo aproximado %
Usinagem de desbaste	250	100
Usinagem padrão	125	200
Usinagem precisa, usinagem bruta	63	440
Usinagem ultra–precisa	32	720

Tabela 1.2 – Aspectos informacionais *reais*, conexões intrínsecas embutidas, no domínio DFM que não são representadas inequivocamente - exemplo.

Tipo de operação	Acabamento μ polegada	Custo relativo aproximado %
Usinagem de desbaste	250	100
Usinagem padrão	125	200
Usinagem precisa, usinagem bruta	63	440
Usinagem ultra–precisa	32	720

Considerando somente os termos limítrofes e apenas a necessidade de troca de informações “diretas” em DFM sem sua compreensão que, sem dúvida, é a parte mais relevante e importante para a sua correta organização, a seção 3.1.1 descreve o excesso requerido. No caso de operações, que fazem parte de processo(s) de manufatura, adicionalmente existe, por exemplo, a ordenação do processo em um sistema produtivo. Embora a tabela 1.1 não denote isto, e descreva tipos de processos similares, em geral a programação (sequenciamento) das operações por seu tipo, primário ou secundário, também pode influenciar o custo. Além disto, claramente, existe o requisito de considerar as capacidades dimensionais e de tolerâncias nas máquinas habilitadas para tais processos dentre vários outros aspectos. No caso, a tabela tem somente uma descrição direta, além de considerar a interação de apenas dois conceitos.

Este aspecto também existe e é ressaltado na tabela 1.3, só que em outro conceito, igualmente importante no domínio: materiais. Por óbvio, existe um número bem maior de interações, dependências entre conceitos relacionados, diversos e distintos, reduzidamente descritos, tanto em relação a sua natureza (numerais, conceitos) quanto em relação a sua taxonomia. Por exemplo, inicialmente, basta analisar o aspecto taxonômico: existe um conjunto elevado de conceitos associados que tem pré-requisitos, que por sua vez também tem suas inter relações e associações. Desta forma, a conceitualização de “semelhança”,

também conhecido como uma das características tecnológicas, de um dado material para usinagem demanda mais informação *corretamente contextualizada*[2].

Desta forma foi escolhido, de forma *resumida*, *simplificada e expressa da forma mais genérica possível*, *portanto incompleta sob contexto de informação*, o conceito (classe) da vida útil da ferramenta para denotar "adequacidade" ao processo de usinagem. Ainda que exista esta redução no escopo do domínio desta classe, é possível perceber a necessidade de uma forma de melhor descrever e representar a informação no escopo do domínio. Desta forma, a taxonomização é fundamental para que sejam reduzidas e simplificadas suas intrínsecas inter relações.

Tabela 1.3 – Aspectos intrínsecos, porém não expressos inequivocamente, em materiais sujeitos ao processo de usinagem [ASM INTERNATIONAL, 1990a, ASM INTERNATIONAL, 1990b].

Material	Característica	Característica tecnológica	Detalhe
Alumínio	Não ferroso com alta condutividade	Excelente	Relativamente mais simples processos de usinagem (2011/A92011) e conformação (1100/A91100)
Berílio	Comercialmente puro, sem liga	Boa	Elevadas rigidez e dureza Grande tenacidade
Nióbio	Comercialmente puro, sem liga	Boa	Utilizado em ligas para aumentar a resistência do aço sem comprometer a maleabilidade
Cobre	Liga	Ruim	Material de elevada fragilidade Possui baixa resistência se fundido
Ferro	Aço-carbono e liga	Boa	Variável dependendo do % de Carbono: Percentual baixo (< 15%): ruim Percentual médio/baixo (15% a 30%): boa em geral Percentual médio/médio (30% a 50%): boa Percentagem alta (> 55%): complexa, pode demandar liga de S,Pb
Magnésio	Liga	Excelente	Muito utilizado como elemento muito resistente e leve
Molibdênio	Sem liga (comercialmente puro)	Média	Utilizado em ligas de aços especiais Elevada dureza e resistência
Níquel	Liga	Excelente	Utilizado em ligas, resistente à corrosão Muito resistente 65% de utilização percentual em aço inox
Tântalo	Comercialmente puro, sem liga	Boa	Dureza e ductibilidade elevada Resistente à corrosão
Titânio	Comercialmente puro, liga	Excelente	Elevada resistência mecânica Resistência à corrosão
Tungstênio	Comercialmente puro, sem liga	Ruim	Metal duro, frágil Difícil manufaturabilidade se impuro Acrescenta dureza ao aço em pequenas quantidades
Tungstênio	Sinterizado, ligas	Excelente	Aumenta resistência ao desgaste Utilizado em pastilhas de corte (carbeto de tungstênio)
Zircônio	Comercialmente puro	Boa	Dureza elevada, resistência à corrosão
Zinco	Ligas	Boa	Resistente à corrosão

Em geral, as abordagens tradicionais e típicas de integração ou subestimam esta característica intrínseca das informações em sistemas complexos, como DFM, reduzindo muito sua completeza, ou desenvolvem sistemas de capacidade limitada (sob contexto informacional). A abordagem utilizando ontologias neste domínio, exemplificada no escopo selecionado, pode reduzir esta limitação.

[2]Dentre os vários conceitos associados é possível exemplificar os seguintes: resistência ao corte, energia requerida do maquinário, necessidade de tratamento térmico, resistência térmica tanto do maquinário/ferramenta quanto do material *per se*, além dos aspectos geométricos da peça usinada e as suas consequências no sequenciamento do plano de processo.

1.3 Objetivos da Pesquisa

O objetivo fundamental desta pesquisa é elaborar um arcabouço de informação para integração de componentes de Projeto Orientado para a Manufatura (DFM) conectada a uma arquitetura CAD de forma a melhor suportar o processo de desenvolvimento integrado de produtos. Especificamente, os objetivos de pesquisa incluem,

- compreensão das necessidades de ambientes de projeto integrado com foco em DFM.
- definição de uma arquitetura flexível de informação passível de integração efetiva e simplificada em uma arquitetura de projeto com foco em DFM.
- desenvolvimento e implementação de uma arquitetura, gerenciamento e interpretação de informação para o domínio DFM provendo serviços efetivos, flexíveis e expansíveis para integração de ferramentas DFM.

A efetivação destes objetivos na pesquisa utiliza as seguintes etapas,

1. Identificação e definição de aspectos estruturais e funcionais dos requisitos associados à informação DFM integrada.
2. Identificação e definição das formas de representação da informação, mais especificamente o *schemata* de dados e conhecimento do domínio DFM selecionado, bem como sua integração com uma arquitetura DFM.
3. Identificação e definição de um conjunto mínimo expansível de serviços de integração da informação para ferramentas DFM.
4. Implementação de um conjunto de ferramentas–exemplo prototípicas no domínio DFM.

1.4 Organização do Documento

Esta pesquisa é organizada em seis capítulos. A premissa básica desta pesquisa foi definida. A problema foi formalizado e uma solução proposta considerando seus requisitos básicos. Finalmente, a pesquisa foi restrita e seu escopo definido.

Os capítulos 2 e 3 disponibilizam a revisão da literatura nos aspectos principais desta pesquisa. Genericamente, uma pesquisa associada com estrutura de informação integrada com distintos componentes contextuais é bastante ampla. Desta forma, estes capítulos são subdivididos de acordo com as principais áreas relevantes a este trabalho: arquiteturas genéricas, sistemas integrados de projeto e arquiteturas DFM no capítulo 2, enquanto aspectos relacionados com integração e intercâmbio da informação: história,

detalhes sobre a troca da informação, estrutura, representação e utilização de informações contextuais são descritos no capítulo 3.

O capítulo 4 define a pesquisa. Os componentes estruturais e funcionais da abordagem proposta de gerenciamento da informação para integração de componentes DFM são detalhados e explicados. Finalmente, o método de avaliação é descrito. A implementação do protótipo–exemplo da pesquisa em uma arquitetura DFM é descrita no capítulo 5. As características, capacidade e limitações do protótipo são descritas e explicadas.

Finalmente, o sumário da pesquisa, sua contribuição e limitações, bem como sua direção futura são descritas no último capítulo deste trabalho, capítulo 6. Alguns dos apêndices descrevem, respectivamente, a estrutura de ontologias, apêndice A, descrição dos conceitos expressos em termos ontológicos, apêndice B, e informações de suporte desta pesquisa em Lógica Modal e Taxonomia de projeto, anexo A.

2 DOMÍNIO DE PROJETO, PRODUTO E DFM

Este capítulo apresenta a revisão da literatura relacionada com o domínio desta pesquisa, primeira parte demandada quando do desenvolvimento da um sistema de informação. As abordagens existentes/disponíveis em cada tópico são descritas criticamente e colocadas no contexto deste trabalho. A primeira seção faz uma revisão de sistemas arquitetônicos em geral incluindo duas abordagens historicamente relevantes nesta pesquisa: *Arquitetura Blackboard* e *Arquitetura de Agentes*. A perspectiva histórica deste desenvolvimento é apresentada juntamente com análise de arquiteturas sistêmicas, tanto de propósito geral quanto específico.

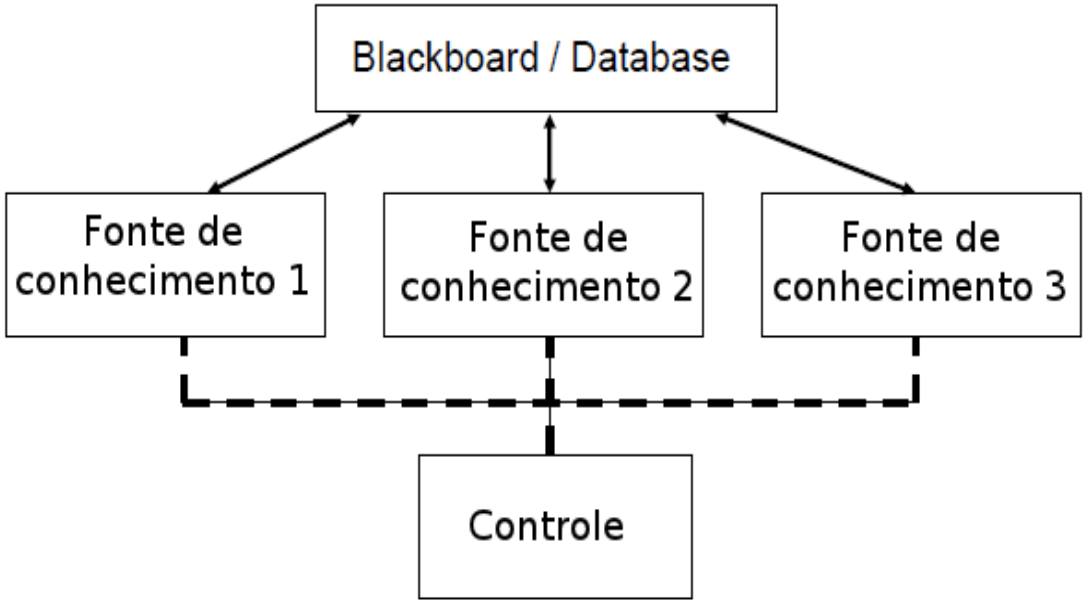
Após uma introdução ao domínio desta pesquisa, a primeira e a segunda seções (2.1 e 2.2 fazem uma revisão sobre sistemas integrados com foco em projeto. As abordagens comuns de integração nestes sistemas são categorizadas e soluções tanto acadêmicas quanto comerciais são descritas. A terceira seção, seção 2.3, descreve algumas arquiteturas sistêmicas integradas de projeto disponíveis, enquanto a última seção revisa sistemicamente arquiteturas de Projeto Orientado para Manufatura (DFM).

2.1 Arquiteturas de Projeto

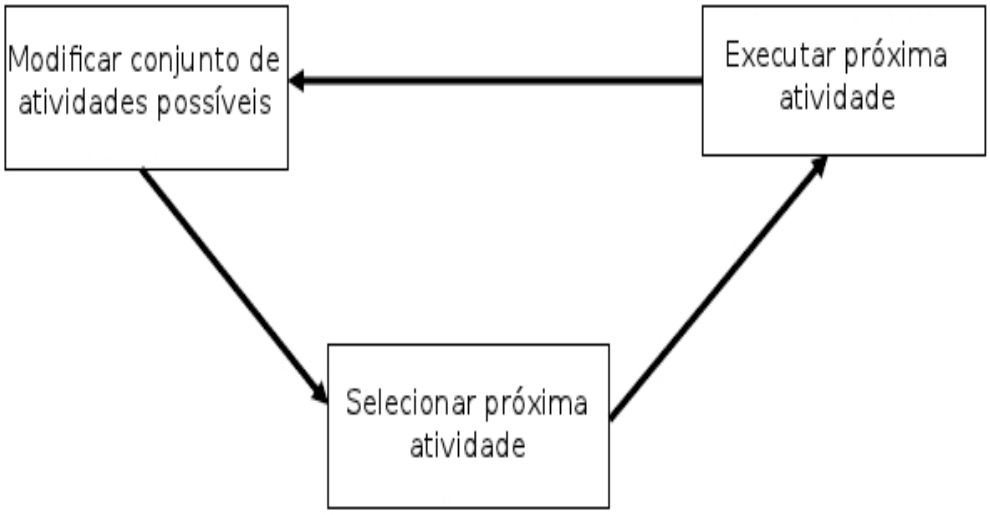
Esta seção apresenta uma revisão das arquiteturas de projeto relevantes disponíveis na literatura atualmente. O objetivo principal é apresentar como as arquiteturas de projeto evoluíram conforme a tecnologia disponível na época ao invés de uma descrição completa de cada arquitetura em cada área de conhecimento. As únicas exceções a este foco são as revisões baseadas nos paradigmas *Blackboard* (Quadro–negro) e Agentes com o objetivo de descrever a flexibilidade destas arquiteturas na área de projeto.

As figuras 2.1a, 2.1b e figura 2.2 mostram descrições simplificadas das arquiteturas *Blackboard* (gerenciamento centralizado) e de Agentes (gerenciamento distribuído), respectivamente. A figura 2.2 mostra, de forma resumida, a diferença estrutural e dinâmica da arquitetura de agentes de componentes de software. A arquitetura de Agentes, utilizada nesta pesquisa, é descrita em maiores detalhes no capítulo 4. Adicionalmente, a seção 2.2.2 descreve algumas implementações teórico–práticas desta abordagem de integração sistêmica.

As arquiteturas descritas nas próximas subseções possuem três aspectos fundamentais em relação a sua implementação computacional fundamental: coerência com código legado, compatibilidade entre rotinas de programas computacionais e flexibilidade de integração com outras rotinas e programas computacionais.



(a) Arquitetura *Blackboard*.



(b) Ciclo de controle *Blackboard*.

Figura 2.1 – Arquitetura Quadro–negro.

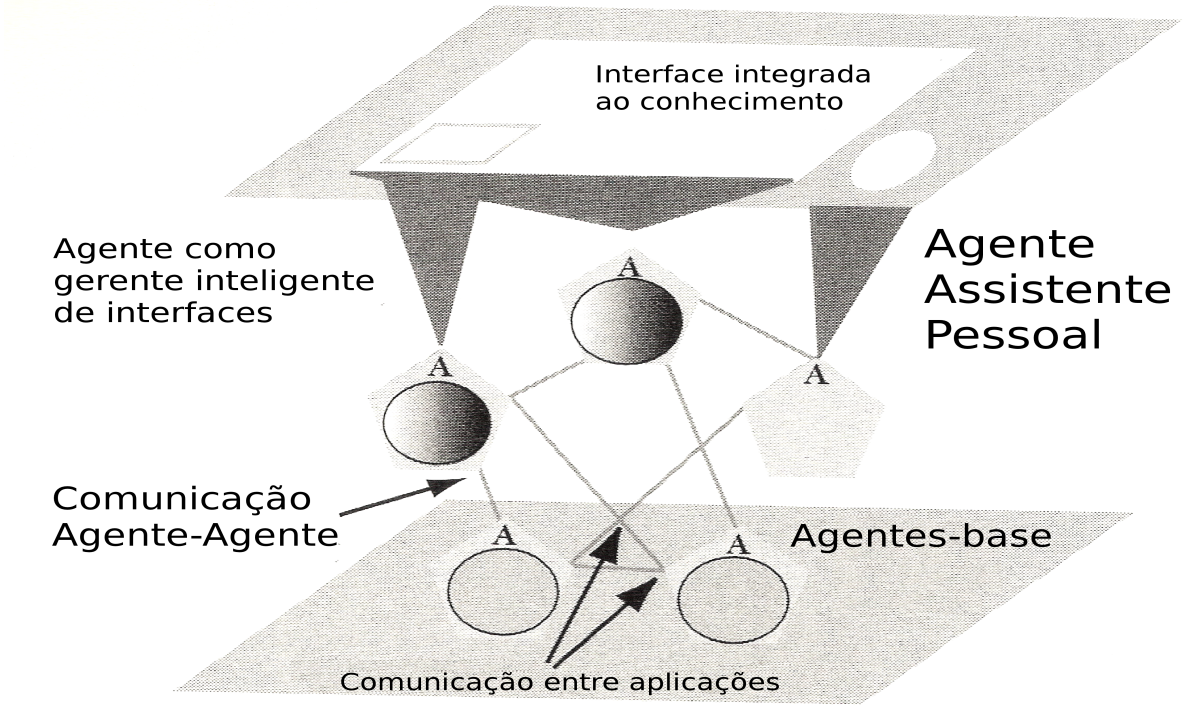


Figura 2.2 – Arquitetura baseada em Agentes, adaptada de Bradshaw [Bradshaw, 1997].

2.1.1 Perspectiva Histórica

Estruturas computacionais que objetivam e/ou permitem funcionalidade para resolução de um dado problema, ou conjunto de problemas, tem tido foco desde os primórdios da revolução computacional. O desenvolvimento de ferramentas computacionais tradicionais também era limitado devido à falta de *hardware* disponível juntamente com as ferramentas de desenvolvimento de *software* e protocolos associados de comunicação e informação. Inicialmente, uma arquitetura era interpretada meramente como um repositório de tradutores entre uma série de ferramentas e sua função principal era apenas disponibilizar troca de informações (limitado a dados sob o contexto histórico) [Wolf, 1994].

O segundo modelo de arquitetura introduziu o conceito de um repositório comum de dados [Katz, 1982]. Este modelo refletiu uma necessidade real de armazenamento de dados de projeto e a popularidade crescente com *disponibilidade* de tecnologia de banco de dados daquele momento. Desta forma, uma arquitetura tornou–se um gerente de dados de projeto sendo responsável pelo gerenciamento tanto de componentes quanto de dados globais de projeto [Katz, 1985].

2.1.2 Implementação de Arquiteturas de Projeto

Esta seção apresenta uma revisão de arquiteturas e sua implementação, estando subdividida em duas subseções com foco em projeto: propósito geral e propósito específico. A subseção de propósito específico tem por foco um domínio individual como, por exemplo, projeto mecânico ou elétrico.

2.1.2.1 Propósito Geral

Os fundamentos iniciais no desenvolvimento de uma arquitetura computacional genérica completa para integração de diferentes ferramentas de *software* deve–se a Nii, 1986 com a definição formal dos requisitos de integração do modelo de arquitetura quadro–negro (*blackboard*). Entretanto, o fundamento teórico desta arquitetura é relacionado com as pesquisa pioneiras de Germano [Erman e Lesser, 1975], Fennel, 1975 e Nii e Aiello, 1978.

As primeiras implementações do paradigma da arquitetura *blackboard* foram os sistemas BB1 [Hayes-Roth, 1985], MXA [Tailor, 1988] e BLOBS [Zanconatto, 1988]. A segunda geração de arquiteturas *blackboard* era composta pelos sistemas GBB [Gallagher, Corkill e Johnson, 1988], Erasmus [Baum, Jagannathan e Dodhiawala, 1989], GEST [Gilmore, Roth e Tynor, 1989] e ATOME [Lâasri *et al*., 1988]. Carver, 1997 disponibiliza uma revisão adequada, consistente e coerente sobre como um sistema *blackboard* funciona.

Woyak, 1995 desenvolveu uma arquitetura rigidamente ligada orientada a objetos para integração de componentes de *software* chamada Sistema de Integração Dinâmica (*Dynamic Integration System*). A arquitetura possui uma abordagem inédita para integração de *software* e é desenvolvido na linguagem C++. A abordagem é fundamentada em uma árvore de dependência entre parâmetros para representar as ligações entre diferentes componentes de *software* [Woyak e Myklebust, 1998].

Os parâmetros representam os requisitos das informações de entrada (*input*) entre módulos do *software* e seus inter–relacionamentos. Desta forma, o gerenciamento de dados entre módulos de *software* é possível. Neste caso, a funcionalidade de cada módulo de *software* é abstraída através de seus dados de entrada e saída. A figura 2.3 mostra uma típica árvore de dependência entre parâmetros.

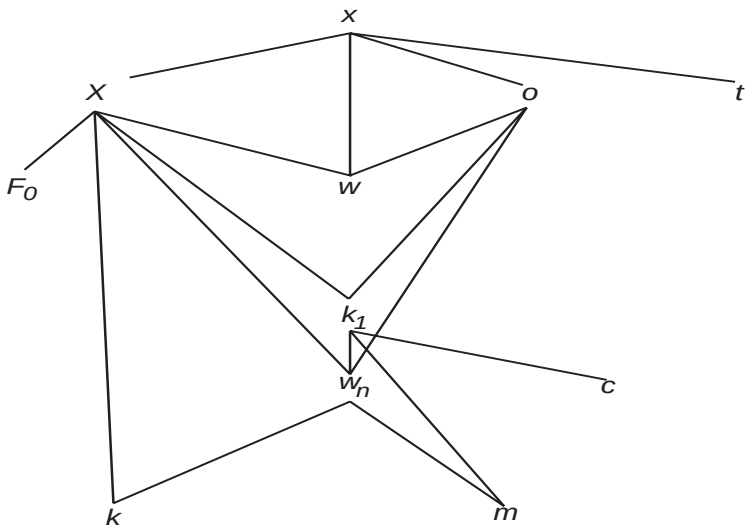


Figura 2.3 – Árvore de dependência entre parâmetros.

As arquiteturas FIPA e MicroFIPA são desenvolvidas a partir de agentes rigidamente ligados que proporcionam tanto uma linguagem para desenvolvimento, **Java**, quanto

um sistema para desenvolvimento e integração de *software* com sistemas e/ou programas desenvolvidos pelo usuário.

2.1.2.2 Propósito Específico

Esta seção descreve arquiteturas de propósito específico, ou seja, estruturas que foram desenvolvidas para um determinado domínio de conhecimento. O foco desta seção é disponibilizar uma descrição genérica das arquiteturas em análise ao invés de análise detalhada das características, requisitos e necessidade de cada aplicação específica de cada domínio.

Sapossnek *et al*., 1989 desenvolveram um sistema CASE (*Computer Aided Software Engineering*: Engenharia de *Software* Assistida por Computador), baseado em uma arquitetura de agentes para integração em projeto mecânico. Neste sistema existem dois tipos de agentes no contexto de projeto mecânico: agentes de projeto e agentes críticos de projeto. Esta foi uma das primeiras arquiteturas baseadas em agentes contextualmente, embora não utilizasse os princípios básicos de arquitetura de agentes como estruturados atualmente devido a aspectos temporais[1].

O sistema *First–Cut* e seu sucessor *Next–Cut*, desenvolvidos por Brown, Cutkoski e Tenenbaum, 1989, disponibilizam uma arquitetura fundamentada em um ambiente de conhecimento com o objetivo de desenvolvimento rápido de produtos em projeto mecânico. Os agentes disponíveis efetuam tanto tarefas detalhadas de projeto como, por exemplo, cálculo através de elementos finitos, quanto tarefas como planejamento de processo.

Guo *et al*., 1993 desenvolveram a arquitetura denominada Ambiente Inteligente Integrado de Projeto – IIDE (*Integrated Intelligent Design Environment*). A arquitetura é subdividida em dois subsistemas: subsistema de caracterização e subsistema de gerenciamento de projeto.

Mo, Fernandez e Llang, 1993 desenvolveram uma arquitetura para integração de atividades de projeto completamente baseada no paradigma de orientação a objetos. A arquitetura é fracamente–ligada, ou fracamente–conectada (*loosely–coupled*), e depende do *input* de um sistema CAD previamente desenvolvido.

A arquitetura–teste Colaborativa Palo Alto (PACT) [Cutkoski *et al*., 1993], ou *Palo Alto Collaborative Testbed*, foi desenvolvida como uma arquitetura para testar aplicações de Engenharia Simultânea. PACT foi desenvolvida como um sistema, ou federação, fracamente–conectado de agentes onde cada módulo de *software* pode ser interpretado como um agente e pode se comunicar com outros agentes através de módulos (ou agentes) facilitadores. Nesta arquitetura um repositório comum de dados e conhecimento não existe.

[1]Nas referências [Sapossnek *et al*., 1989], [Brown, Cutkoski e Tenenbaum, 1989], [Cutkoski *et al*., 1993] e [Shen e Barthes, 1994] a palavra agentes não necessariamente se refere a estrutura formal atual de agentes de *software*.

Ao invés disto, todos dados são armazenados em bancos de dados locais. A arquitetura gerencia toda atualização e intercâmbio de mensagens através dos facilitadores.

A arquitetura PACT também serviu como protótipo-teste para outras tecnologias, por exemplo a linguagem comum de agentes. Embora protocolos de comunicação como TCP/IP, Opendoc, OLE e CORBA disponibilizem a forma e meio de troca de informações em ambientes distribuídos, existe muito pouca informação sobre o conteúdo de informação. A arquitetura PACT, que possui troca de informações no nível de conhecimento ao invés de somente dados, requer informação mais completa para intercâmbio útil e efetivo.

O domínio PACT original foi o projeto de um manipulador robótico. Os principais agentes foram *Next–Cut* (sistema de projeto mecânico e planejamento do processo), *Designworld* (sistema de projeto, simulação, montagem e teste eletrônico), *NVisage* (ambiente de integração baseado em conhecimento) e *Device Modeling Environment* – DME (ambiente de simulação). Estes sistemas foram utilizados para projetar, modelar e simular um manipulador robótico.

Shen e Barthes, 1994 desenvolveram o sistema DIDE, que é uma arquitetura de rede de agentes para integração de agentes de informações de projeto. Cada agente tem sua própria interface e uma interface comum de rede. Esta importante característica é disponibilizada pela linguagem Java para facilitar a integração entre todos agentes da arquitetura.

Kuokka e Livezey, 1995 desenvolveram uma arquitetura de projeto colaborativo, ParMan, baseada nos conceitos iniciais propostos pela arquitetura–teste PACT. ParMan é baseado em projeto paramétrico e utiliza um paradigma de programação baseado em lógica de restrições associado com as características de comunicação entre agentes como KQML e KIF. Adicionalmente possui uma interface baseada em grafos para definição das restrições dos parâmetros. ParMan foi utilizado para o projeto preliminar de um satélite.

O domínio de projeto eletrônico foi um dos pioneiros na pesquisa de arquiteturas de integração de componentes de *software*. Neste domínio, o pioneirismo de Katz, 1985 utilizando um banco de dados como meio de integração entre vários componentes de *software* para projeto de *chips* VLSI é digno de destaque. Exemplos relativamente mais recentes de integração através de banco de dados incluem a utilização do paradigma de orientação a objetos [Fox *et al*., 1990, Heijenga, Jasnoch e Radeke, 1992]. O sistema Ulisses é um exemplo de arquitetura de projeto eletrônico mais completo baseada no paradigma do quadro–negro (*blackboard*) [Brockman e Director, 1992, Bushnell e Director, 1989].

Arquiteturas eletrônicas de pesquisa adicionais incluem o sistema Version Server [Katz, 1983] (gerenciador teórico de dados de projeto) e o sistema Oct/VEM [Harrison *et al*., 1986] (integração centrada em banco de dados rigidamente ligada) com seu sucessor Octane da Universidade da Califórnia, Berkeley. A arquitetura Roadmap é um gerenciador teórico de dados do processo de projeto desenvolvido pela Phillips Research [Hammer e

Treffers, 1990] para controle do fluxo de dados entre ferramentas CAD.

2.1.2.3 Arquiteturas Comerciais de Projeto

A pioneira arquitetura genérica desenvolvida foi o sistema GBB comercializado pela empresa Blackboard Technology Group Inc. Sua versão comercial possui módulos de controle intrínsecos bem como interface para módulos desenvolvidos em outras linguagens de programação. Adicionalmente, utiliza Lisp Orientado a Objetos através do CLOS sendo a arquitetura *blackboard* mais flexível no mercado atualmente. GBB tem versões disponíveis para múltiplas plataformas (Macintosh, PC, estações de trabalho UNIX), integração com bancos de dados comerciais, conectividade distribuída em rede pela Internet e um gerador de interface com o usuário. Uma versão gratuita também existe: GBBopen[2].

A arquitetura baseada em conhecimento *Intelligent CAD Framework* (ICAD) [Concentra, 1997] é uma arquitetura CAD para integração de aplicativos de engenharia desenvolvida inicialmente pela ICAD Inc. e posteriormente pela Concentra, Inc. ICAD tem uma abordagem baseada em conhecimento para atividade de projeto e integração de sistemas. A arquitetura Catalyst [Software Productivity Solutions, 1997] integra componentes sistêmicos através de um banco de dados utilizando CORBA. A arquitetura Cadence, desenvolvida pela Cadence Design Systems e Digital Equipment Corporation, é baseada em múltiplas camadas permitindo diferentes níveis de abstração para aplicativos customizados [CADENCE, 1990]. Adicionalmente, é relevante ressaltar que existem arquiteturas de domínio específico especificamente na área de projeto eletrônico.

2.2 Sistemas Integrados

A introdução desta seção descreve sistemas com determinadas características de sistemas *blackboard* e/ou agentes sob o contexto histórico–temporal, desta forma a nomenclatura não necessariamente é igual à definição utilizada atualmente. A arquitetura de agentes **Jess** [Friedman-Hill, 2003] é um sistema especialista gratuito disponível para pesquisa sobre sistemas especialistas. Através da utilização da interface da ferramenta (ou *shell*) é possível compreender intrinsecamente o que é um agente Jess. Adicionalmente, é possível estender as características básicas de agentes genéricos e Jess, especificamente. Estas características permitem utilizar efetivamente um agente Jess, ou seja, fazer que um agente realize seu objetivo.

O sistema **eMarkets** [NORTEL, 2000d] possui diversos agentes configurados como compradores ou vendedores em diferentes níveis arquitetônicos. O aspecto mais interessante do sistema **eMarkets** é a possibilidade de um agente ser definido (programado) como comprador ou vendedor ao mesmo tempo.

[2] <http://gbbopen.org/>

O agente **Ping** [NORTEL, 2000c] emula o popular comando **`ping`** disponível em sistemas UNIX comerciais ou livremente disponíveis. Neste caso, um agente **`ping`** é criado para cada conexão efetuada. Um dos aspectos mais interessantes deste sistema é a interconexão de tarefas de comunicação entre tarefas–mãe e tarefas-cliente. Desta forma, a funcionalidade do agente é subdividida em processos distintos de envio e recepção de mensagens de uma tarefa.

O agente **Search** [NORTEL, 2000b] auxilia usuários a encontrar outros agentes em um sistema computacional bem como cria uma tarefa para interação com estes agentes. O agente **Generic** [NORTEL, 2000a] é uma ferramenta–base que permite a definição de um agente "vazio" utilizando o sistema de agentes FIPA OS. Adicionalmente, possibilita o fundamento de como registrar um agente em uma plataforma local através da interface de programação da aplicação (*API*) do agente.

2.2.1 Sistemas Integrados Comerciais de Projeto

Esta subseção descreve os principais sistemas integrados comerciais de projeto atualmente disponíveis no mercado. O foco desta revisão é nos aspectos arquitetônicos ao invés de características específicas de cada sistema de projeto. Adicionalmente, os aspectos de integração relacionados com esta pesquisa são argumentados sob este contexto.

O sistema CATIA/CADAM [IBM Inc., 1997], desenvolvido pela IBM[3], é um sistema modular de projeto cuja flexibilidade e capacidade de integração são fortemente dependentes de sua modularidade. Neste contexto, o sistema é customizável e flexível se os módulos adicionais forem adquiridos da IBM ou de seus parceiros oficias de *software*. A versão básica do sistema CATIA possui funcionalidades de projeto adicionais mais avançadas que outros sistemas CAD focados somente em aspectos tradicionais de modelagem. A integração de módulos customizados no CATIA/CADAM é relativamente complexa, entretanto é possível acesso aos dados internos do sistema.

O sistema de projeto Pro/Engineer, desenvolvido pela Parametric Technology Corporation (PTC) [PTC, 2012, PTC, 2001, PTC, 2004a, PTC, 1999][4],possui características similares ao sistema CATIA/CADAM. Entretanto, o sistema é baseado em uma fundação mais completa permitindo a utilização de características mais avançadas de projeto a partir de sua versão básica. Adicionalmente, a PTC também comercializa uma série de produtos de projeto relacionados para necessidades específicas, tais como projeto mecânico e industrial, CAD/CAM, montagem, simulação, visualização e inclusive até matemática, mais recentemente.

Esta plataforma disponibiliza uma biblioteca de integração para acesso ao dados de projeto utilizando uma API comum. Esta conexão é feita através da linguagem de progra-

[3]Posteriormente, uma parceria entre a IBM e a Dassault Systèmes foi estabelecida.
[4]<http://www.ptc.com>.

mação Java utilizando a biblioteca J/Link [PTC, 2004b]. A disponibilização de *hardware* mais potente associado a avanços em programação permitiram que a plataforma Pro/Engineer fosse atualizada várias vezes a partir de sua primeira versão comercial principal, versão 2001. O sistema principal subsequente, denominado Wildfire, possuiu diversas versões até a sua última versão 6. O sistema Pro/Engineer é relevante arquitetonicamente logo é utilizado neste trabalho como conexão ao CAD. Em 2011, foi disponibilizada pela PTC uma nova arquitetura CAD sistêmica: CREO. Paralelamente ao fato de novas características mais poderosas computacionalmente terem sido introduzidas com uma nova interface, existe um direcionamento bastante específico com a interoperabilidade entre os componentes da arquitetura. Esta é uma direção e estratégia historicamente relevante nesta arquitetura. A informação e seu gerenciamento é, portanto, um aspecto–chave na arquitetura da PTC.

A PTC também comercializa produtos para integração de seus aplicativos; estes componentes possuem foco em intercâmbio de informação. O objetivo é uma arquitetura de ciclo de vida do produto (PLM). Inicialmente foi disponibilizado o produto Pro/Intralink focado quase exclusivamente somente em dados. Posteriormente foi introduzida a arquitetura Windchill que possui características de integração e utilização da informação mais amplas, ou seja, em um contexto PLM. A arquitetura Windchill permite inclusive customização da utilização da informação através de arquivos de lote (*scripts*) para automatizar atividades repetitivas. Uma vantagem da arquitetura Pro/Engineer é sua capacidade multi–plataforma, que permite que a administração de projetos seja feita sobre um conjunto diferente de plataformas de *hardware*. Neste contexto, a arquitetura do conjunto de programas da PTC é a contextualmente mais adequada disponível da área de projeto em relação a esta pesquisa.

O sistema Unigraphics [UGS, 2006b], desenvolvido originalmente pela UGS (atualmente Siemens e denominado NX na versão 7.5), é o principal concorrente da arquitetura do sistema de projeto Pro/Engineer. Entretanto, seus componentes, em geral, são mais rigidamente interconectados com sua arquitetura global. O sistema possui excelente integração de *software*. A capacidade de integração de pacotes adicionais é mais limitada que o Pro/Engineer. A Siemens também possui uma solução PLM para gerenciamento de todo processo de projeto [UGS, 2006a], que inclusive é comercializada como solução completa para projeto mecânico.

O sistema NX possui um excelente modelador de sólidos com módulos específicos completos em diferentes domínios de projeto mecânico, juntamente com PLM [UGS, 2006b]. Adicionalmente, a arquitetura possui um produto PLM popularmente considerado entre os melhores disponíveis atualmente devido a sua abstração do processo de projeto. Entretanto, se a arquitetura geral for o principal parâmetro de análise, a solução baseada no Pro/Engineer da PTC é superior.

O sistema de projeto I–DEAS Master 5 [SDRC, 1997a, SDRC, 1996], desenvolvido pela

Structural Dynamics Research Corporation (SDRC), possui características de integração bem definidas e disponíveis associadas com uma arquitetura customizável de modelagem de dados de produto (PDM). Ele permite aplicações integradas em diversos domínios embora seu foco principal seja projeto mecânico. As aplicações típicas de projeto mecânico disponíveis incluem montagem, integração CNC, prototipagem rápida, mecanismos e chapas de metal. Estas aplicações podem ser adaptadas em uma solução final customizada para um cliente específico. A arquitetura sistêmica I–DEAS permite integração simplificada de módulos customizados específicos através de uma API padrão.

A SDRC desenvolve e comercializa um produto na área de modelagem de dados de empresas denominado Metaphase 2 [SDRC, 1997b]. No contexto deste produto são disponibilizadas características de uma arquitetura sistêmica com foco em desenvolvimento de produto. Sua arquitetura é objeto–orientada distribuída, híbrida no modelo cliente–servidor e interligada com um banco de dados relacional. O sistema Methapase é consistente e poderoso em geral, entretanto possui limitada possibilidade de customização do controle no processo de desenvolvimento de produto.

O *software* AutoCAD, desenvolvido pela Autodesk Inc. [AUTODESK Inc., 1997, AUTODESK Inc., 1996], é provavelmente o sistema integrado de projeto mais popular atualmente. Originalmente, foi disponibilizado como um sistema puramente de desenho CAD com módulos embutidos de customização em uma versão reduzida da linguagem Lisp, intrínseca do sistema, bem como através de interfaces mais básicas com módulos integrados na linguagem C, através de sua API. No domínio de projeto mecânico existe um modelador de sólidos denominado Inventor. Em geral, a Autodesk produz um conjunto de aplicações de suporte em diferentes domínios, tanto por si própria quanto associada a outras empresas. Esta característica das soluções da empresa permitiu que ela seja a líder do mercado CAD de pequeno e médios clientes.

O *software* CATIA [DASSAULT, 2006a], desenvolvido pela Dassault Systems, é um dos principais concorrentes no mercado global de CAD avançado. O sistema CATIA tem grande capacidade de integração entre pacotes embora sua arquitetura seja rigidamente conectada. A arquitetura CATIA inclui, de forma equivalente, uma solução PLM denominada ENOVIA [DASSAULT, 2006b].

Adicionalmente, empresas terceirizadas produzem componentes rigidamente integrados a sistemas CAD. Entretanto, quando comparado aos sistemas previamente descritos, eles oferecem características para integração mais restritas para usuários finais. Também é relevante descrever que, em geral, o fluxo da informação e, consequentemente, dos processos é centrado no documento.

Duas empresas que desenvolvem *software* neste contexto são relevantes destacar: EdgeCAM [EdgeCAM, 2005] e OneCNC [OneCNC, 2005]. O *software* EdgeCAM é relativamente consistente porém mais simples comparativamente que o Pro/Engineer, NX, ou

I–DEAS, por exemplo; existem bons módulos tais como um Editor de Controle Numérico (*Numerical Control* – NC) associado ao modelador de sólidos. As principais qualidades do *software* OneCNC estão relacionadas a seus módulos embutidos de Fresagem, Furação e Manufatura. Adicionalmente, possui ótima interface bastante similar ao estilo do sistema operacional Windows™, juntamente com operações detalhadas de modelagem e usinagem, fazendo que seja a melhor solução dentro de seu nível de custo. Alguns aspectos de manufatura são superiores, inclusive ao Pro/Engineer, se a facilidade de uso for considerada. Ambas soluções são ótimos modeladores com aspectos de fabricação/usinagem consistentes, possibilitando boas soluções para iniciantes nesta área.

O rápido crescimento e disponibilização do poder computacional interligado com a teoria permitiu o desenvolvimento de uma nova classe de sistemas de projeto que englobem todo o seu processo. O sistema ModelCenter, desenvolvido pela PHOENIX INTEGRATION, 2012[5], faz parte desta classe de tipos de solução com grande capacidade de customização, interligando à integração com sistemas computacionais da área. Neste caso, entretanto, é relevante destacar que, enquanto a flexibilidade em customização é elevada, ainda persiste uma relativa dependência em relação ao desenvolvedor da solução.

O *software* ModelCenter [Malone, 2012] é um ambiente visual de suporte e integração no processo de projeto. Ele pode integrar tanto *software* novo quanto legado. As três características principais são um Gerenciador de Restrições (*Constraint Management*), que coordena a forma como os dados são compartilhados entre programas componentes, Servidor de Análise (*Analysis Server*™), que incorpora componentes de *software* de análise como componentes reutilizáveis que podem ter seus resultados disponibilizados em uma rede independente da plataforma (como a Internet, por exemplo) e Otimização, que consiste de um algoritmo de otimização baseado no gradiente permitindo que projetistas criem problemas restringidos para obtenção da melhor solução. Adicionalmente, possui uma arquitetura *Plug–In* flexível permitindo a integração de uma nova funcionalidade customizada ao ModelCenter, consequentemente facilitando a integração como outros algoritmos.

A empresa Engineous desenvolve o *software* iSIGHT [Engineous, 2012] que integra aspectos–chave no processo de projeto de produto, automatizando–os e executando suas etapas–chave. Dentre as ferramentas de análise tipicamente utilizadas no processo de projeto estão otimização, técnicas de Projeto de Experimentos (DOE) e Projeto por Seis Sigma (DFSS), que permitem agregar valor real no resultado para os clientes. Adicionalmente, também existe integração relativamente direta com soluções comerciais.

[5] <http://www.phoenix-int.com>

2.2.2 Sistemas de Projeto baseados em Agentes

Historicamente, as arquiteturas de software iniciaram estáticas e assim permaneceram até a arquitetura de quadro–negro. A figura 2.4 descreve contextualmente a dinamicidade de uma arquitetura de agentes de *software*. A arquitetura de agentes de *software* permite a contextualização estruturada e definida do processo dinâmico de projeto. Esta seção descreve, resumidamente, alguns dos sistemas de projeto típicos relevantes em seus respectivos domínios (vide figura 2.5).

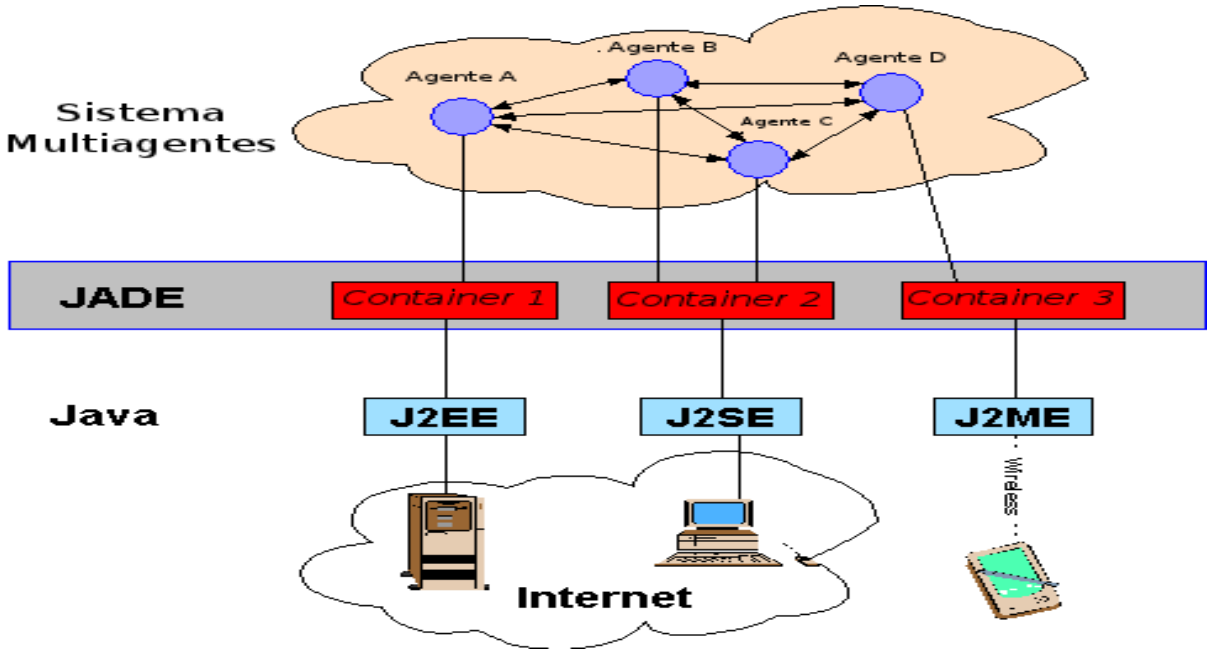


Figura 2.4 – Arquitetura de Agentes baseada em JADE [Bellifemine, Caire e Greenwood, 2007].

Huang, Huang e Mak, 2000 utilizaram agentes na tradicional atividade multi–disciplinar colaborativa do fluxo de projeto, na pesquisa denominada TeleDSS. Neste caso, agentes são utilizados para o gerenciamento do fluxo distribuído de conteúdo e informação entre membros de um projeto que podem, inclusive, serem geograficamente distribuídos. O fluxo de informações é representado através de interconexões no formato de um grafo, ou rede. Os nós da rede correspondem a atividades, ou elementos de trabalho, e as arestas, ou conexões entre nós, perfazem os fluxos da informação entre elementos de um projeto. Neste contexto, na verdade, agentes são utilizados como facilitadores do processo de gerenciamento da informação de um projeto.

A arquitetura proposta é do tipo cliente–servidor onde os agentes estão em um repositório, que por sua vez é disponibilizado através de servidores Web. As decisões dos usuários relativas a um determinado projeto são armazenadas em um banco de dados para que sejam disponibilizadas a outros agentes no sistema. A principal vantagem desta arquitetura é a separação entre o gerenciamento do fluxo da informação em projetos e seu conteúdo. A arquitetura de informação proposta neste trabalho também pode utilizar

esta abordagem eventualmente pois, de forma análoga, existem diversos componentes integrados com necessidade intrínseca de intercâmbio da informação.

Frost e Cutkosky, 1996 introduziram um arcabouço DFM fundamentado em agentes de *software* através da Internet. Tipicamente, DFM demanda que, idealmente, especialistas de todas áreas tenham interação adequada. A utilização desta arquitetura permite que enquanto uma peça esteja sendo projetada a partir do seu fundamento geométrico, regras de projeto e regras prático/teóricas de manufatura sejam dinamicamente integradas. A integração destes aspectos através de agentes permite que projetistas adquiram conhecimento inclusive da área de expedição final, por exemplo. O domínio utilizado é o desenvolvimento de modelos–prova de produtos através de Prototipagem Rápida.

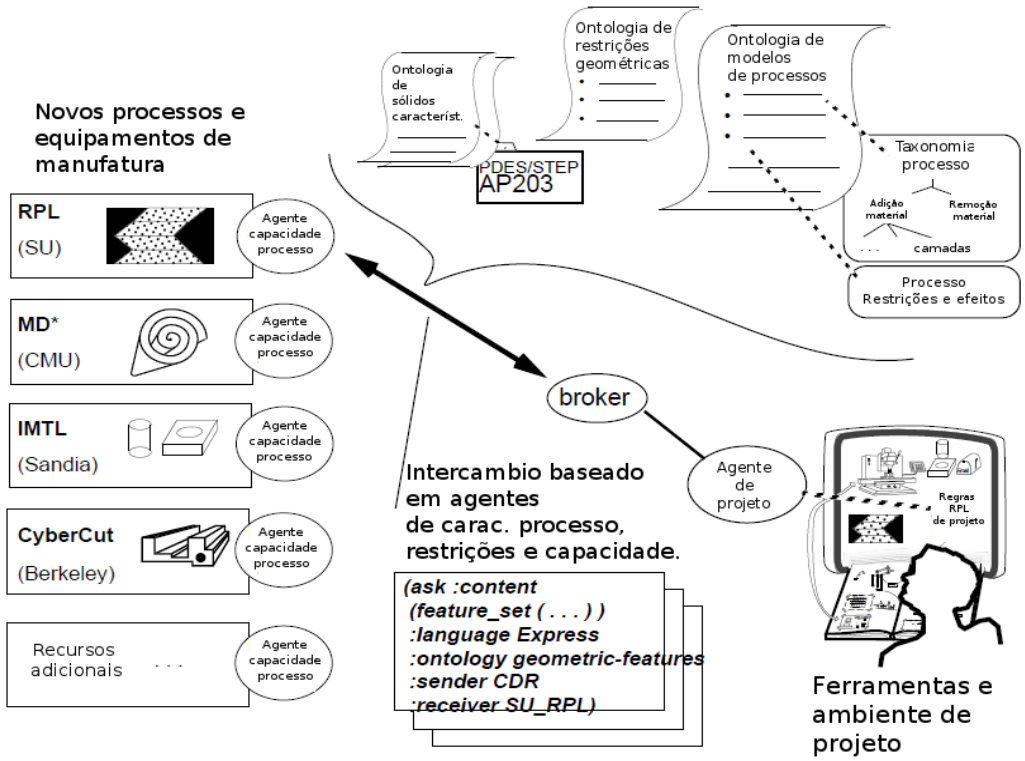
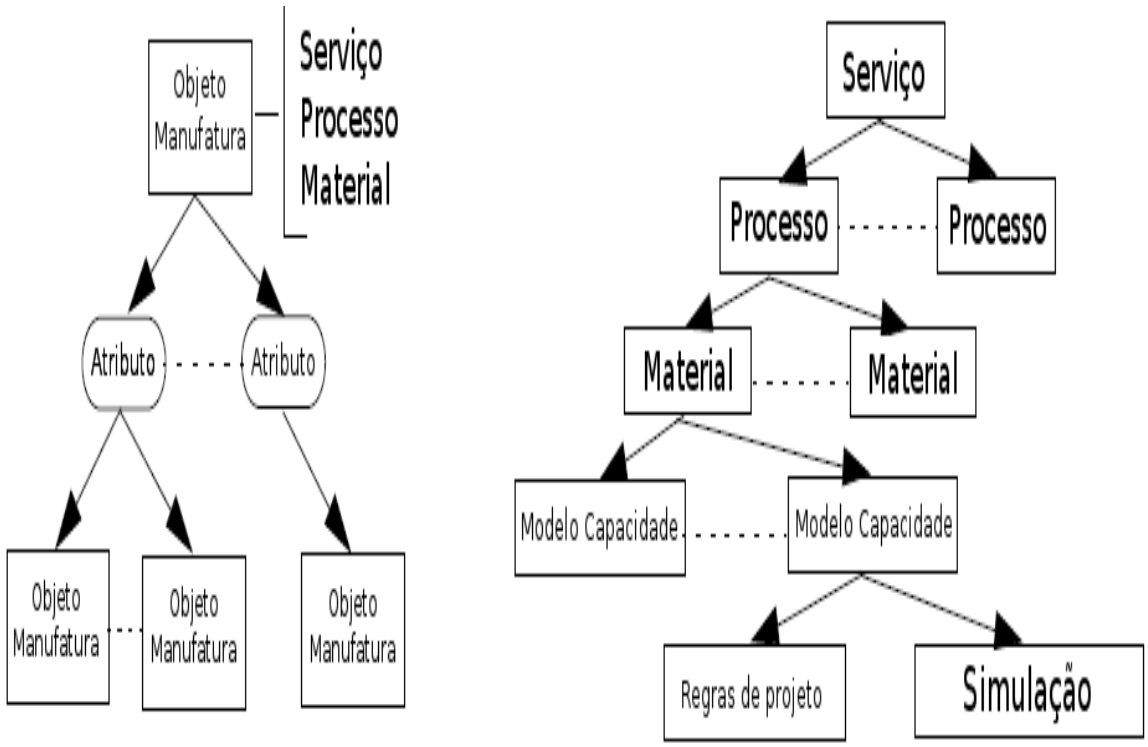


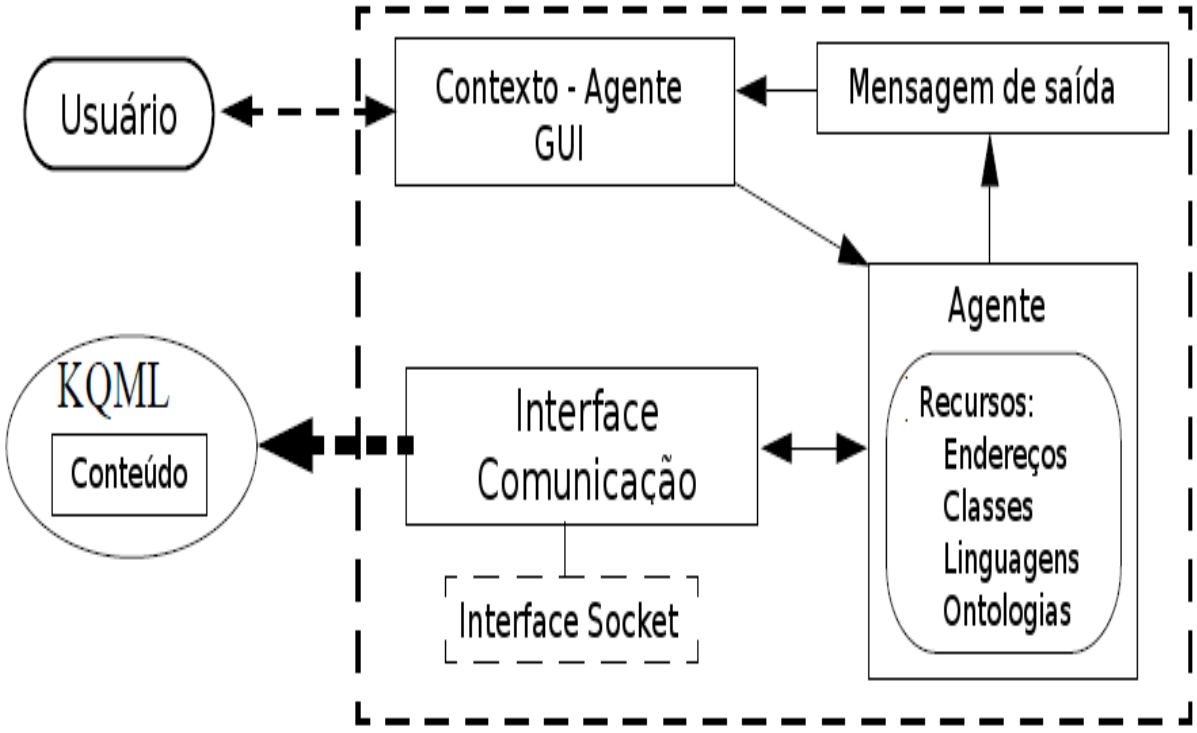
Figura 2.5 – Arquitetura integrada baseada em Agentes no domínio DFM [Frost e Cutkosky, 1996].

Nesta pesquisa existe a separação explícita entre *features* de projeto e *features* de manufatura (figura 2.6a, página 24). A arquitetura é do tipo usuário–sistema e pode ser integrada através da Internet (figura 2.4) conforme necessário.

Alvares e Ferreira, 2005 focaram a realidade tecnológica moderna no desenvolvimento de uma metodologia sistêmica integrada de usinagem através da Internet denominada *WebMachining*. O novo contexto de manufatura moderna atual engloba a possibilidade de manufatura eletrônica (*e–Mfg*) colaborativa. O domínio utilizado é o torneamento de peças cilíndricas através da Internet. O sistema, desenvolvido arquitetonicamente a partir de modelo IDEF0 [Menzel e Mayer, 1998], possui três componentes principais: modelagem – *WebCADbyFeatures*, plano de processo – *WebCAPP* e manufatura – *WebTurning*. Os módulos



(a) Taxonomia de objetos de manufatura.



(b) Agente–objeto de manufatura.

Figura 2.6 – Arquitetura DFM baseada em Agentes [Frost e Cutkosky, 1996].

são integrados através de uma arquitetura de agentes baseada, provavelmente, na primeira solução prática nesta área: JATLite. Os dados são armazenados em um banco de dados SQL (MySQL). Adicionalmente, a arquitetura de troca de informação é levada em consideração através do protocolo KQML utilizado pelos agentes [Chalupsky *et al.*, 1992, KQML Advisory Group, 1992]. O resultado final do sistema é um arquivo NC desenvolvido através do módulo STEP–NC, que será enviado para usinagem eventualmente.

2.3 Arquiteturas de Projeto para a Manufatura (DFM)

Projeto para Manufatura é um tópico relativamente extenso no gerenciamento do processo de projeto, especificamente o relacionado a projeto industrial. Existem diferentes áreas nas quais DFM pode ser utilizado tais como Engenharia Elétrica/Eletrônica, Engenharia de Materiais, bem como o foco desta pesquisa: Engenharia Industrial. A revisão bibliográfica deste aspecto, portanto, é limitada apenas aos tópicos específicos relacionados à pesquisa. Granta, 2006 expressa claramente que o aspecto mais relevante para o avanço efetivo de negócios, projeto e manufatura é o gerenciamento adequado da informação. O foco de sua análise resolve problemas de manufaturabilidade analisando dados críticos de engenharia. Boothroyd e Dewhurst, 2006 descreveram que os principais aspectos–foco no processo de projeto devem ser analisados para resolução de problemas relacionados com a manufatura. Ambas organizações possuem soluções comerciais para resolução deste tipo de problema.

Gupta e Nau, 1995 definiram os aspectos fundamentais de Projeto para Manufatura. Sua abordagem sistêmica é baseada em um modelador baseado em sólidos característicos (*feature based model* – FBM) integrado ao processo de projeto. A análise de manufatura é realizada após os estágios preliminar e CAD. Um projeto é retornado ao CAD caso não seja aprovado. Os planos de operações de manufatura consideram todas operações de manufatura de um determinado ambiente de fabricação. Após toda informação estar disponível, uma métrica quantificadora denominada índice de fabricação é calculada para todas alternativas de projeto. Um exemplo desta abordagem são os tempos estimados de operações, incluindo tempos de preparação (*setup*).

Bajaj *et al.*, 2003 consideraram a relevância de arquiteturas DFM em projeto, especificamente em desenvolvimento de produtos eletrônicos (EPR). A arquitetura DFM é baseada em quatro módulos de *software*: integrador do projeto, sistema especialista baseado em regras, gerador de visualizações de projeto e gerenciador dos resultados. Estes módulos descritos são integrados através de um modelo de informação baseado no padrão STEP utilizando a AP210 como sua fundação.

Sanchez, Priest e Souto, 1997 desenvolveram um assistente inteligente de projeto que incorpora aspectos de fabricação no processo de projeto. O primeiro aspecto mais importante da arquitetura é o reconhecimento adequado dos sólidos característicos

(*features*) de um modelo. Na segunda etapa o foco é a lógica necessária para analisar a manufaturabilidade do modelo. O principal aspecto inovador desta arquitetura é o Vetor de Abordagem (AV), que representa a direção na qual a ferramenta faz contato com a peça para iniciar a remoção de material. A arquitetura, entretanto, é limitada a características geométricas dentro do sólido mínimo possível (MEV).

Zhao e Shah, 2005 desenvolveram uma interface (ou casca, *shell*) DFM independente para processos de chapas e injeção moldada. Os aspectos abordados são tanto técnicos quanto econômicos em diferentes níveis de abstração. A interface possui métricas tanto qualitativas quanto quantitativas, tais como tempo e custo. A arquitetura, tipicamente, é utilizada por dois usuários: projetista e engenheiro de conhecimento.

Ji e Lau, 1999 utilizaram o método digráfico para análise de encadeamento de tolerâncias em DFM/CE. Embora a abordagem seja rápida computacionalmente e de resolução simples, ela não resolve todos requisitos de uma solução DFM completa. Sambu, 2001 desenvolveu uma abordagem DFM sistêmica para facilitar o gerenciamento do excesso de aspectos em análise na interface entre um projetista e um fabricante em Prototipagem Rápida (RP). A abordagem DFM é baseada em um conjunto de ferramentas, especificamente planejador de processo RP, revisor do tempo de vida de moldes e C–OptdesX.

Yim, 2005 propõe a utilização de uma arquitetura DFM para gerenciamento adequado dos seguintes aspectos de projeto:

- Modelo de informação que gerencie de forma adequada o processo de projeto.
- Abordagem sistemática que identifique similaridades e condições de projeto similares entre projetos distintos para obtenção de soluções pré–existentes apropriadas, bem como a decomposição de suas estratégias de solução.
- Gabaritos de soluções de projeto pré–existentes que possam ser combinadas e utilizadas para coordenação de intercâmbio da informação em um ambiente distribuído.

Embora todos os aspectos descritos sejam relevantes em DFM, a arquitetura ainda não foi completamente desenvolvida.

Howard e Lewis, 2003 desenvolveram um sistema de banco de dados para otimização de processos de manufatura durante um projeto. O banco de dados leva em consideração todos requisitos de informação embutidos nas atividades intrínsecas de projeto de um produto. No sistema, DFM é utilizada como uma ferramenta de análise (ou avaliação) de projeto. O modelo do sistema é fundamentado no padrão IDEF0 (*Integrated DEFinition*).

Changchien e Lin, 2000 utilizaram um sistema de decisão multi–variável para o projeto simultâneo (ou concorrente) de produtos usinados. Embora o sistema não seja uma arquitetura como, tradicionalmente, uma arquitetura é estruturada, utiliza os principais

aspectos de projeto: conceitual, DFM, DFA (*Design for Assembly* – Projeto para Montagem) e DFP (*Design for Production* – Projeto para Produção).

Ong, Sun e Nee, 2003 desenvolveram uma ferramenta DFM difusa baseada em AHP (*Analytical Hierarchy Process* – Processo Hierárquico Analítico) para peças rotacionais. Os principais aspectos de inovação da ferramenta são a utilização da intrínseca incerteza no processo de projeto e AHP. A combinação destes aspectos descritos permite o cálculo de índices de análise da melhor solução alternativa. Especificamente, a ferramenta utiliza índices de manufatura (MI) para ranquear as melhores soluções dentre as alternativas disponíveis.

Dai *et al*., 2002 propõem a utilização de um sistema multi–agente para uma área mais ampla: Projeto para X (DFx). A informação é centrada em ontologias [Gašević, Djurić e Devedžić, 1998]. O sistema, em estágio–protótipo, demonstra suas vantagens: reconfigurabilidade, facilidade de reutilização e escalabilidade.

3 INFORMAÇÕES NO DOMÍNIO DE PROJETO, PRODUTO E DFM

O domínio de uma arquitetura, capítulo 2, precisa ser compreendido para que as abordagens e técnicas de seu gerenciamento, sob o contexto de informação, possam ser contextualizadas corretamente. Este capítulo disponibiliza uma revisão histórica das principais formas de intercâmbio da informação, particularmente no domínio descrito anteriormente. As técnicas descritas aqui são utilizadas tanto *ipsis literis* quanto em parte nesta pesquisa, descrita no capítulo 4.

A próxima seção, seção 3.1, descreve as formas de armazenamento e consequentemente utilização da informação em arquiteturas computacionais. Especificamente, os tópicos são revisados organizacionalmente sob o contexto histórico. Portanto, a revisão da literatura pode nesta área ser associada também com a crescente disponibilidade de maior poder computacional, particularmente em computadores de médio e pequeno porte (ou pessoais). O foco é descrição das abordagens existentes para gerenciamento e análise da informação.

Especificamente, na seção 3.2 é disponibilizada uma revisão sobre a utilização e análise do contexto da informação (ou ontologias) com exemplos de sua estrutura e utilização neste domínio.

3.1 Integração e Intercâmbio de Informação

3.1.1 Contexto Histórico

O crescimento, popularização e acessibilidade de recursos computacionais permitiram o crescimento de soluções, juntamente sua disponibilização, particularmente em Engenharia. Associado a este fato a necessidade de intercâmbio entre as informações aumentou de forma exponencial (por exemplo, no caso de n aplicações que necessitassem trocar informações o número de traduções necessárias seria 2^n). A maioria dos sistemas computacionais necessita trocar informações com outros sistemas, especificamente no mesmo contexto da informação ou em contextos relacionados.

Desta forma, surge a necessidade do desenvolvimento de meios de padronizar a informação para este intercâmbio. As próximas seções descrevem, sob o contexto histórico com foco em engenharia, os diferentes meios para estruturar e padronizar a representação da informação e intercâmbio entre aplicações. Especificamente, os meios de troca da informação são fundamentados para o domínio de Engenharia Mecânica e/ou Engenharia Industrial.

Na área de projeto a necessidade de troca de informações é amplificada pela natureza da informação de projeto, que varia desde a descrição geométrica de uma peça até informações de manufatura, as quais tem grande variabilidade. Desta forma, não é suficiente descrever apenas seus tipos e natureza: materiais, processos relacionados, custo, lista de materiais bem como especificidades de uma determinada organização.

No contexto desta pesquisa, Young, 2003 especificou o quão importante e relevante seja que a área de DFM possua consistência e eficácia no gerenciamento de em informações. Esta especificação ressalta a importância da integração das informações entre os componentes de projeto e manufatura. Esta integração é diretamente relacionada com a necessidade de um processo DFM integrado de forma a garantir competitividade das organizações.

Inicialmente, entretanto, é relevante introduzir brevemente a modelagem de dados através do tradicional Método Integrado (IDEF) [Menzel e Mayer, 1998], que foi utilizado na definição da estrutura de vários modelos de informação, alguns deles descritos a seguir, inclusive. IDEF é uma linguagem e metodologia de modelagem utilizada para união, armazenamento e manutenção da representação da informação de um determinado domínio. Desta forma, existem dois aspectos intrínsecos: sintaxe e semântica definidas por um conjunto de regras, implícitas ou explícitas, que definem a validade da semântica e o significado de suas primitivas. O IDEF é subdividido em três partes descritas através de três modelos de representação:

IDEF0 método de modelagem de componentes (funções) de organizações. Um modelo IDEF0 é uma coleção hierarquicamente organizada de diagramas IDEF0; a hierarquia perfaz um arranjo do tipo árvore inversa com um nó "raiz" único e um número finito de nós "filhos".

IDEF1X método de modelagem de dados. IDEF1X visa a estruturação dos dados necessários para representar as funções descritas através de IDEF0. Temporalmente, IDEF1X foi focado na disponibilização de um método para representação de linguagens associadas com bancos de dado, tal como Entidade–Relacionamento (ER). Neste sentido, as primitivas existentes são entidades, atributos e relações (*entity, attributes, relationships*)

IDEF3 método de modelagem de processos (atividades) genéricos de organizações. Aparentemente, existe similaridade em termos do significado entre IDEF0 e IDEF3 entretanto o foco de cada método é diferente. IDEF0 é focado nas formas que as atividades de organizações são definidas e conectadas através de produtos e recursos enquanto IDEF3 é um método genérico de modelagem de processos. A primitiva básica de IDEF3 é o comportamento de uma unidade (UOB), caracterizado através de objetos que podem ter instantes temporais nos quais eles acontecem, inclusive eventualmente associado com relações temporais de outros processos. Portanto,

modelos IDEF3 são utilizados preferencialmente em processos genéricos de organizações nos quais a correta utilização do sequenciamento e temporalidade dos processos/atividades é crítico.

A modelagem da informação levando em consideração conceitos, ou contextualização, da informação é definida através de IDEF5 [Menzel e Mayer, 1992].

3.1.2 Intercâmbio Estático de Informações

A variabilidade da informação descrita na seção anterior necessita ser representada de forma adequada para troca entre diferentes sistemas de projeto. Neste contexto, é importante destacar que a abordagem inicialmente utilizada foi um formato neutro de arquivo. A informação representada neste formato neutro pode ser interpretada e expressa por componentes que funcionam como conversores, tipicamente intrínsecos na grande maioria dos sistemas de projeto. Esta ainda é uma forma popular para intercâmbio de informações entre sistemas diferentes. Desta forma, neste trabalho o intercâmbio de informações através de um formato neutro apenas com este foco é denominado estático; apenas quando características da forma de utilização da informação são diferentes do ciclo ler–traduzir–escrever outra nomenclatura é utilizada, conforme descrito nas próximas seções.

É relevante destacar que a especificação completa de um "formato adequado" ainda não está finalizada e provavelmente ainda levará um longo período de tempo até sua padronização completa. Isto ocorre devido à grande complexidade e variabilidade de sistemas CAD/CAM e processos de manufatura, diferentes requisitos do formato de intercâmbio de dados, informação e conhecimento entre organizações, associado com a complexidade crescente da tradução de informação proprietária para a representação padronizada comum. Adicionalmente, existe um progresso constante no ferramental de *software* disponível aumentando ainda mais este problema. No contexto de utilização pura de formatos padronizados, esta subseção descreve os formatos disponíveis mais utilizados tipicamente, descrevendo o quão adequado cada um é para proporcionar uma solução completa de intercâmbio de informações.

Zeid, 1991 reporta que qualquer formato de troca de informações de projeto deve adequadamente especificar quatro tipos de informação para ser efetivo: forma, descritiva, projeto e manufatura. Informações de forma se referem à descrição de sólidos característicos, bem como sua representação geométrica e topológica. A informação descritiva pode ser utilizada para representar imagens do produto e/ou parâmetros globais, tais como precisão e resolução dos dados, por exemplo. A informação de projeto é tradicionalmente associada com a representação dos dados gerados por procedimentos de análise durante o processo de projeto, por exemplo os dados de saída (*output*) de um módulo de análise por elementos finitos. Entretanto, informações de fabricação englobam, tipicamente, a representação de

materiais, planos de processos, ferramental, tolerâncias, etc. Gruber, 1994b adiciona a estes aspectos descritos um nível de representação do conhecimento. A representação desta especificação permite que processos, ou formas, de projetar sejam trocadas entre sistemas distintos.

A representação inicial para trocas de dados, denominado IGES 1.0, disponibilizado em 1980, permitia somente representação geométrica. Em 1981, a organização ANSI (*American National Standards Institute*) utilizou esta especificação para definição de dados de produto que, consequentemente, tornou–se o padrão Y14.26M. As versões 2.0 e 3.0, disponibilizadas em 1983 e 1986 respectivamente, possibilitaram a definição de mais sólidos característicos e novos aspectos descritivos, tais como informações de análise por elementos finitos. A versão 4.0, disponibilizada em 1988, permite que sólidos sejam representados através de Geometria Construtiva de Sólidos (*CSG*). A versão 5.1 flexibilizou-o permitindo representação por fronteira (*B–rep*). Alguns aspectos em relação à representação de desenhos podem ser representadas em campos adicionais neste formato.

Uma grande vantagem do padrão IGES é a compatibilidade reversa, ou seja, arquivos IGES 1.0 podem ser utilizados por tradutores da última versão IGES. O padrão IGES reflete um estágio no desenvolvimento de sistemas CAD/CAM quando o foco era fundamentalmente apenas a troca de informações geométricas, especificamente dados. Apesar desta limitação, o formato IGES ainda é largamente utilizado e disponibilizado por praticamente todo fabricante comercial de sistemas CAD/CAM.

O formato DXF, desenvolvido pela Autodesk, Inc., todavia é o formato–padrão de troca *de facto* para computadores pessoais. O formato foi desenvolvido inicialmente para o produto principal da empresa, AutoCAD, fundamentalmente para troca de dados geométricos, embora também seja utilizado em pacotes de *software* de outros tipos como renderização, etc. O formato DXF é baseado em uma arquitetura flexível de arquivos com campos obrigatórios e opcionais. Os campos opcionais podem ser utilizados por desenvolvedores de aplicativos terceirizados integrados com AutoCAD para representar informações adicionais como, por exemplo, propriedades de materiais, tolerâncias, etc. A flexibilidade deste formato foi um dos principais fatores para o sucesso comercial do produto.

A sintaxe de um arquivo DXF é bastante similar com a estrutura da linguagem LISP [Winston e Horn, 1989, Wilenski, 1986, Norvig, 1992], aspecto que simplifica o desenvolvimento de tradutores para seu formato. Cada entidade geométrica em DXF pode ter campos adicionais associados permitindo a customização de características específicas de um modelo de um produto. Entretanto, é relevante destacar que a representação direta de sólidos não é possível sem a utilização de campos adicionais. Este fato é devido ao modelo e estrutura da informação em DXF. Adicionalmente, infelizmente, as extensões do formato DXF não são regidas por um padrão, logo a troca e tradução de informações não

puramente geométricas é complexa.

Ambos os formatos, IGES e DXF, podem ser utilizados para intercâmbio de dados puramente geométricos com resultados razoáveis. Entretanto, para troca completa de informações, incluindo manufatura, representação do conhecimento e dados de projeto, não existiam soluções até meados da década de 1980 do século passado. Levando em consideração esta realidade em 1985 foi organizada a Definição de Intercâmbio de Dados de Produto (PDES) . Adicionalmente, em paralelo, ocorreu o projeto ESPRIT de intercâmbio de informações de produto na Comunidade Econômica Europeia (CEE).

A Organização Internacional de Padronização (ISO), em 1984, iniciou o desenvolvimento do padrão STEP [Wilson, 1992]. O objetivo principal do STEP é o desenvolvimento de uma arquitetura de informação para troca de modelos de produtos em diversas indústrias, tais como mecânica, eletrônica e arquitetura e construção (*Architecture and Engineering Construction* – AEC). A arquitetura STEP foi influenciada por diversos outros formatos como IGES, PDES, TAP, VDA/FS, ESPRIT e PDDI. Entretanto, é fundamental relevar que ela estende estes "formatos puros" através de um modelo em camadas possibilitando o desenvolvimento de modelos focalizados em cada área. Adicionalmente, STEP possui uma linguagem padrão independente do contexto para modelagem da informação denominada EXPRESS [ISO, 2004]. A arquitetura de informação STEP e sua flexibilidade comparativa com os padrões existentes, ou propostos, permitiu sua utilização como o padrão ISO 10303 [ISO, 2002a, ISO, 1995, ISO, 1995].

O foco da arquitetura STEP é a definição de uma representação uniforme de dados do produto bem como funções que permitam a troca destes dados entre sistemas computacionais distintos durante todo ciclo de vida do produto [Zhao e Liu, 2008a]. STEP é dividido em diferentes aspectos, ou partes, em quatro áreas principais: métodos descritivos (utilizando a linguagem EXPRESS), métodos de implementação, metodologia de teste de compatibilidade e modelos de informação com protocolos de aplicação.

O modelo da arquitetura STEP é composto de três camadas: física, lógica e aplicação. A figura 3.7 mostra a arquitetura STEP. A camada de aplicações possui a implementação dos modelos de informação para determinados domínios, por exemplo produtos mecânicos ou elétricos. O esquema de informação, ou Protocolo de Aplicação (AP), é utilizado para descrever a informação em termos de sua semântica com dados. Esta é a interface entre o usuário e o modelo STEP. Por exemplo, a AP–239 [ISO, 2005] é destinada ao gerenciamento do ciclo de vida do produto, a AP–203 [Pratt, 2001, STEP Tools, 2004] descreve o modelo STEP de dados, a AP–22 informa como acessar e trocar dados na arquitetura STEP, enquanto a AP–21 [ISO, 2002a] define a estrutura de representação da informação em um arquivo–texto.

A camada lógica contém os modelos de informação das entidades válidas em todos domínios da arquitetura STEP. Os modelos de informação são denominados Recursos

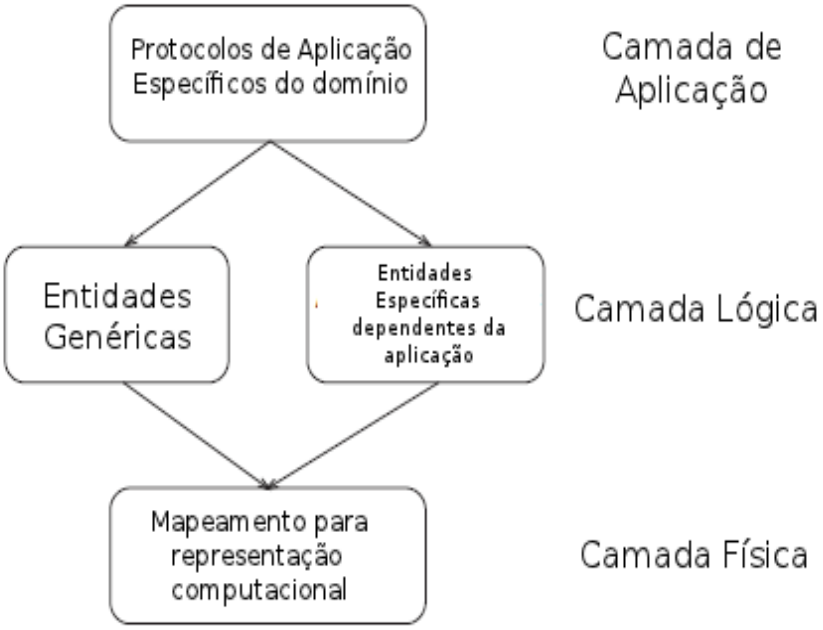


Figura 3.7 – Arquitetura STEP

de Informação (IR) [Loffredo, 1999]. As entidades representando informação podem ser subdivididas em genéricas ou específicas em uma aplicação. Entidades genéricas podem ser utilizadas por qualquer protocolo de aplicação que descreva a estrutura do significado semântico de um dado de um domínio. Entitades específicas de uma determinada aplicação são necessárias em domínios que requerem diversos tipos de entidades para representar dados.

A camada física possui os métodos implementacionais utilizados para mapear os modelos de informação para uma determinada representação computacional. Por exemplo, um método de implementação poderia mapear a informação de um determinado modelo de produto para um arquivo ou uma rotina computacional para armazenar dados em um banco de dados.

A implementação do modelo de intercâmbio de dados e informação STEP é baseada em uma arquitetura em diferentes níveis, analogamente à arquitetura em camadas descrita anteriormente. Este tipo de implementação permite o mapeamento do esquema de informação dos protocolos de aplicação para a representação computacional na camada física do modelo de informação STEP. Existem quatro níveis de implementação STEP, especificamente: troca de arquivos, troca de dados por tradutores (ou nível de execução/-trabalho), troca de dados por banco de dados e troca de bases de conhecimento. Estes níveis permitem acessar de formas distintas a informação através do modelo STEP de informação do produto. As figuras 3.8 e 3.9 mostram o modelo STEP de implementação.

O nível de troca de arquivos corresponde à situação atual na troca de dados de

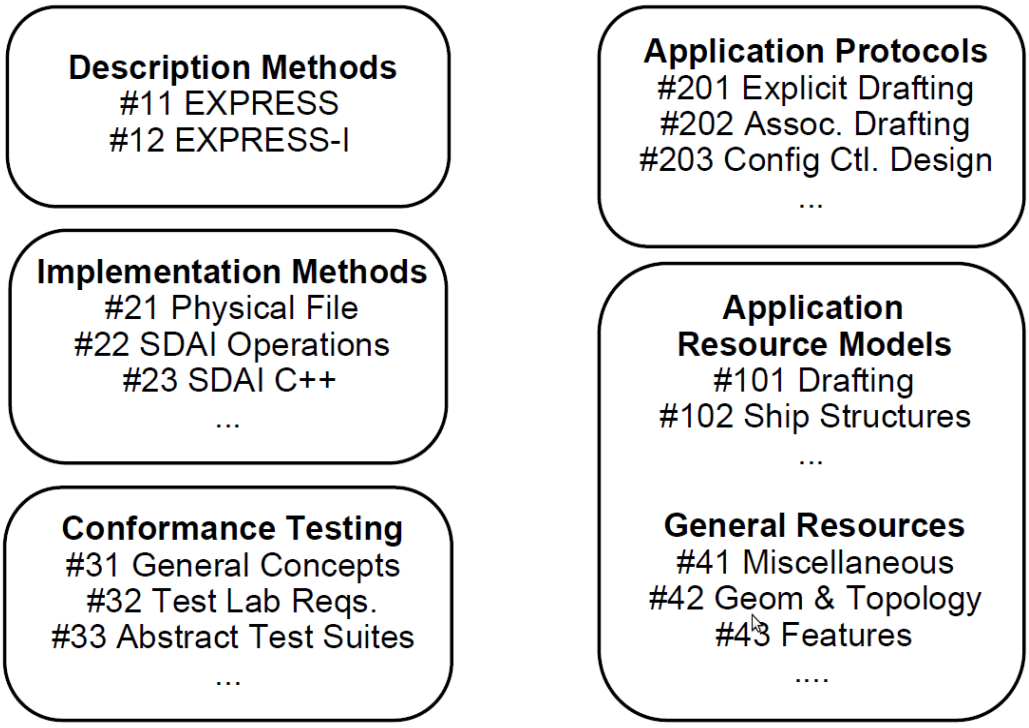


Figura 3.8 – Arquitetura de alto nível de implementação no modelo STEP [Loffredo, 1999].

produto, ou seja, em resumo é importar ou exportar a informação através de um módulo customizado para um dos protocolos de aplicação STEP. O nível de execução/trabalho permite a utilização de componentes externos de *software* para efetuar a tradução. O nível de troca por bancos de dados abstrai a troca de informação representada em STEP através de bancos de dados compartilhados. Neste contexto inclusive a troca de informação em tempo de execução (*runtime*) é possível. O nível de base de conhecimento, teoricamente, permite o intercâmbio de bases de conhecimento. Embora a arquitetura STEP de informação seja consistente e a mais completa disponível apenas os níveis de troca de arquivos e execução são bem desenvolvidos, inclusive com aplicações comerciais disponíveis. O nível de banco de dados também possui algumas aplicações comerciais, algumas delas em conjunto com iniciativas de pesquisa.

O último componente do modelo STEP de informação para integração de sistemas é a Interface Padrão de Acesso aos Dados (SDAI) [ISO, 2002b]. A SDAI define uma API para armazenamento e obtenção de informação em um meio de banco de dados (é relevante destacar que o banco de dados não necessariamente precisa ser um banco de dados relacional). Esta característica permite o desenvolvimento de aplicações de forma independente da tecnologia de armazenamento. Sob o ponto de vista prático, a SDAI pode ser utilizada para conexão de diferentes protocolos de aplicação com o nível de implementação de banco de dados na arquitetura do modelo STEP.

A linguagem EXPRESS é definida em esquemas (*schemas*) que representam o modelo do produto. O esquema de um produto é composto de entidades e tipos de dados que

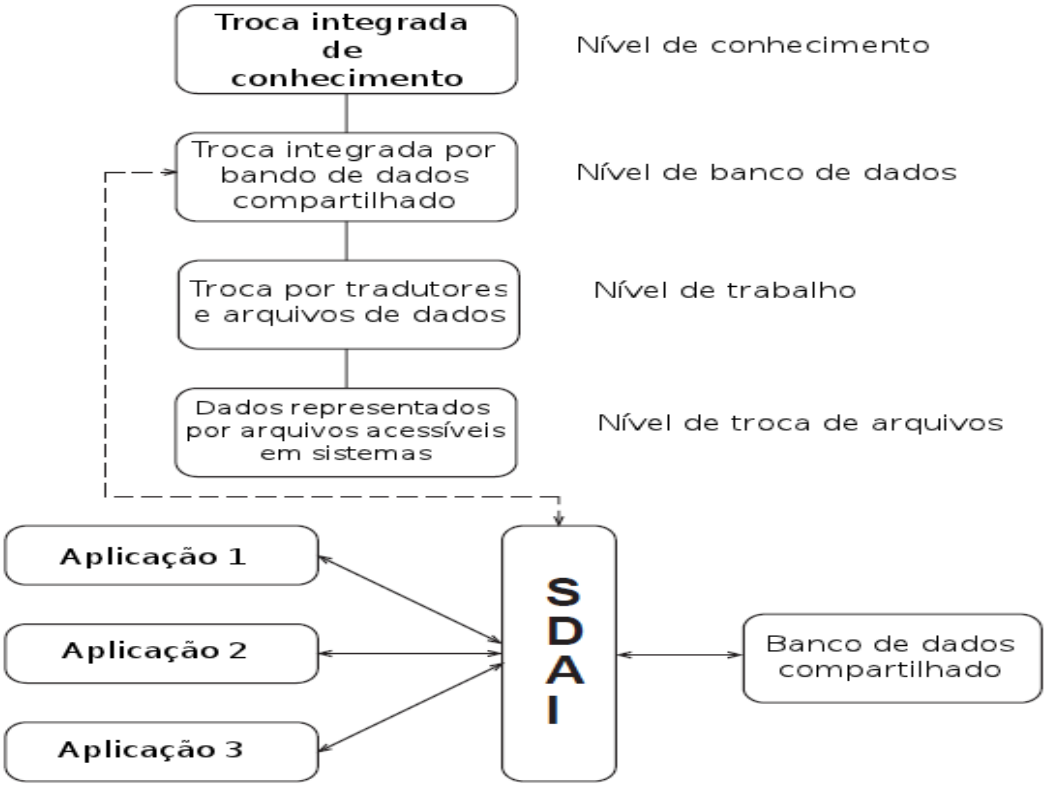


Figura 3.9 – Modelo de implementação STEP e interface SDAI.

especificam a definição destas entidades. As entidades possuem atributos e restrições (ou limites) sobre seus valores conforme necessário. Adicionalmente a estes componentes, *schemas* também podem ter operações, comandos, funções, processos e regras.

O *schema* é um conjunto de recipientes (*containers*) para um conjunto de tipos de dado e descrições de suas entidades. Os tipos de dados podem ser simples (número, inteiro, real, lógico, booleano, *string* e binário), agregados (*array, list, bag* e *set*), construídos (enumeração e seleção), tipo de dado de entidade e tipo de dado customizado. Os tipos de dados são passíveis de ser hierarquizados (sub ou super tipos), podem ter atributos (explícitos, derivados e inversos) e possuírem regras locais (com restrições do tipo UNIQUE e WHERE). Os operadores são as operações de manipulação dos tipos das entidades e tipos de dados. Comandos são elementos similares aos existentes em linguagens de programação para definição de algoritmos de funções e processos. Funções customizadas podem ser definidas pelo usuário. Finalmente, regras globais, especificadas através da primitiva RULE, podem ser definidas como restrições a uma instância de tipo de entidade ou a um conjunto de instâncias de vários tipos de entidades.

Adicionalmente, considerando que STEP obedece modelos de dados representados em EXPRESS, o acesso ao banco de dados é simplificado para uma determinada aplicação. STEP utilizando SDAI define um modelo de dados de produto praticamente completo, exceto pela troca de bases de conhecimento. Embora ainda incompleto, dado o suporte

e apoio da organização ISO, conjuntamente com a maioria dos desenvolvedores comerciais, provavelmente STEP tornar–se–á o padrão de troca de informações de produto eventualmente.

Fenves *et al.*, 2008 definiram um modelo abstrato de informação do produto – CPM2 com semântica genérica para produtos eletromecânicos. O modelo, focado no dado, é constituído por três componentes principais inseridos num artefato (entidade correspondente à parte física de um produto): função (descrição do objetivo do artefato), forma (é a solução proposta para o problema de um projeto) e funcionamento (descrição sobre como a forma implementa a função). A partir da descrição de um artefato três diferentes modelos de descrição (representação) podem ser obtidos.

O modelo conceitual é puramente abstrato e tem por objetivo "capturar" informação genérica do produto e definição de relacionamentos entre suas classes. O modelo intermediário é um modelo de informação subdividido em dois aspectos genéricos da informação utilizando *slots* específicos: domínio e tipo (descreve e representa um componente da **taxonomia** de uma classe) → projeto conceitual. Finalmente, o modelo de informação é obtido a partir da compilação de modelo intermediário que foi definido baseado em um modelo conceitual. Eventualmente, um modelo UML pode resultar da utilização de CPM2.

Conforme inicialmente previsto pelo modelo de implementação PDES, o intercâmbio somente de dados não é suficiente para troca completa de informações de um, ou vários, produtos entre sistemas. Assim, é necessário conhecer a forma sob a qual determinados dados bem como seu *significado semântico* são definidos, obtidos e utilizados. Por exemplo, em determinadas situações pode ser mais importante saber **como** um determinado dado foi obtido do que o resultado numérico final de um processo de cálculo. A forma e/ou estrutura de como determinada informação/dado é adquirida é relevante no processo de projeto de uma peça ou produto. Bases de conhecimento são fundamentais neste caso.

Neste contexto, bases de conhecimento disponibilizam e estruturam o conhecimento necessário para obtenção dos dados, seja de forma explícita (como em bases de regras) ou de forma implícita (codificado por redes neurais, por exemplo). A representação e troca deste tipo de informação é complexa devido à possibilidade de utilização de diferentes paradigmas para representar conhecimento [Gruber, 1994b]. Adicionalmente, o conhecimento sobre como resolver um determinado problema está intrinsecamente embutido nele, tornando sua representação, explicitação e extração muito difíceis de serem utilizadas de forma eficiente e efetiva.

O Formato de Intercâmbio de Conhecimento (KIF) [Genesereth e Filkes, 1992] tem por objetivo a definição de uma estrutura padronizada representação e suporte do conhecimento, bem como sua troca entre sistemas distintos. KIF é uma linguagem declarativa que permite que o usuário compreenda semântica. Ela é uma linguagem baseada em cálculo de predicados com parâmetros flexíveis de construção.

A capacidade de construção de sentenças arbitrárias baseadas em cálculo de predicados utilizando KIF é superior às limitações de outros paradigmas de representação, por exemplo SQL, baseado em sentenças atômicas, ou Prolog, que é baseado em cláusulas de Horn. Provavelmente esta é sua principal vantagem pois permite a representação de meta–conhecimento, ou conhecimento sobre o conhecimento, disponibilizando o desenvolvimento de paradigmas de representação mais complexos utilizando apenas a fundamentação da linguagem.

O foco do formato KIF não é o usuário final, ele disponibiliza estruturação para outros paradigmas de representação do conhecimento utilizando fundamentos semânticos básicos e sua sintaxe. Sob este contexto, KIF é equivalente à linguagem EXPRESS no modelo STEP. Enquanto STEP tem como foco modelos de informação descrevendo dados, KIF foca o desenvolvimento do formato do intercâmbio de paradigmas de representação de conhecimento.

A linguagem mais aceita como padrão de troca de conhecimento é denominada Linguagem de Manipulação e Aquisição do Conhecimento (KQML) [Chalupsky *et al.*, 1992] . Ela é uma linguagem de segunda geração para representação do conhecimento implementada utilizando fundamentos do KIF (embora isto não tenha sido seu objetivo inicial). KQML tem por objetivo o intercâmbio de conhecimento entre agentes inteligentes de *software*. KQML foi utilizada de forma restrita, ou limitada, na arquitetura PACT [Cutkoski *et al*., 1993] como mecanismo de troca de informação entre sistemas de projeto.

KQML possui tanto sintaxe própria quanto protocolos para troca de informações entre agentes inteligentes [KQML Advisory Group, 1992, DARPA, 1993]. Ela tem uma arquitetura em camadas similar à arquitetura STEP ou ao protocolo de informações da Internet, por exemplo TCP/IP. A arquitetura possui três camadas principais: conteúdo, mensagens e comunicação. A camada de conteúdo descreve o conhecimento em uma linguagem representacional adequada, por exemplo KIF. Cada agente de *software* que necessite conhecimento também precisa ser capaz de interpretar a linguagem desta camada. A figura 3.10 mostra a arquitetura em camadas KQML.

A camada de mensagens contém a codificação sobre o tipo de mensagem que será trocada. Duas formas de codificação de mensagens existem: *conteúdo* e *declaração*. Mensagens de conteúdo contém o conhecimento sendo trocado entre componentes de sistemas. Estas mensagens podem descrever, inclusive, o tipo de conhecimento sendo disponibilizado ou procurado. Desta forma, mensagens de conteúdo descrevem o interesse de um determinado componente seja ele por uma informação necessária (*input*), ou conhecimento necessário, para realizar uma atividade ou o conhecimento disponibilizado para um sistema.

Mensagens de declaração descrevem especificamente meta–informação sobre mensagens de conteúdo, ou seja, informação sistêmica sobre cada componente de um sistema

Figura 3.10 – Arquitetura da linguagem KQML.

em relação à sua capacidade e requisitos. Adicionalmente, mensagens de declaração também podem ser utilizadas para serviços sobre o registro de componentes tal como um requisito especificado por determinado tipo de conhecimento. Mensagens de conteúdo também podem ser embutidas em mensagens de declaração conforme o código–exemplo 3.1. Este intercâmbio de mensagens pode ser feito através de uma conexão par-a-par ou através de um repositório central de conhecimento. O código–exemplo 3.2 mostra uma mensagem de comunicação KQML típica.

Lista 3.1 – Mensagem de declaração com conteúdo embutido

```
(DCL
    :TYPE query
    :DIRECTION import
    :MSG
    (MSG
    :TYPE query
    :QUALIFIERS (:number-answers 3)
    :CONTENT-LANGUAGE KIF
    :CONTENT-ONTOLOGY (operações-torneamento)
    :CONTENT-TOPIC (parâmetros-torneamento)
    :CONTENT (parâmetro velocidade ?velocidade)))
```

A camada de comunicação é responsável pela estruturação de "envoltórios" nas mensagens trocadas entre componentes de sistemas. Esta camada, diferentemente de TCP/IP, não possui capacidade de comunicação em rede e verificação de erros. O último aspecto é particularmente complexo para sistemas distribuídos pois não é possível saber se uma determinada mensagem foi ou não recebida ou se seu conteúdo é exatamente

igual ao que foi enviado. Esta limitação foi eliminada através do desenvolvimento do Protocolo Simplificado de Transferência de Conhecimento (SKTP). Em resumo, SKTP é a implementação do padrão KQML sobre o protocolo TCP/IP. A implementação é uma customização das camadas de mensagens e comunicação. A camada de mensagens permite a definição de bibliotecas de interfaces, denominadas facilitadores (FIL), para intercâmbio customizado de dados entre componentes de sistemas.

Lista 3.2 – Mensagem KQML com mensagem de declaração embutida e conteúdo embutido

```
(PACKAGE
    :FROM turning-expert
    :TO framework-registry
    :ID reg.1908765
    :COMM sync
    :CONTENT
    (DCL
        :TYPE query
        :DIRECTION import
        :MSG
        (MSG :TYPE query
        :QUALIFIERS (:number-answers 3)
        :CONTENT-LANGUAGE KIF
        :CONTENT-ONTOLOGY (operações-torneamento)
        :CONTENT-TOPIC (parâmetros-torneamento)
        :CONTENT (parâmetro velocidade ?velocidade))))
```

3.1.3 Intercâmbio de Informações em Tempo de Execução

Tradicionalmente, o acesso à informação em tempo de execução está associado com a disponibilidade de uma interface de programação, pública ou proprietária. Este fato tornou a integração simplificada um aspecto complexo em sistemas distribuídos. Desta forma, estes sistemas ficaram limitados a utilizar formas mais tradicionais de intercâmbio de informações como arquivos ou características específicas dependentes de sistemas operacionais, como execução de rotinas remotas (RPC) e/ou áreas de memória compartilhada.

Na área de projeto, especificamente, o acesso a este tipo de informação é ainda mais difícil devido à complexidade e organização das estruturas internas dos dados. A disponibilização e acessibilidade de novas formas de definição e estruturação na tecnologia da informação, capacidade de processamento computacional e padronização permitiu acesso mais simplificado a informações distribuídas em tempo de execução.

Historicamente, a principal inovação nesta área foi o desenvolvimento e padronização da Arquitetura Comum de Gerenciamento e Intercâmbio de Objetos (CORBA) pelo organização OMG [Mowbray e Zahavi, 1995]. CORBA versão 2.0 é a especificação

de serviços no nível de aplicações através de uma arquitetura de objetos distribuídos. Adicionalmente, CORBA também é neutro em relação à sua linguagem de implementação. A implementação CORBA é utilizada e adotada de forma abrangente atualmente e cada vez mais torna–se o padrão *defacto* para troca de informações e processamento em ambientes distribuídos.

Finalmente, a interface SDAI da arquitetura STEP também permite intercâmbio de informações em tempo de execução, embora de forma mais limitada que CORBA. Teoricamente, uma implementação conjugada da SDAI com CORBA permitiria um poderoso paradigma de intercâmbio de dados e informações de produto integrando a capacidade de processamento distribuído CORBA com as especificações do domínio de projeto STEP. Relativamente mais recente nesta área, a disponibilidade da linguagem Java com facilidades para o intercâmbio de informações em tempo de execução, tanto como um sistema servidor como cliente, promete ser uma nova alternativa para especificação e implementação de sistemas de suporte em projeto no contexto desta pesquisa.

Em geral, todas estas abordagens algorítmicas recém–expressas não podem ser interpretadas e/ou utilizadas sob contexto binário, pois o nível de conhecimento da informação não é representado desta forma. Sudarsan *et al.*, 2005 ressaltam a relevância de uma modelagem adequada e mais completa para suporte mais eficiente e eficaz em PLM. Desta forma, a utilização da informação, incluindo contexto, através de ontologias é descrita a seguir.

3.2 Utilização de Informações Contextuais

Considerando que o foco desta pesquisa é propor uma forma adequada para utilizar informações, que possuem heterogeneidade, neste domínio, a revisão de cada aspecto relacionado é, quando possível, descrita e comparada contextualmente com ela. Nesta área, a nova disponibilidade teórico-computacional permitiu que a abstração, definição, representação e utilização da informação tenham maior "semelhança" com a forma humana de compreensão, como nas diversas soluções descritas a seguir.

Consequentemente, seu relacionamento com o foco da pesquisa não necessariamente é de compreensão simples. Portanto, ao final de cada aspecto relacionado descrito é feita uma análise comparativa sucinta de cada solução a fim de informar se a mesma é passível de utilização.

Guerra-Zubiaga e Young, 2008 exploraram as formas das inter–relações entre conhecimento e manufatura em um modelo deste tipo. Adicionalmente, os autores propõem uma abordagem específica para aquisição das interações entre conhecimento e informação neste domínio através de orientação a objetos.

O fundamento desta abordagem é baseado na ampla gama de informações existentes e necessárias para tomada de decisão; desta forma é obtida vantagem mercadológica para uma empresa. A pesquisa é baseada puramente em conhecimento, sua estrutura e modelada através de UML. Desta forma, o conhecimento em manufatura é classificado em três tipos, identificados como:

Explícito é o conhecimento formal e sistemático representado por fatos básicos e conjuntos armazenáveis de documentos (texto, tabelas, diagramas, especificações de produtos, etc.).

Implícito é o tipo de conhecimento existente que pode ser expresso, ou articulado, explicitamente porém não é. Sua definição é expressa a partir de um comportamento observável ou atividade (ação) assumida.

Tácito tipicamente o conhecimento tácito é considerado não passível de articulação, entretanto ele pode ser comunicado e/ou trocado bem como adquirido através de descrições humanas.

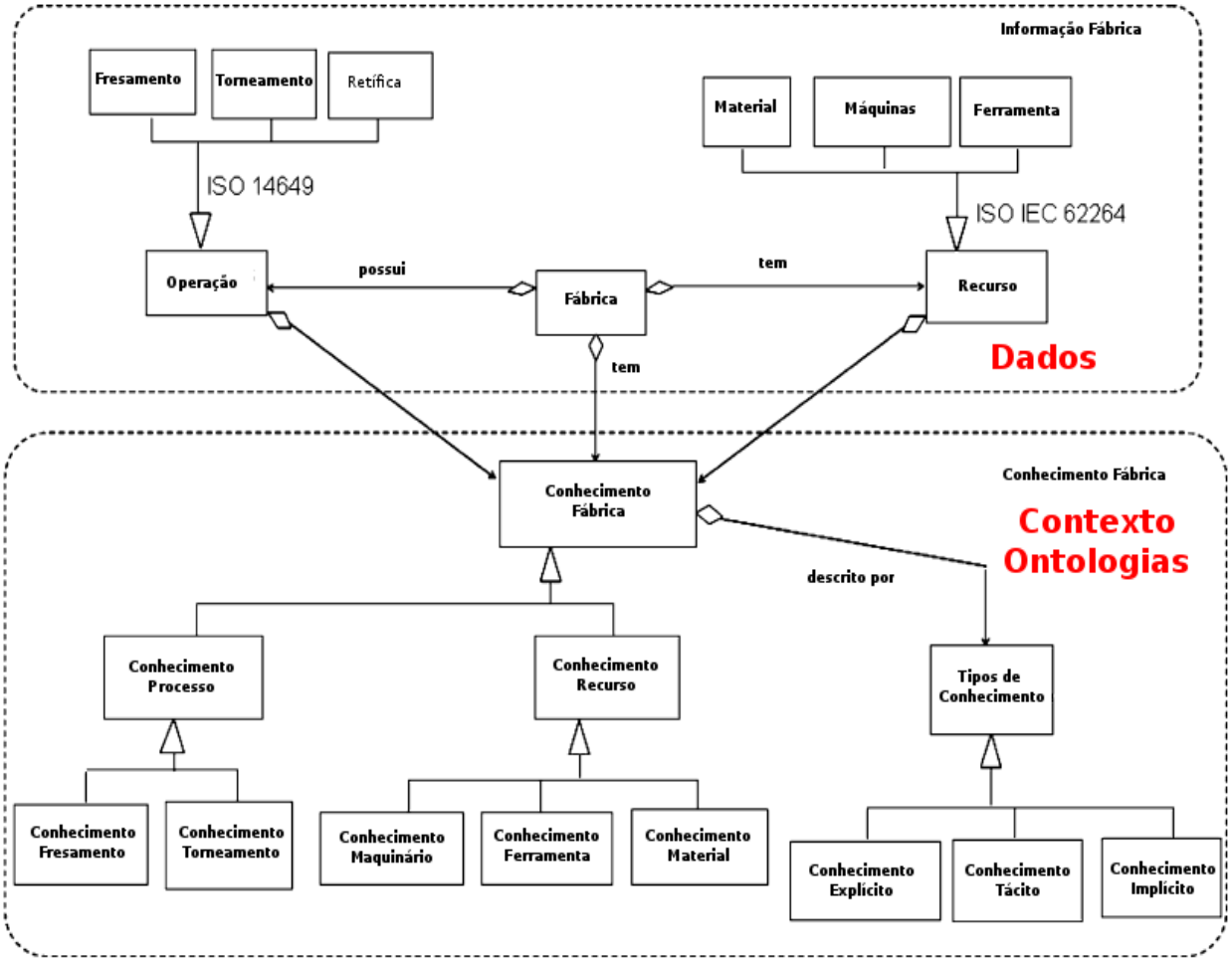


Figura 3.11 – Relações entre conhecimento e informação em um sistema de manufatura baseado em conhecimento segundo Guerra-Zubiaga e Young [Guerra-Zubiaga e Young, 2008].

Os autores desenvolveram um modelo de manufatura baseado em conhecimento, *Manufacturing Knowledge Model* (MKM), figura 3.11, que utiliza três tipos de informação: recursos de manufatura, processos e estratégias subdivididos em quatro níveis correspondentes a estruturas arquitetônicas sistêmicas: fábrica, oficina (*shop*), célula e estação. A estrutura de conhecimento/informação, denominados recurso (*facility*), é baseada em duas classes principais: recurso de informação e recurso de conhecimento, onde estão definidos os tipos de conhecimento, tanto o tipo de conhecimento *per se*, conforme descrito acima, quanto o conhecimento de fabricação, subdividido em processo (usinagem, torneamento) e recurso (maquinário, ferramental e material).

As informações do sistema são modeladas em UML com suas características, como ligações nomeadas e seu conteúdo conforme a figura 3.11, e interligadas *diretamente* com o modelo representado de forma direta/inequívoca. Esta abordagem é utilizada em processos e recursos. A descrição do conhecimento de cada aspecto e tópico é classificada e distribuída em categorias na figura 3.11. Desta forma, o sistema funciona como repositório de informações e conhecimento para planejamento do processo em sistemas de manufatura.

O arcabouço definido possui eficácia e eficiência dentro do objetivo proposto; é relevante salientar que o recurso de informação utiliza padrões ISO para características de manufatura (ISO 14649[1]) e fabricação (ISO IEC 62264[2]), fato que proporciona uma solução padronizada. Esta característica é díspar nesta pesquisa devido a seu foco ser distinto, além de utilizar um arcabouço ontológico. A solução é implementada em um sistema especialista (KBS) utilizando regras SE-ENTÂO (IF-THEN). O mapeamento feito pelo KBS entre o espaço de soluções e as informações necessárias é direto, ou seja baseado no dado, e neste aspecto reside a principal limitação de sua flexibilidade, como será descrito no capítulo 4. Embora esta pesquisa, com certeza, tenha eficiência e eficácia dentro do escopo proposto, todavia ela não contempla a forma mais completa de utilização de informações e conhecimento através ontologias, descritas a seguir e que são parte intrínseca desta pesquisa.

Gašević, Djurić e Devedžić, 1998 identificaram a importância bem como relevância de uma representação mais completa da informação incluindo ontologias. As formas e técnicas de representação do conhecimento tais como objeto–atributo–valor, incerteza, lógica difusa, regras, redes semânticas e *frames* são descritas e detalhadas, ainda que brevemente. Adicionalmente, as linguagens tradicionais de representação do conhecimento são expressas. O foco da pesquisa dos autores é ressaltar a relevância da utilização do conhecimento com sua informação associada em arquiteturas baseadas no modelo (*Model Driven Architecture* – MDA).

[1]ISO/TS 14649-201:2011 Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers.

[2]IEC 62264-1:2013 Enterprise-control system integration.

Sistemas MDA são partes intrínsecas da Internet do futuro, tradicionalmente denominada Web Semântica (*Semantic Web*) com seu protocolo e forma de representação. Finalmente, o modelo e sua representação através de ontologias são descritos utilizando UML com exemplos. A utilização de UML simplifica a definição, estrutura e implementação de ontologias efetivamente integradas com MDA.

Gruber, 1993 identificou, de forma pioneira, a necessidade de estruturar de forma adequada o conhecimento e sua representação da informação utilizando ontologias. Adicionalmente, considerando-se esta nova forma adequada de representação dos tópicos descritos, é relevante em um ambiente de projeto integrado e colaborativo de produto [Gruber, Teinembaum e Weber, 1992]. Neste contexto, ele identificou e estruturou o fundamento da arquitetura e princípios para o desenvolvimento de ontologias [Gruber, 1994b].

Guarino, 1997 sugere, inicialmente, um processo para definição, construção e utilização de ontologias a partir dos aspectos relevantes na representação contextual de conhecimento expressos [Heijst e Schreiber A. Wielinga, 1997]. A hipótese básica, inicialmente, está relacionada com a dependência entre uma determinada aplicação/tarefa/atividade e, majoritariamente, que *tipos* de conhecimento são necessários, ou devem ser estruturados/codificados [Bylander e Chandrasekaran, 1988]. Sob o contexto de conhecimento, esta hipótese é relacionada com a *representação* do conhecimento, portanto no *nível de símbolo*.

O autor defende, especificamente, a independência do conhecimento de um domínio, ou seja, ontologias são definições sobre determinados "pontos de vista". Em resumo, ontologias são acordos sobre conceitos compartilhados [Gruber, 1994a]. Conceitos compartilhados geralmente incluem arquiteturas de modelagem do conhecimento de um domínio, protocolos de comunicação entre componentes/agentes e acordos sobre a representação de determinado domínio. Sob o contexto de compartilhamento de conhecimento, ontologias são representadas como a definição de um vocabulário representacional e sua taxonomia. Logo, conforme descrito, ontologias e seus conceitos são diferentes. Uma consequência deste fato é que uma ontologia não é uma *especificação* de um conceito mas um *acordo (possivelmente incompleto) sobre um conceito*.

Adicionalmente, ontologias de uma determinada aplicação são, na verdade, casos específicos de um repositório de ontologias de aplicações e ontologias de um determinado domínio[3]. Consequentemente, ontologias pertencem ao nível de conhecimento e *podem* depender de diferentes "pontos de vista". A usabilidade, ou *valor*, de uma ontologia portanto depende do nível desta dependência[4].

[3]Biblioteca de ontologia: conhecimento reutilizável entre diferentes aplicações. Ontologia de aplicação: conhecimento específico de determinada aplicação.

[4]Uma descrição completa sobre ontologias e web semântica está fora do contexto desta pesquisa entretanto o livro Web Semântica para Leigos [Pollock, 2010] fornece um bom ponto inicial neste tópico.

Poli, 2003 definiu algoritmicamente a estrutura, hierarquia e limites da representação através de ontologias, inclusive sua implementação, e sugeriu um processo sequencial estruturado para utilização de ontologias. Inicialmente são utilizadas ontologias tanto como base, ou *philosophia prima*, quanto objeto final de definição da estrutura filosófica de um determinado domínio, *philosophia ultima*. Em geral, a informação pode ser classificada em três categorias (ou dimensões): propedêutica, semântica e ontológica. Especificamente, em relação a ontologias deve ser analisado um aspecto fundamental: a dependência ou independência do domínio. Em cada um destes dois aspectos ontologias podem ser tipificadas como descritivas, categorizadas e formais. Estes diferentes tipos são organizados de forma hierárquica a partir da mais abstrata que é a descritiva. Finalmente, é sugerido o desenvolvimento de um padrão de estruturas ontológicas levando em consideração a elevada quantidade de combinações com variabilidade entre os componentes. O gabarito poderia incluir primitivas de ontologias tais como PROCESS, OBJECT, GROUP, PART e TODO[5], com sua estrutura léxica associada, para auxiliar a construção de outras ontologias adicionais, obviamente desde que o conhecimento e informações possam ser estruturados desta forma em um determinado domínio. Particularmente em relação a esta pesquisa, a utilização da classificação em categorias sugerida auxiliou como separar a descrição puramente semântica de sua instanciação/implementação ontológica em, por exemplo, uma ferramenta computacional como Protégé [Horridge e Brandt, 2011].

Wand e Weber, 1990 explicitaram que a falta de formalização adequada da informação limita o desenvolvimento de sistemas computacionais e/ou de informação. A hipótese tradicional no desenvolvimento de sistemas e subsistemas deste tipo assume que eles sejam descritos e definidos em termos de especificações absolutas, ou seja, mapeamento direto entre estados de *input* e consequentes estados de *output*. Estas entidades básicas são mapeadas através de programas que contextualmente são designados como estados (*states*). Os estados de um sistema, portanto, são o mapeamento entre um programa (sistema) e valores. Desta forma, o foco primordial de um sistema é *resolver o objetivo* de um problema a despeito de conhecer *como resolver* o problema em análise.

A descrição ontológica é fundamentada em dois conceitos: parte interna (*como resolver*) e parte externa (*resolver o objetivo*) integrados através de um operador de decomposição. A compreensão completa de um sistema ocorre quando um mapeamento claro entre sua visão externa (*o que deve ser obtido, ou qual deve ser o resultado final*) e sua visão interna (*como obter o resultado final*) existe. O mapeamento adequado proposto nesta pesquisa é através de ontologias que permitem um formalismo adequado em ambas as visões. O mapeamento na ontologia é realizado através de grafos agregados representando um sistema. A utilização da estrutura da ontologia resultante de um sistema que melhor descreve o comportamento de sistemas mais complexos é obtida pelo princípio da decomposição, descrito acima.

[5] As primitivas podem representar semanticamente conceitos estáticos, p.ex. PART, ou dinâmicos, p.ex. TODO.

A utilização de ontologias permitiu o desenvolvimento de um modelo–protótipo de identificação do ponto de controle e auditoria quando a informação sistêmica é alterada. Adicionalmente é possível identificar os atributos mais relevantes dos requisitos ótimos de um sistema. Esta abordagem de solução permite identificar o quão adequadas são a decomposição funcional, fluxo de dados e estrutura de dados em função do fundamento básico de decomposição sistêmica.

A solução proposta utiliza esta abordagem descrita através da representação de ontologias de sistemas e seus subsistema com a utilização efetiva e eficiente das primitivas (por exemplo, agregação). Sistemas e seus componentes são expressos em hierarquias conceituais e descritos através de ontologias. A estrutura geral de uma ontologia é expressa, geralmente, em termos de substância e descrições determinísticas. A substância é definida pela união de três aspectos: teoria dos particulares – o quão específicos são os itens de uma ontologia, níveis de realidade – interpretação específica de um item/abstração e finalmente o todo e suas partes – categorização de conceitos, por exemplo agregado, todo, parte e sistema. As descrições determinísticas expressam aspectos relacionados a inter-relações entre conceitos como, por exemplo, oposição passível de definição ou definida e oposição intensa ou extensível.

Noy e McGuinness, 2004 propõem etapas sequenciais para desenvolvimento de ontologias. As razões principais para estas etapas estão relacionadas com os objetivos gerais de ontologias, dentre eles: compartilhamento da compreensão global da estrutura da informação entre os componentes de um sistema (pessoas ou *software*), possibilitar a reutilização do conhecimento de um domínio, tornar explícitas as hipóteses de um domínio, separação entre conhecimento de um domínio e conhecimento operacional e analisar o conhecimento de um domínio. As etapas[6], que são sequenciais, são baseadas nos elementos–base de ontologias: classes, *slots* e *facets*. O conjunto destes elementos forma a base do conhecimento de um domínio.

Após breve introdução inicial sobre a relevância do ato de projetar, Tian, Zou e Guo, 2006 ressaltaram o quão relevante projeto conceitual é no domínio de engenharia mecânica, particularmente mecanismos. Devido à complexidade desta atividade, um sistema coerente e completo de projeto de mecanismos auxiliado por computador ainda permanece sem solução. A solução proposta pelos autores envolve a utilização de um modelo de representação do conhecimento. Especificamente, o modelo é baseado em uma função de tópicos de topologia, estrutura e funcionamento de mecanismos genéricos (TSBF). Estes aspectos são analisados tanto quantitavamente quanto qualitativamente. Nesta pesquisa, a mesma fundamentação utilizada por Tian *et al*., qual seja a falta de integração de ambos tópicos quantiqualitativos, porém, nesta caso, para um domínio e escopo díspar é utilizada. Adicionalmente, existe uma hierarquia de classes destes tópicos

[6] As etapas são descritas no capítulo 4 para o domínio desta pesquisa.

representando um mecanismo abstrato. A hierarquia, entretanto, possui uma estrutura que não possui a abstração mais adequada, conforme descrito a seguir.

A limitação principal se refere à falta de abstração e representação adequada das definições formais do domínio, por exemplo a definição do que seja movimento de um elemento de uma peça, pois ele é *explicitamente representado*. Este tipo de representação é análoga na parte qualitativa. A solução deste aspecto utiliza vocabulários do domínio, embora com estrutura distinta da estrutura e representação abstrata ótima através de ontologias. A utilização dinâmica do sistema é através da integração do conhecimento e informação sob contexto algorítmico. A solução computacional integrada baseada em conhecimento é obtida através de técnicas típicas de sistemas especialistas, tais como *backward–chaining* e *forward–chaining*. Um estudo de caso de mecanismos de uma máquina de costura zigue–zague é descrito e resolvido adequadamente utilizando este método.

Sim e Duffy, 2003 definiram os aspectos envolvidos no desenvolvimento de ontologias genéricas de projeto em engenharia. Inicialmente, existe a proposição que não existe em engenharia de forma exatamente igual o conceito ontológico de conhecimento compartilhado (*shared knowledge*) em relação às distintas etapas e atividades de um projeto. Baseado nisto, se uma eventual definição deste tipo existisse ela não descreveria de forma completa o trabalho de um projetista. Neste contexto, os autores propõem que a necessidade/solução deste tópico é uma ontologia–base que sirva como fundação para que pesquisadores de modelos e teorias de projeto possam a utilizar no conhecimento compartilhado da abrangência de cada atividade de projeto. A implementação da solução utiliza de uma arquitetura de agentes de *software*. A utilização e implementação desta arquitetura é devido ao fato que a parte cognitiva de projetar é relacionada com o nível de conhecimento. Agentes de *software* possibilitam a melhor analogia com o nível de conhecimento [Newell, 1981, Dasgupta, 1994]. A ontologia específica da pesquisa é uma taxonomia de uma atividade genérica de projeto. Estes componentes funcionam em *loop* com A_d como foco, figura 3.12. Uma breve introdução à Lógica Modal e Taxonomia com exemplo de um componente de um dos diversos componentes de projeto está descrita no Apêndice A.

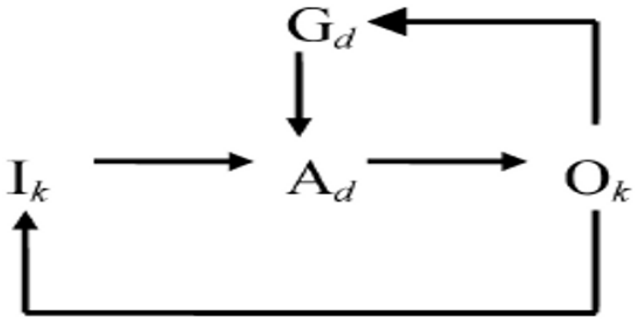


Figura 3.12 – Formalismo da atividade de projeto.

- Conhecimento de entrada (*input*) do projeto: I_k.
- Atividade de projeto: A_d.
- Meta, ou objetivo, do projeto: G_d.
- Conhecimento de saída (*output*) do projeto: O_k.

As atividades descritas são muito genéricas e abstratas, logo os autores propõem uma organização contextual das atividades de projeto conforme descritas a seguir.

Definição atividades que gerenciam a complexidade das etapas de um projeto enquanto respeitando seus limites; o objetivo é especificar todos detalhes necessários para produção.

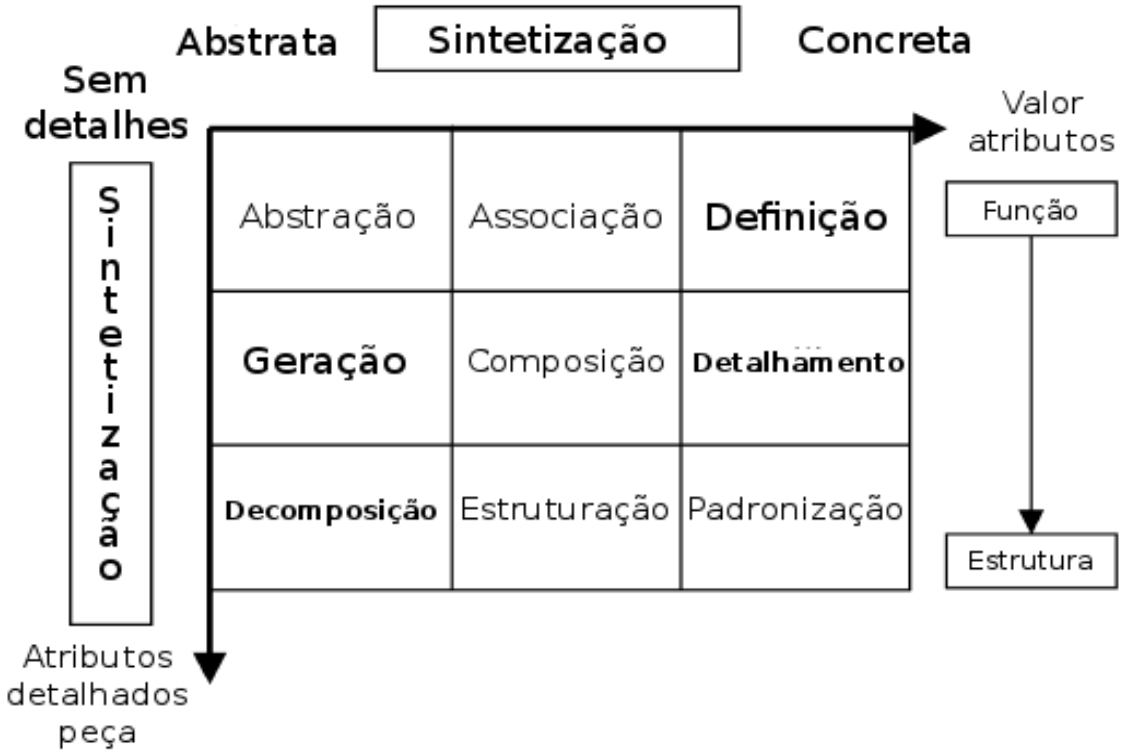
Avaliação são atividades que analisam a viabilidade de soluções potenciais de projeto descartando soluções inviáveis; através disto o espaço de busca de soluções é reduzido.

Gerenciamento atividades responsáveis pela coordenação iterativa das tarefas de um projeto (dinamicamente) bem como seu processo.

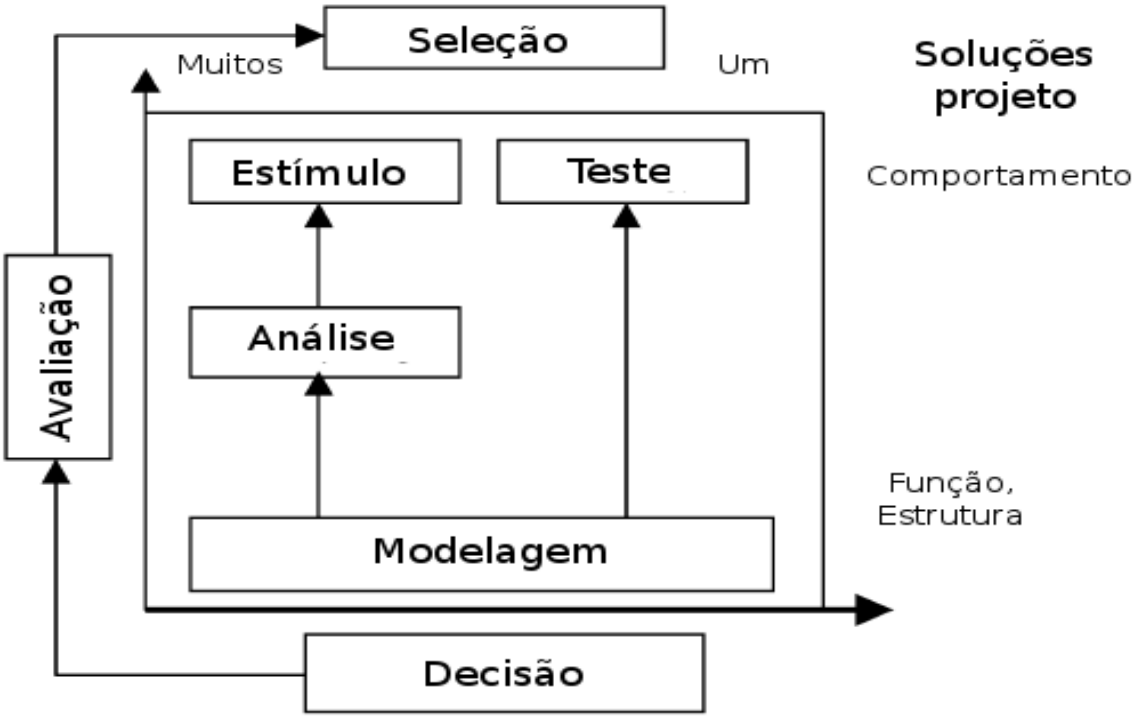
Desta forma, tanto a abstração é diminuída quanto o detalhamento e especificidade das atividades podem ser mais eficientemente realizados. A redução da generalidade e abstração é efetuada através de atividades de sintetização em projeto, conforme as figuras 3.13a e 3.13b demonstram. Esta abordagem integrada é utilizada como base para a atividade de gerenciamento do processo de projeto, descrita contextualmente na figura 3.14. A pesquisa deste trabalho não utiliza-a devido a seu nível de abstração ser excessivo, demandando o desenvolvimento de conceitos, consequentemente suas classes e propriedades, díspares de seu foco e objetivo.

Bock *et al.*, 2010 desenvolveram uma linguagem de modelagem de produtos em projeto colaborativo, ou conjunto, baseada em ontologias. Em geral, modelos de produtos desenvolvidos colaborativamente podem ser sem intenção, ou intencionalmente, incompletos. Adicionalmente, vários modelos de um produto podem existir em um determinado instante. Modelos de produto desenvolvidos em grupo tradicionalmente são especificados em termos de sua estrutura, função (ou comportamento) ou ambos. A combinação de ambos os aspectos define um *sistema completo*, ou total. A arquitetura STEP ainda não suporta completamente o desenvolvimento colaborativo de produtos, especificamente:

- Modelo do produto e seu ambiente: projetos e seus requisitos, sistemas totais.
- Generalização: taxonomias e aperfeiçoamento.



(a) Taxonomia de atividades de definição em projeto e suas inter–relações.



(b) Taxonomia de atividades de avaliação/análise em projeto e suas inter–relações.

Figura 3.13 – Desenvolvimento de ontologias genéricas de projeto em engenharia [Sim e Duffy, 2003].

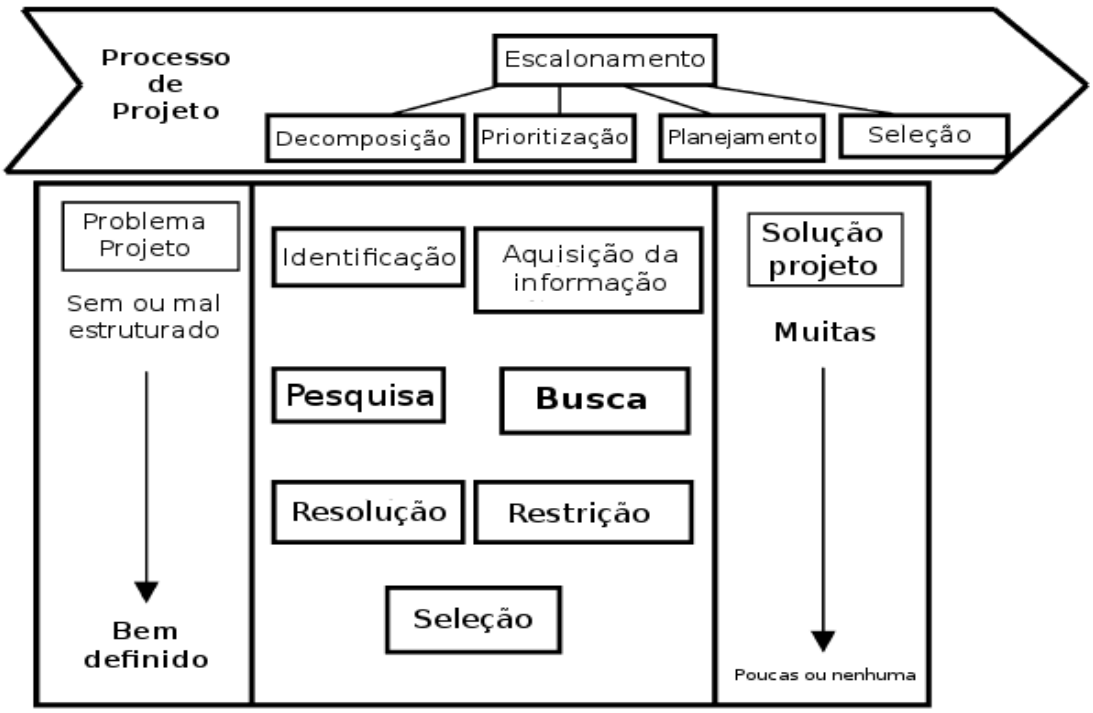


Figura 3.14 – Taxonomia genérica de atividades de gerenciamento de projeto.

- Interconexões entre componentes: reutilização, herança, decomposição e interconexões entre si próprias.
- Funções, ou comportamentos, tais como relações e interconexões.

A utilização de ontologias nesta área, segundo os autores, permite uniformizar a interpretação e análise em projeto através de utilização de semântica e taxonomia. A solução proposta modela a informação em projeto colaborativo por ontologias em níveis do mais específico, ou instâncias/indivíduos, ao mais conceitual/abstrato, ou classe, conforme a figura 3.15.

Utilizando esta abordagem descrita resumidamente, ou seja a subdivisão utilizando ontologias e níveis (M0, M1, M2), é possível sua extensão incluindo outras descrições em projeto colaborativo tais como forma (geometria, material por exemplo) e artefato (modelo e tipo de indivíduo). Adicionalmente, é possível qualificar os extremos das conexões com expressões como /montagemDe, /subartefatoDe, por exemplo. Este tipo de característica enriquece a representação de modelos de produtos desenvolvidos colaborativamente além de permitir uma descrição flexível, amigável e precisa para combinar e verificar sua consistência.

Este trabalho possui, associada, grande relevância com esta pesquisa pois forneceu os subsídios sistêmico-ontológicos para defini-la globalmente. No caso, esta pesquisa possui seu domínio e escopo ontológicos de modelagem equivalentes ao nível M0 e M1 da pesquisa de Bock *et al*.

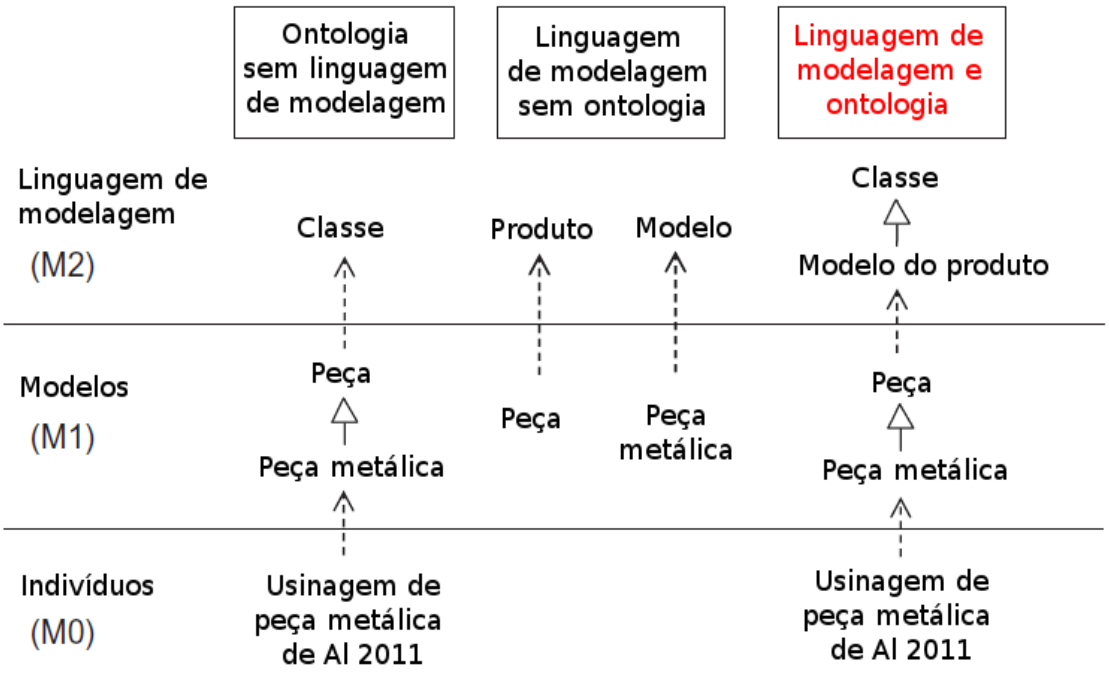


Figura 3.15 – Linguagens possíveis de modelagem de produtos.

Wei, Qin-yi e Tian-yi, 2009 desenvolveram uma ferramenta baseada em conhecimento utilizando ontologias, *Product Knowledge Model* (PKM), para redução na busca e interpretação adequada da informação. A redução no tempo total de um projeto de produto, que pode chegar até 30% do tempo total de projeto, inclui a busca efetiva e eficiente por quatro tipos conhecimento, geralmente: recursos necessários, competência organizacional, tecnologia do produto e processos de negócios. A utilização de ontologias nesta ferramenta é devido à variabilidade da informação, bem como seu tamanho e abrangência. Desta forma, existem ontologias para estes quatro tópicos descritos capacitando que os projetistas sempre possuam a informação semanticamente válida. A implementação da estrutura da informação é feita em OWL e SWRL [W3C Member Submission, 2004]. A busca de informação é definida segundo a equação 3.1 baseada nos requisitos do usuário Q.

$$Q = \{KC, UP, UQ, HS\} \tag{3.1}$$

onde KC é o contexto do conhecimento de uma situação (cenário) de projeto, UP são as preferências do usuário, UQ é uma pergunta (*query*) do usuário e HS é um conjunto semântico híbrido formado pelos três tópicos anteriores. O número real R representa a relevância entre os requisitos do usuário Q e o componente de conhecimento K sendo calculado por $R : D \times Q \rightarrow R$.

O processo de busca da informação, ou assistente de projeto, é feito a partir da obtenção do conhecimento dos requisitos do usuário incluindo suas preferências, cenário e *query*. O sistema constrói o modelo de busca do conhecimento (KRM) e utiliza o

algoritmo descrito para calcular a relevância *R* entre *Q* e *K*. Um protótipo cliente–servidor foi desenvolvido e implementado em J2EE; o sistema possui quatro camadas: nível de conhecimento e fonte de dados (RDBMS e OWL), nível do domínio (subsistemas de suporte e administração do conhecimento), nível de rede – *Web* (servidor HTTP) e interface com o usuário (acesso a aplicações, acesso aos serviços de projeto, acesso ao conhecimento e ontologia). Embora disponibilizando uma métrica relevante, que racionaliza a validade de um determinado cenário de projeto no domínio de manufatura, esta pesquisa não possui interligação direta com pesquisa-foco expressa neste documento. Na verdade, como DFM é fundamentado em regras e princípios, que no caso de ontologias possuem uma taxonomia diferente, a utilização ótima da pesquisa revisada seria *a posteriori* numa ontologia mais completa deste domínio.

Borst, 1997 propõe uma forma de projetar e representar ontologias sob um contexto quasi–algorítmico. Neste caso, entretanto, o desenvolvimento de ontologias tem como foco um domínio específico, a configuração e análise da desmontagem (*dissassembly*) de um veículo. A estrutura ontológica do domínio de análise da desmontagem de um veículo, PROMOD, é fundamentada em PhysSys (ontologia–base de componentes físicos), bem como descrita em função de seus objetos interligados e *loops* com operações de desmontagem e sub-montagens eventuais. A implementação da PROMOD é baseada na estrutura da informação da PhysSys com métodos associados de análise de desmontagem, geometria e suas ontologias respectivas. Aspecto relevante nesta pesquisa é que a estrutura sistêmica fundamentalmente é definida, descrita e utilizada inicialmente apenas semanticamente. O aspecto de implementação de ontologias não é seu foco *a priori*, ou seja, é apenas uma consequência da estrutura e das relações entre componentes da informação.

Borst, Akkermans e Top, 1997 utilizam a ontologia PhysSys como fundamento, extendendo–a para outros domínios através da subdivisão de ontologias mais amplas/genéricas. Esta abordagem possibilita a utilização de PhysSys como base conceitual para desenvolvimento de ontologias em diferentes campos, tais como sistemas termodinâmicos ou o sistema térmico de um hospital, por exemplo. A efetivação da abstração e representação de conceitos de forma genérica permite que posteriormente eles sejam categorizados com suas propriedades segundo uma forma/padrão. A categorização permite a definição de partes ontológicas mais específicas mantendo coerência conceitual consistente, enquanto não aumenta o agrupamento, ou " união", de conceitos. Esta solução, na verdade, reduz a dependência ontológica descrita pelos conceitos. Adicionalmente, é introduzido o conceito *projeção ontológica* para conectar ontologias distintas. As projeções ontológicas podem variar desde relações simples do tipo inclusão–especialização até relações mais complexas dependentes cada vez mais do conhecimento. Sua taxonomia é muito abstrata sob contexto conceitual, inclusive, portanto não aplicável ao problema resolvido nesta pesquisa, que é conceitualmente mais definido.

Bittner, Donnelly e Winter, 2005 verificam a relevância da representação contextuali-

zada da informação, particularmente em ambientes de projeto 3D. Neste contexto, Cho, Han e Kim, 2006 verificaram que a integração direta da informação representada em bibliotecas de peças é geralmente limitada pela heterogeneidade semântica. Esta heterogeneidade refere-se essencialmente à abstração e representação da informação no caso múltiplas bibliotecas de peças. A solução desta limitação foi o desenvolvimento de uma ontologia de representação unificada de biblioteca de peças de forma que uma interpretação, ou visão, comum esteja disponível para o desenvolvimento de ontologias de peças de domínios específicos. Embora a descrição conceitual de Bittner e Donelly seja consistente, inclusive semanticamente, para os domínios CAD/AEC/GIS, ela não é utilizável no domínio desta pesquisa que, neste aspecto, eventualmente utilizará a arquitetura STEP.

Neste contexto de integração descrito, Panetto, Assisti e Tursi, 2012 desenvolveram uma arquitetura de gerenciamento de dados de produto (PDM) focada na interoperabilidade com ambientes de manufatura. Os dados, em realidade, são conceitualizados, expressos e utilizados em uma visão abstrata como informação. A hipótese básica desta pesquisa é que este tipo de abstração, que utiliza ontologias, é o mais adequado para ambientes deste tipo, portanto facilitando a interoperabilidade de todos componentes, ou ferramentas, de *software*. A utilização contextual da informação é especificada a partir dos aspectos mais abstratos do processo integrado da utilização da informação em um sistema de manufatura, que utiliza ISO 10303 através da STEP PDM e IEC 62264, conforme descrito na figura 3.16.

Todas as atividades sistêmicas globais integradas no ambiente de manufatura são definidas. Neste caso, entretanto, um aspecto interessante e relevante relacionado a esta pesquisa é que a informação é representada baseada no produto, ou em uma classe instanciada a partir de um produto. A interpretação (e/ou compreensão) da informação para diferentes estados (*status* ou instâncias de interesse de cada componente) é realizada através de ontologias. As ontologias são utilizadas como habilitadores semânticos para correta interpretação e compreensão da heterogeneidade da informação, que inclusive pode ser distribuída sistemicamente. A pesquisa descrita, entretanto, apesar de possuir descrição semântica adequada baseada em FOL, com eventual extensão a STEP, possui maior abstração sistêmica que este trabalho, justamente por ter como foco a IEC 62264. Ainda assim, suas descrições ontológicas de produto, que englobam material e manufatura, disponibilizaram valiosas sugestões taxonômicas, particularmente em relação às ligações (*links*) relativos a materiais e manufatura.

Sánchez *et al*., 2012 ressaltaram a relevância de ontologias e o quão relacionadas elas são semanticamente, visto que, a princípio, este é o fundamento básico deste paradigma que permite a definição e contextualização da informação. Desta forma, é fundamental para que a contextualidade da informação seja corretamente analisada que seja possível quantificar/qualificar a similaridade intrínseca entre ontologias. Os três principais métodos de análise da similaridade de ontologias são: quantificação de conexões, ou *links* (*edge coun-*

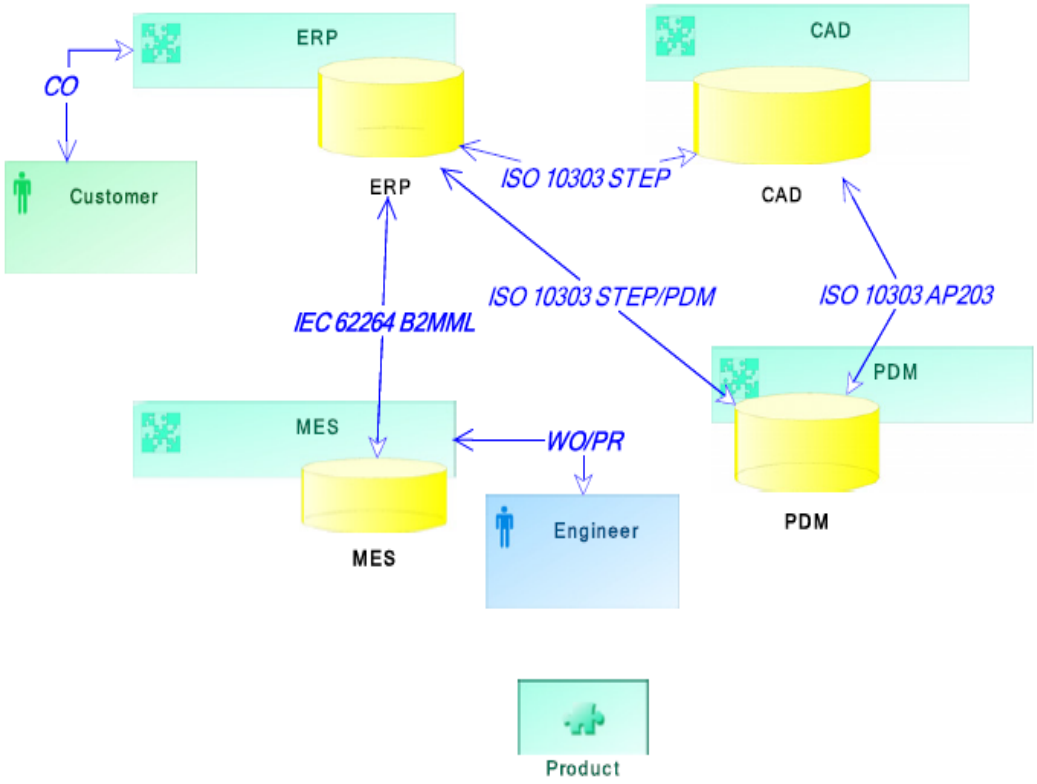


Figura 3.16 – Arquitetura ONTO–PDM de interoperabilidade de sistemas de manufatura [Panetto, Assisti e Tursi, 2012].

ting measure), quantificação de características taxonômicas pontuais básicas de informação (*feature–based measure*), ou propriedades, e quantificação do conteúdo da informação - IC (*information content measure*).

A técnica *edge counting* utiliza ontologias definidas como grafos direcionados nos quais conceitos são conectados por *links*. Taxonomicamente, em geral, os *links* correspondem a uma conexão do tipo (`é-um`), ou (`is-a`). A medida quantitativa de similaridade é obtida através do tradicional método do caminho mais curto (*minimum path length*) de Pesquisa Operacional. Quanto mais longo for o caminho menos similares são os conceitos, ou termos.

A métrica baseada em características taxonômicas pontuais básicas (*feature–based*) de informação de cada nó não é baseada puramente na distância, conforme descrito, visto que as distâncias entre nós não necessariamente são uniformes. Nesta abordagem, a qualificação da similaridade é feita considerando adicionalmente propriedades de cada nó. A medida considera as propriedades comuns e não comuns entre nós utilizando o modelo de similaridade de Tversky, 1977. O contexto semântico é mais profundamente e adequadamente definido nesta abordagem devido à comparação entre aspectos comuns e não comuns dos conceitos representados pelos nós.

A técnica do conteúdo da informação é fundamentada na probabilidade de cada palavra, representado por um nó, na taxonomia e calculado em função de sua ocorrência em

um determinado *corpus* de informação (IC). O IC é calculado como o negativo do logaritmo da probabilidade de ocorrência de um conceito, *a* por exemplo, segundo a equação 3.2. O cálculo do IC desta forma permite concluir que palavras infrequentes tenham maior valor que palavras frequentes.

$$IC(a) = -\log P(a) \tag{3.2}$$

Resnik, 1995 postula que a similaridade semântica é relacionada com a quantidade de informação compartilhada entre dois conceitos. Esta quantidade é mais tipicamente quantificada através do número de *links* de cada conceito com seu antecessor taxonômico comum (LCS), segundo a equação 3.3. Logo, se dois conceitos são dissimilares ao máximo seu LCS é zero. Sánchez *et al.* fundamentam sua métrica em dissimilaridade puramente nas características (*features*) taxonômicas, eliminando o requisito de designar pesos a características taxonômicas semânticas potencialmente mais escassas. Desta forma, existe aumento na generalidade da métrica. A equação 3.4 descreve o conjunto de características taxonômicas de um conceito *a* enquanto a equação 3.5 descreve a fórmula da dissimilaridade normalizada entre dois conceitos *a* e *b*[7].

$$sim_{res} = IC(LCS(a,b)) \tag{3.3}$$

$$\phi(a) = c \in C | a \leq c \tag{3.4}$$

$$dis_{norm}(a,b) = \log_2\left(1 + \frac{|\phi(a) \setminus \phi(b)| + |\phi(b) \setminus \phi(a)|}{|\phi(a) \setminus \phi(b)| + |\phi(b) \setminus \phi(a)| + |\phi(a) \cap \phi(b)|}\right) \tag{3.5}$$

Esta pesquisa permitiu calcular de forma eficaz a dissimilaridade entre termos além de fornecer uma comparação com valores ótimo e quase ótimo segundo as duas metodologias disponíveis de comparação entre correlações: Miller & Charles e Rubenstein & Goodenough. Adicionalmente, ela depende somente da taxonomia, aspecto vantajoso pois este é o mais comum. Enquanto relevante sob o contexto de análise qualitativa de uma ontologia, pois DFM, claramente, é associado, também, com regras e práticas qualitativas, que dependem de cada contexto de fabricação; no estágio atual esta pesquisa a métrica proposta somente terá utilização em um estágio posterior.

Yang, Dong e Miao, 2008 desenvolveram um sistema fundamentado em ontologias para compreensão e estruturação da configuração de produtos. O sistema hierarquiza as configurações possíveis de um determinado produto. Esta hierarquia é utilizada em um meta–modelo de configuração. O meta–modelo é definido a partir de uma base ontológica de um domínio, que neste caso é uma máquina furadeira de grande porte em construção. A partir da configuração das classes e suas relações é possível obter a

[7]Seja C o conjunto de conceitos de uma ontologia logo $\leq$ é a relação binária na qual um conceito está contido segundo $C \times C$, ou subconjunto contido.

estrutura do conhecimento do domínio e a configuração do produto. A configuração do produto é feita utilizando o Método de Resolução de Restrições (CSP). Os modelos de configuração são definidos utilizando a linguagem OWL, consequentemente possuem semântica corretamente estruturada e livre de inconsistência lógica na configuração da base de conhecimento.

A meta–ontologia de configuração de produto utiliza o conceito fundamental associado com ontologias, classe, e sua subdivisão conceitual associada com o domínio. Desta forma, existem diversos tipos de classes como por exemplo component, product, sub_assembly, part, function, port, resource, etc. que são utilizadas para *conceitualizar* a configuração de um produto. Adicionalmente, as restrições descrevem, resumidamente, atributos conceituais específicos de cada classe. A associação entre as restrições, relações e as classes permite estruturar a configuração de um produto. A abordagem CSP propõe uma forma distinta desta pesquisa para solucionar os problemas relacionados a DFM, entretanto parte de sua taxonomia, especificamente a modelagem ontológica conceitual da classe Produto foi utilizada; adicionalmente a interconexão da taxonomia com um sistema especialista baseado em regras facilitou a operacionalização da inferência. Embora esta abordagem tenha disponibilizado relevante percepção sobre a estrutura taxonômica hierárquica de um produto, seu domínio, bem como escopo pois refere-se à configuração sistêmica, são díspares do desta pesquisa.

Rahmani e Thomson, 2012 desenvolveram um aspecto relevante em qualquer estrutura de informação que deseje proporcionar acesso fácil e efetivo: sua interface. Considerando que sistemas, em geral, possuem subsistemas que podem, inclusive, ser detalhados como componentes sua pesquisa propõe que as interfaces entre as partes de um sistema sejam definidas em função de portas com atributos. Uma porta possui atributos que, em geral, são de dois tipos: forma (aspectos reais, por exemplo geometria) e função (descrição qualitativa expressa como pares verbo–objeto). As portas com seus atributos possuem três partes em relação à representação da informação intercambiada. As portas passíveis de serem conectadas são obtidas a partir de regras (*mating rules*) sobre como as diferentes partes da informação se relacionam. Estas regras são tipicamente conhecidas como regras de compatibilidade (*compatibility rules*).

- Definição dos atributos das portas.
- Requisitos, ou restrições, dos atributos das portas.
- Relações de conetividade entre portas.

A ontologia é definida nos três níveis hierárquicos da informação existente nas interfaces: instância (que é o dado), domínio (informação específica da área) e meta (informação da interface associada com a abstração do domínio). Seja $C(x)$ um predicado

válido quando x é uma instância da classe C e $P(x, y)$ um predicado válido quando x é mapeado para y através da propriedade P. O requisito do intercâmbio de informação em uma porta é feito através de uma forma genérica que pode ter uma, ou ambas, equações (3.6), (3.7).

$$\begin{aligned} R \equiv & \, Porta(?x) \wedge temForma(?x, ?fm) \wedge Forma(?fm) \\ & \wedge \, temAtributo(?fm, ?t_k) \wedge Atributo(t_k) \wedge \psi_{ik}(?t_k) \end{aligned} \tag{3.6}$$

$$\begin{aligned} R \equiv & \, Porta(?x) \, \wedge \, temFuncao(?x, ?fn) \wedge Funcao(?fn) \\ & \wedge \, temFluxo(?fn, ?fl) \wedge Fluxo(?fl) \\ & \wedge \, temAtributo(?fl, ?t_k) \wedge Atributo(?t_k) \wedge \psi_{ik}(t_k) \end{aligned} \tag{3.7}$$

onde ψ_{ik} é uma restrição de um atributo $?t_k$.

A regra de compatibilidade entre dois atributos de portas, somente, é definida conforme a equação 3.8.

$$\begin{aligned} M \equiv & \, Porta(?x_i) \wedge temCompatibilidade(?x_i, ?x_j) \wedge Porta(?x_j) \\ & \wedge F(?x_i, ?y_i) \wedge temAtributo(?y_i, ?t_k) \\ & \wedge Atributo(?t_k) \wedge F(?x_j, ?y_j) \\ & \wedge temAtributo(?y_j, ?s_k) \wedge Atributo(?s_k) \wedge \psi_{ijk}(?t_k, ?s_k) \end{aligned} \tag{3.8}$$

Onde s é um subsistema, $?s_k$ um atributo de s, $?t_k$ e $?s_k$ tem o mesmo tipo e ψ_{ijk} é uma restrição em relação aos atributos $?t_k$ e $?s_k$ agrupados. As equações 3.6 e/ou 3.7, 3.8 necessitam ser expandidas para todos os n atributos ($1 \leq k \leq n$). Em resumo, uma relação de compatibilidade entre portas é equivalente a uma relação de compatibilidade entre seus atributos. A interface lógica entre duas portas é formada pelo conjunto de todas regras derivadas da composição entre compatibilidade e requisitos.

Se um atributo é alterado de forma que alguma das restrições ψ_{ijk}, ψ_{ik} seja modificada de forma incorreta, o projetista é informado sobre o erro para tomada de decisão. O conjunto de regras desta interface lógica pode ser utilizado por um sistema de regras de negócio. Este tipo de sistema é utilizado para comparar padrões definidos pelas regras e objetos específicos, definir de regras e efetuar decisões baseado no *status* de suas regras. Um sistema de controle das interfaces é responsável pela utilização das regras para verificar eventuais erros nas interfaces.

No caso do foco de um escopo de um domínio de uma ontologia requerer informações que são *rigidamente* conectadas a regras, que eventualmente podem definir dados *resultantes* de um processo de inferência, esta abordagem é efetiva e eficiente. Quando o

nível de abstração é maior, como no caso desta pesquisa, a especificação estrita de regras, estrutura de controle e queries torna-se inviável, entretanto.

Posada *et al*., 2005-2006 descrevem a complexidade da representação de modelos CAD, particularmente devido à falta de compreensão e representação adequada do contexto da informação, consequentemente conhecimento, em diferentes domínios através do padrão STEP. A utilização de ontologias é, portanto, proposta para eliminar esta limitação. O foco de análise da informação é a compreensão e revisão de dados de projeto bem como a visualização de grande quantidade de dados de sistemas CAD. Ontologias permitem a simplificação desta complexidade de interação entre dados diversos através de *análise semântica* da informação com algoritmos de categorização, simplificação e adaptação de componentes de engenharia de acordo os padrões de um determinado domínio.

Especificamente, a solução proposta utiliza análise semântica através ontologias, que utiliza STEP 10303-AP227 (configuração espacial de fábrica), para otimização da compreensão de informações de sistemas CAD agrupadas por similaridade e seu consequente agrupamento. A similaridade e agrupamento utilizam o conceito de triangulação semântica: intenção e fundamentação do usuário, recursos disponíveis e características do modelo introduzido por Posada *et al*., 2004. Através desta técnica foi possível um ganho efetivo de até 78% na compressão de triângulos dos dados de um modelo CAD de uma fábrica num determinado estudo de caso. Embora seja uma abordagem válida de otimização em semântica de dados STEP, ela não possui utilização efetiva no contexto desta pesquisa; provavelmente poderá ser útil quando o arcabouço estiver pronto, integrado com STEP e necessitar utilizar as informações otimizadas neste padrão.

Jin *et al*., 2008 desenvolveram uma arquitetura implementada em um sistema de empresas de pequeno e médio porte para integração de bibliotecas de peças em um ambiente contextual de análise utilizando uma analogia da web semântica customizada para engenharia (ESW), vide a figura 3.17. A ESW é baseada em uma arquitetura contextual de informação através da internet: HowNet. A utilização e obtenção de informações contextuais, que são rigidamente conectadas aos dados, representados em XML [World Wide Web Consortium, 2000] como atributos, é feita através de um algoritmo específico utilizando o clássico algoritmo do caixeiro viajante (TSP). O foco nos dados dificulta a utilização desta abordagem em arcabouço de informação conforme o desta pesquisa, como será descrito no próximo capítulo.

Alberts, 1993 estruturou e desenvolveu um método para definição de ontologias genéricas das atividades de projeto. Em geral, segundo o autor, a atividade de projeto deve ser analisada como uma *atividade de síntese*. Neste contexto, portanto, projeto é visto como uma atividade de síntese descrita como uma série de blocos básicos conectados segundo a funcionalidade e performance necessárias de um produto em projeto. Desta forma, em resumo, a atividade de projeto consiste em encontrar a definição e configuração

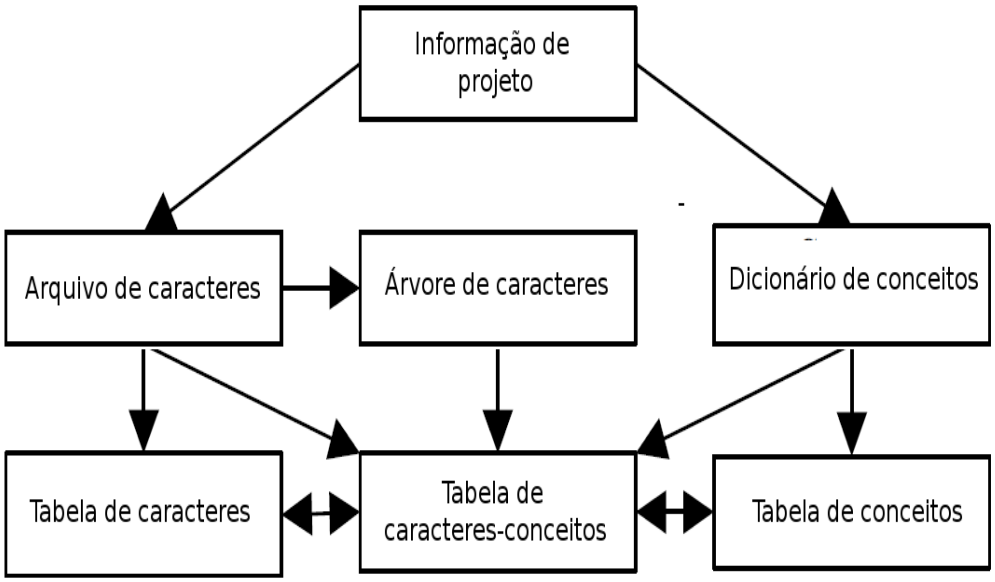


Figura 3.17 – Utilização de bibliotecas de peças como dados na arquitetura contextual–semântica de informação de engenharia ESW [Jin *et al.*, 2008].

ótima entre estes blocos. Os blocos, ou elementos, básicos são de dois tipos: forma, que corresponde à parte física, e função que é o comportamento específico, ou performance, dos elementos de uma parte de um produto (ou de um produto completo). A síntese pode ser descrita, resumidamente, a partir de *elementos genéricos de modelos* (GSM) que possuem determinadas características.

- Representação explícita das *dependências* entre forma, função e comportamento de um produto.
- *Vários níveis de abstração* variando desde a especificação original até a versão final de um produto.
- Descrição científica exata e arquitetura/estrutura *orientada ao produto*.

GSMs são, essencialmente, subsistemas interligados que perfazem a descrição completa de um determinado produto. Um tipo de representação tradicional nas características associadas ao conhecimento intrínseco de um subsistema, ou comportamento, é a representação matricial. Um exemplo de subsistemas individuais e o sistema completo de engenharia estrutural composto por suas interligações pode ser visto na figura 3.18. Este sistema de vigas considera apenas deslocamentos e forças axiais; as variáveis são as forças p, deslocamentos d e os parâmetros de forma comprimento l, área da seção A e módulo de elasticidade E.

$$\begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} . k_1 = A_1.E_1/l_1 \tag{3.9}$$

$$\begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{bmatrix} d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} p_3 \\ p_4 \end{bmatrix} . k_2 = A_2.E_2/l_2 \tag{3.10}$$

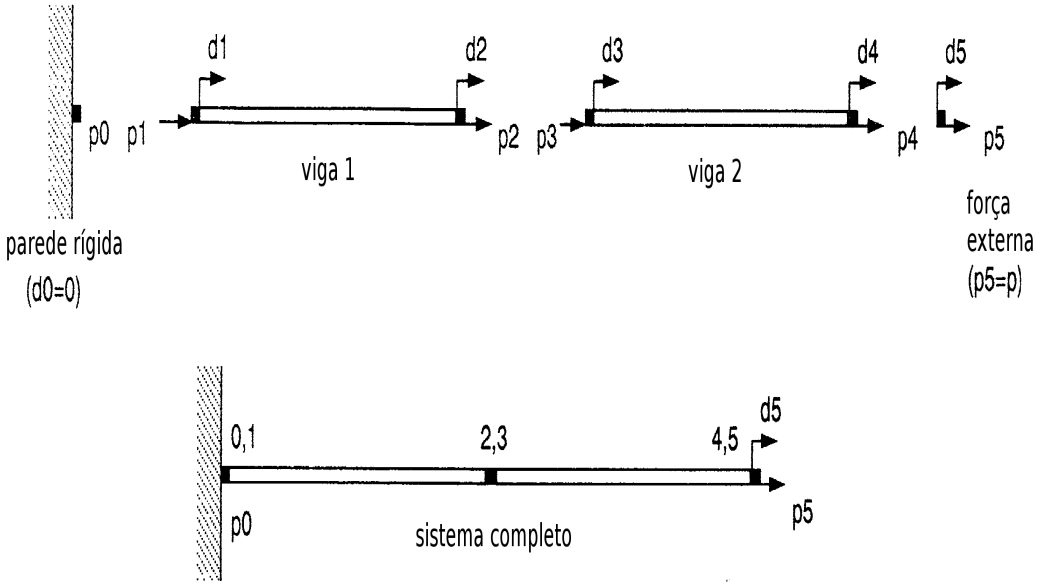


Figura 3.18 – Sistema de engenharia estrutural [Alberts, 1993].

A estrutura da interface associada com a representação da informação utiliza portas que são pares de variáveis, consequentemente seus valores, ou seja sua conetividade, é representa por ∩. Desta forma as estruturas da informação no exemplo são $[(p_1, d_1), (p_2, d_2)]$ e $[(p_3, d_3), (p_4, d_4)]$, respectivamente. O sistema completo possui as estruturas da porta 2 e porta 3 interconectada: $[(p_2, d_2) \cap (p_3, d_3)]$; estas são portas internas do sistema. As portas 1 e 4 são portas externas: $[(p_1, d_1), (p_4, d_4)]$.

A partir destes requisitos foi desenvolvida a ontologia de modelagem de conhecimento compartilhado de projeto YMIR[8], que possui um conjunto adequado de conhecimento de referência. Logo, YMIR, que pertence ao nível de conhecimento, descreve a semântica fundamental associada com a atividade de projeto. A arquitetura utiliza elementos genéricos para obter as propriedades gerais de modelos de sistemas: comportamento, forma e estrutura. Tipicamente, estas propriedades de síntese são obtidas através de algumas questões típicas.

- Quais são as variáveis que descrevem as equações pertencentes aos modelos matemáticos de um sistema?

[8]Curiosidade: segundo a mitologia alemã Ymir era um ser gigante antigo. Ymir foi morto pelos deuses Odin, Vili e Ve, que após este ato criaram a terra, céu e água a partir das partes deste corpo. O mundo formado por estas partes, Midgard, foi dado para as pessoas viverem.

- Quais são o comportamento, forma e estrutura definidos do projeto?
- Quais são os GSMs básicos, ou primitivos, ou seja conjunto básico de equações por exemplo, no projeto de sistemas?

A abstração de um sistema torna–se mais específica conforme um projeto é finalizado para produção. A ontologia leva em consideração esta especificidade crescente de um determinado projeto através de camadas técnicas, ou tecnológicas (TL), que são compostas por GSMs. Os GSMs também são mais específicos e com menor abstração conforme maior possibilidade de efetivação/fabricação de um projeto ocorrer de acordo com a hierarquia destas camadas.

Especificamente, os componentes genéricos (GC), que no início de um projeto são conhecidos como componentes genéricos primitivos (PGC), são alterados nas TLs, inclusive eventualmente dentro de TLs, através da modelagem por GSMs por funções conforme o projeto progride. Estas funções podem ser de combinação, ou inclusive agregação, com formação de instâncias parciais. A reinterpretação da informação de PGCs na TL adjacente envolve a tradução da implementação da função na TL corrente em uma especificação funcional na camada–objeto em função dos vocabulários dos PGCs nestes níveis. Esta sequência de etapas é executada no processo de projeto até que uma representação viável seja obtida satisfazendo os requisitos funcionais. Uma das formas de representação da informação associada com as funções das GSMs é a através de matrizes conforme as das equações 3.9 e 3.10.

A utilização da estrutura de conhecimento é feita através de uma máquina de inferência denominada ODIN. A máquina de inferência associa solicitações de informação à ontologia utilizando a descrição contextual indireta armazenada nas matrizes das equações do modelo–exemplo descrito, tradicionalmente significado, objetivo e comportamento da matriz de rigidez. ODIN é uma forma básica de representação da síntese dos conceitos associados à estrutura do conhecimento na YMIR.

YMIR é fundamentada em modelos de comunicação em rede de Teoria de Sistemas e é capaz de representar a estrutura do conhecimento de projeto em diferentes domínios e com distintos níveis de granularidade da informação. Adicionalmente, ODIN permite inferência dos principais tipos de classes associadas com síntese de acordo com a granularidade utilizada.

Chang, 2008 desenvolveu uma abordagem baseada em ontologias para o problema de informação em DFM. A abordagem ontológica proposta gerencia mais eficientemente a variedade de dados em engenharia. DFM, especificamente, é caracterizado pela complexidade relacionada com múltiplos critérios técnicos e econômicos, formas variáveis de relacionamentos e objetivos simultâneos. Desta forma, uma ontologia genérica é proposta na qual as restrições lógico–matemáticas de DFM são inclusas.

Adicionalmente, são propostos e desenvolvidos um editor de ontologias DFM embutido em uma ferramenta computacional pré–existente para gerenciamento da alteração dinâmica intrínseca no processo de projeto e uma ferramenta de tomada de decisão como suporte a decisões de projeto mais eficientes e eficazes. Finalmente, uma ontologia de análise de erros, aspecto também bastante comum em projeto, é proposta. Os exemplos DFM utilizados são análise de custo, material da solda e maquinário de soldagem MIG, seleção da ponta de solda em *Friction Stir Welding* (FSW), plataforma ontológica de integração de dados e suporte a tomada de decisão em FSW.

Aspectos taxonômicos conceituais de DFM considerados relevantes para a taxonomia da desta pesquisa foram utilizados como, por exemplo, a separação dos conceitos de produto genérico, manufatura, características (*features*) de geometria e fabricação e, particularmente, condição (equivalente a uma regra) em classes distintas. Como a taxonomia depende do domínio e escopo, bem como das hipóteses formuladas no seu projeto, embora os conceitos sejam análogos toda a estrutura informacional consequente é diferente.

Sarder, 2006 desenvolveu um método explícito baseado em ontologias para estruturação da informação integrada entre produtos e processos. O Processo de Aquisição de Conhecimento de Projeto (DKAP) é baseado em IDEF5 para modelagem do contexto da informação e conhecimento através de assertivas sobre objetos reais, suas propriedades e inter–relacionamentos. A consistência da informação é verificada e garantida através de uma matriz relacionando aspectos específicos da informação. A partir da matriz de consistência, a ontologia é gerada como um conjunto de termos organizados hierarquicamente.

Sob o aspecto teórico, a pesquisa disponibiliza uma estruturação consistente e efetiva da utilização de ontologias no domínio da integração entre produtos e processos. Seu foco é, primariamente, utilizar ontologias para definir a taxonomia e documentar o conhecimento de produtos e processos industriais metodologicamente, ou seja, mais sob contexto sistêmico, sem definição explícita, visando posterior utilização como fonte de conhecimento. Desta forma, além do domínio e escopo serem diferentes desta pesquisa, sua abstração também é limitando sua utilização efetiva.

Sun *et al*., 2012 utilizam ontologias na análise das atividades envolvidas nas operações internas dos componentes de desenvolvimento colaborativo integrado de produtos (FOM), por exemplo programas CAD e/ou CAPP, enquanto respeitando a arquitetura de alto nível (HLA). A ontologia de desenvolvimento colaborativo de produto utilizada é segundo a equação 3.11.

$$O ::= (C, H_C, R_C, H_R, M, R_M, A) \tag{3.11}$$

onde O é uma tupla dada por,

- C é um conjunto de conceitos de desenvolvimento colaborativo de produto.

- H_C é o conjunto de solicitações parciais sobre conjunto C. As solicitações são relações herdadas dos conceitos envolvidos em uma determinada asserção. Os conjunto de conceitos e relações herdadas forma um grafo acíclico direcionado (DAG).

- R_C é um conjunto não–herdado de relações parciais de solicitações do conjunto C que correspondem aos atributos de conceitos.

- H_R é o conjunto de relações herdadas do conjunto de relações parciais do conjunto R_C.

- M é o conjunto de conceitos de meta–ontologia de produto colaborativo. Este conjunto é formado por instâncias de R_C passíveis de serem herdadas.

- R_M conjunto de solicitações parciais do conjunto M que descrevem as relações entre elementos do conjunto de meta–ontologia. Este conjunto forma a base para o raciocínio sobre produto colaborativo.

- A é o conjunto de axiomas sobre o conjunto de conceitos ontológicos e o conjunto de relações da meta–ontologia. Este conjunto forma as premissas principais sobre o raciocínio com ontologias de desenvolvimento colaborativo de produto.

A ontologia desenvolvida possui duas camadas: abstrata e de aplicação. A ontologia abstrata é uma meta–ontologia enquanto a ontologia de aplicação, ou domínio, analisa conceitos específicos relacionados a um determinado componente da arquitetura. A meta–ontologia descreve o raciocínio *per se*: regras e gabaritos associados com a forma de utilização dos componentes de projeto de produto colaborativo. A ontologia de domínio é desenvolvida a partir de arquivos de análise formal de conceitos (FCA) e modelo de simulação de objeto (SOM). O conjunto dos dois tipos de ontologias permite a definição distinta de axiomas associados com cenários de utilização de aplicações–componentes conforme a figura 3.19.

A meta–ontologia armazena a forma ótima de inferência sobre os conceitos de um domínio, ou ontologia do domínio (figura 3.20). Essencialmente, a meta–ontologia, ou ontologia de interoperação, descreve modelos de interoperação orientados ao objeto possibilitando utilização de herança e extensão de forma simplificada.

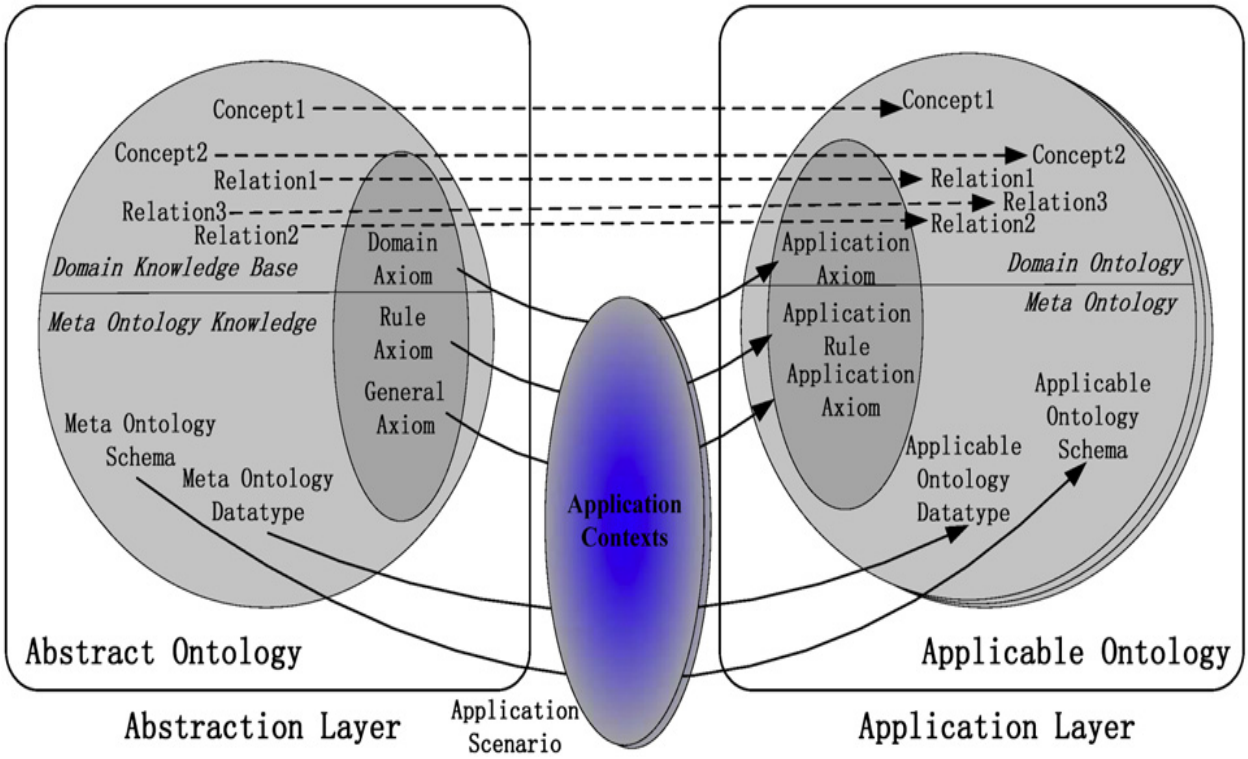


Figura 3.19 – Utilização de ontologias em camadas no desenvolvimento colaborativo de produtos [Sun *et al.*, 2012].

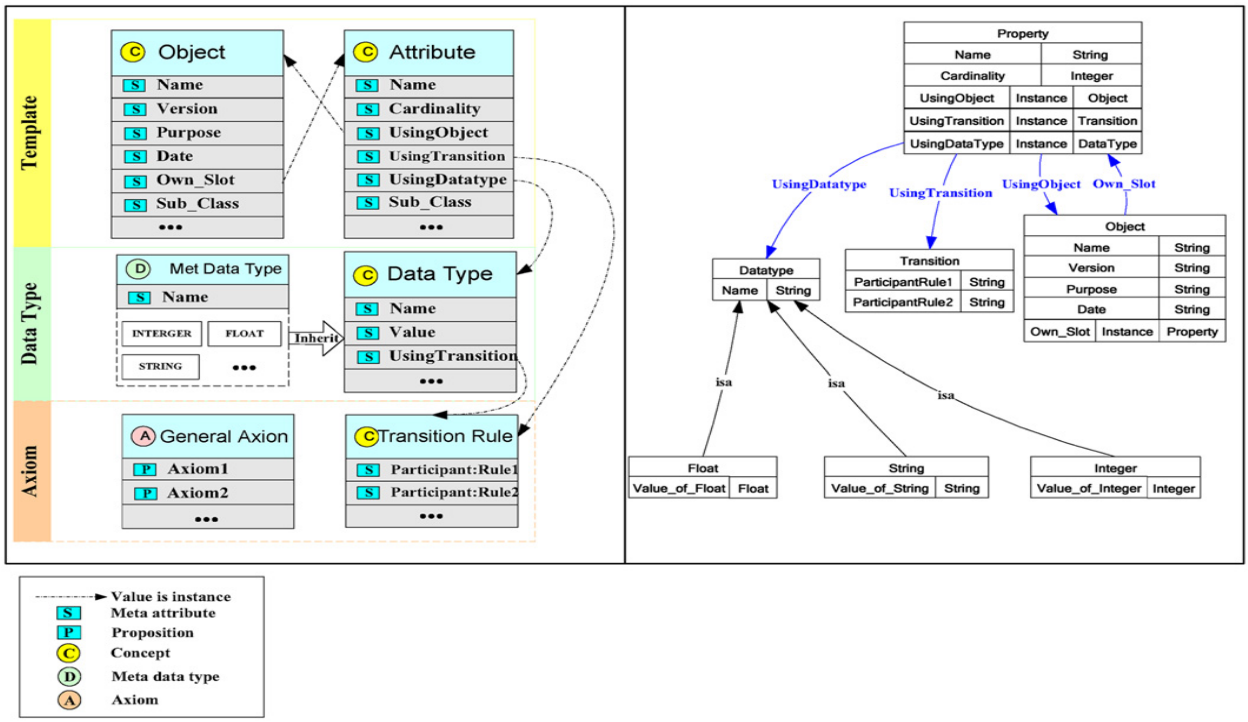


Figura 3.20 – Meta–ontologia de desenvolvimento colaborativo de produto [Sun *et al.*, 2012].

A abordagem proposta simplifica o desenvolvimento colaborativo de produto baseado em ontologias, pois introduz o conceito de meta–ontologia. Além disto, é superior à situação dos métodos anteriores, que assumiam a existência e/ou disponibilidade de

uma ontologia. A existência de uma ontologia previamente desenvolvida é um fator complexo em HLAs devido a aspectos de semântica funcional, consistência implícita entre componentes e a própria compatibilidade interna da HLA. Formalmente, enquanto a pesquisa demonstrou a estruturação mais adequada da informação através de ontologias em sistemas heterogêneos, sob certo aspecto análogo com DFM, ao mesmo tempo conclui que uma solução efetiva em problemas industriais reais todavia persiste. Em relação ao foco desta pesquisa, a abstração, logo sua taxonomia, é díspar devido a ter um domínio e escopo diferente, porém auxiliou-a através da necessidade de conhecer e estruturar melhor taxonomicamente a própria ontologia através de sua meta descrição, por exemplo.

Considerando a complexidade associada com a informação e seu gerenciamento no transcorrer do desenvolvimento de projetos de reatores químicos, Brandt *et al*., 2008 desenvolveram um sistema de gerenciamento do conhecimento (KMS) para o domínio de reatores químicos. O sistema gerencia a complexidade da informação, que possui aspectos heterogêneos tais como dados, documentos, formas de trabalho, descrição de atividades de tomada de decisão, bem como suas interdependências através de ontologias. Um sistema de armazenamento de dados de processo (PDW) é baseado no armazenamento e disponibilização de ações e sua interação (*trace*) do domínio de projeto de reatores (figura 3.21).

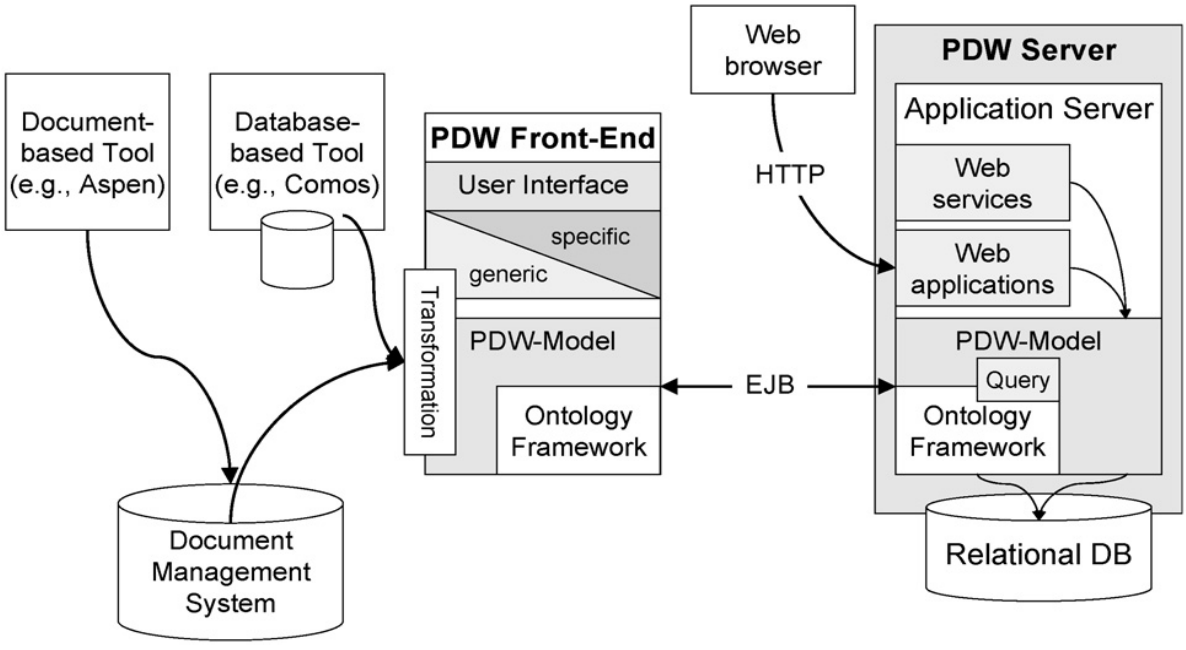


Figura 3.21 – Sistema de conhecimento baseado em ontologias para projeto de reatores químicos [Brandt *et al*., 2008].

As ontologias do PDW tipicamente podem ser desenvolvidas individualmente, porém são integradas através de uma ontologia central, ou *Core Ontology*. Esta ontologia de nível mais abstrato possui conceitos de nível mais elevado que descrevem produtos e processos assim como suas inter-relações e dependências sem considerar um domínio

específico [Brandt *et al.*, 2006]. A ontologia *Core Ontology* genericamente subdivide a informação em quatro áreas: produto, armazenamento, processo e descrição (ou descritiva).

A área descritiva contém os conceitos básicos, ou meta–conceitos (meta–informação), que possibilitam a utilização efetiva das ontologias das outras áreas. Esta ontologia contém conceitos básicos descrevendo classes de produtos (*ProductObjects*) semanticamente. Estes conceitos descrevem um vocabulário específico que descreve adequadamente os conteúdos de um *ProductObjects*. A meta–ontologia é baseada em uma extensão que especifica melhor o conceito, denominada OntoCAPE [Morbach, Yang e Marquadt, 2007]; esta ontologia foi customizada de acordo com os requisitos deste domínio. Neste caso, especificamente, OntoCAPE estende a área descritiva customizando a classe de descrição do conteúdo ContentDescription e fornecendo um vocabulário específico de classes de objetos de produtos *ProductObjects* e processos *ProcessObjects*.

A ontologia desenvolvida possui flexibilidade em modelos de dados e processos para gerenciar processos dinâmicos e criativos, ambiente de navegação e busca da informação baseado na Web Semântica. Adicionalmente, tem representação integrada de recursos, descrições do conteúdo e contexto organizacional, permitindo a reutilização do interfaceamento do usuário com o sistema, bem como funcionalidade para aquisição semi–automática do interfaceamento do usuário com o sistema durante a atividade de projeto. Esta pesquisa utiliza KAON[9] para busca ótima de informação; este sistema é uma arquitetura mais eficiente que as tradicionais baseadas em OWL puro, porém existe comparativamente certa perda de expressividade. Entretanto, a utilização desta forma de inferência diminui o *trade–off* tradicionalmente necessário em sistemas reais entre eficiência implementacional e poder lógico–descritivo.

Devido a seu domínio e escopo, esta solução difere, em geral, desta pesquisa. Entretanto, sua abordagem de disponibilização ótima da informação no processo de projeto focada na flexibilidade de ontologias demonstra a eficácia e eficiência deste tipo de solução.

Colace, Santo e Napoletano, 2009 propõem a utilização de ontologias para configuração de produtos. Esta abordagem, segundo os autores, decorre da necessidade do mercado de compreensão adequada dos desejos e requisitos do cliente enquanto respeitando os limites de custo impostos pela produção em massa. Neste sentido, ontologias são utilizadas para representar a estrutura do conhecimento e informação necessários na customização real e efetiva de um produto.

A relevância da análise apropriada da semântica é que, adicionalmente, ela é dividida em níveis segundo sua abstração é expressa. Neste contexto, tipicamente ela é subdividida em alguns elementos incluindo palavras, conceitos e percepções. O sistema executa a configuração do produto segundo associação palavra–palavra com análise semântica e probabilidade conjunta. A probabilidade conjunta se refere a quanto duas

[9]KAON é um arcabouço para ontologias [Oberle *et al.*, 2004].

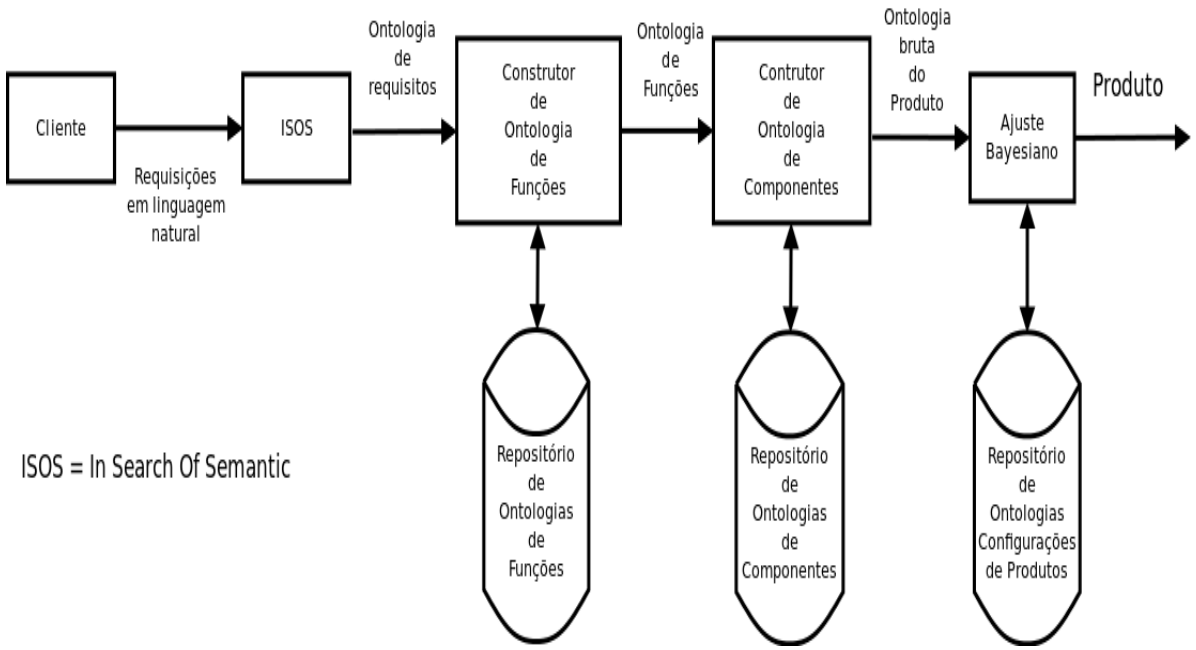


Figura 3.22 – Ontologia para configuração de produtos [Colace, Santo e Napoletano, 2009].

palavras são estatisticamente dependentes. Desta forma, a expressão de probabilidade conjunta entre duas palavras w de um conjunto de tópicos T, cada uma com probabilidade independente z, é dada segundo as equações 3.12 e 3.13. A arquitetura proposta de configuração de produtos utiliza um conjunto definido de conjuntos e funções, ambos descritos conforme a tabela 3.4.

$$P(w_i, w_j) = P(w_i|w_j)P(w_j) = \sum_{k=1..T} P(w_i|z_i = k)P(w_j|z_j = k) \tag{3.12}$$

$$P(w_i) = \sum_{k=1..T} P(w_i|z_i = k)P(z_i = k) \tag{3.13}$$

Tabela 3.4 – Conjuntos–base para ontologia de configuração de produto.

Notação	Conjunto	Função
N	Novos requisitos do cliente	**n** : $N^N \rightarrow F^N$ **n** é uma função que define as necessidades do cliente e as relaciona com as funcionalidades do produto
C	Novos componentes do produto	**g** : $C^N \rightarrow C^N$ **g** é uma função que define o conjunto de novos componentes agregados
	Novas funcionalidades	**f** : $C^N \rightarrow F^N$ **f** é uma função que define as funcionalidades de um novo componente obtido da agregação de outros componentes
F	Funcionalidades dos componentes	**i** : $F^N \times R^N \rightarrow O^N$ **i** é uma função que descreve a ontologia das funcionalidades de um novo componente
O	Ontologia de componente	**h** : $C^N \times R^N \rightarrow O$ **h** é uma função que retorna uma ontologia que explica como o componente foi obtido a partir de agregação e relacionamentos relativos
R	Relações entre componentes do produto	

Após a entrada de informações do cliente em linguagem natural no sistema através de ISOS uma ontologia resultante é desenvolvida no Construtor de Ontologias (figura 3.22), que mapeia a descrição do cliente em funcionalidades efetivas do produto.

Subsequentemente, esta ontologia, que representa explicitamente e implicitamente as funcionalidades do produto desejado pelo cliente, é utilizada para gerar as ontologias dos componentes do produto. Finalmente, o ajuste bayesiano analisa qualitativamente as alterações nas ontologias em função das ontologias anteriores. Análise prática real da abordagem desta proposta mostra que 60% dos clientes ficam satisfeitos com o produto resultante a ser manufaturado, comparativamente com 37% de clientes satisfeitos segundo técnicas tradicionais. A pesquisa demonstra a flexibilidade e relevância, juntamente com eficiência e eficácia, da abordagem de ontologias para classificar e mapear a informação em níveis de abstração deveras distintos, tal como o descrito acima, ainda que seja diferente do foco deste trabalho.

A otimização em projeto de engenharia baseada em ontologias é proposta por Witherel, Krishnamurty e Grosse, 2007. O objetivo principal deste trabalho é um aspecto frequentemente desconsiderado: a representação do conhecimento de projeto, que tipicamente é de alto nível e abstrato. Particularmente, na área de otimização de projeto aspectos desde sua terminologia, incluindo sua parte léxica, podem ser caracterizados limitadores de uma melhor compreensão e eventual utilização da informação. A taxonomia das técnicas de otimização foi desenvolvida a partir de vocabulários conhecidos, aceitos e populares na literatura. As propriedades (*slots*) da classe raiz da taxonomia de otimização são subdivididas em três grupos: genérico, relacionadas especificamente com a otimização e relacionadas à técnica, ou método, utilizado. Dois exemplos de otimização de projeto são resolvidos utilizando o mapeamento entre dados e ontologias -ontop- [-ontop-, 2015] desenvolvida no *software* Protégé [PROTÉGÉ TEAM, 2014].

A ontologia é instanciada com os dados de um projeto e o problema específico de otimização é definido. O resultado da primeira iteração, pois é um processo de otimização, por exemplo um parâmetro, é adicionado como uma instância na ontologia e o processo pode ser repetido até que valore(s) ótimo(s) desejados seja(m) obtido(s). Os autores corretamente expressam, em acordo com o fundamento teórico da área, que embora sua ontologia permita resultados consistentes todavia ela não é completa. Para tal, pesquisa mais avançada seria necessária de forma a definir mais a taxonomia hierárquica e operacional da ontologia neste domínio e escopo. Em relação a esta pesquisa, embora o domínio e o escopo sejam bastante díspares, o detalhamento e documentação sobre as classes foi utilizado através de analogias.

Garcia-Crespo *et al*., 2010 desenvolveram uma abordagem genérica para representação de processos industriais através de sua semântica. A semântica é utilizada como base para representação e utilização do conhecimento através da estrutura formal, confiável e madura proporcionada por ontologias. Desta forma, as unidades básicas de um processo industrial de manufatura são compostas por um sequência análoga a processos dinâmicos reais na sua forma mais simples *Objetos* e *Atributos*. Estes são sujeitos a determinados *estados* durante um processos denominados *situações*; as situações implicam em ações

consequentemente gerando um novo estado. Conceitualmente, o modelo proposto é descrito a seguir. A figura 3.23 descreve o modelo conceitual e seus atributos enquanto as figuras 3.24a e 3.24b descrevem contextualmente um processo de manufatura e sua representação por ontologia, respectivamente.

Ações operacionais correspondem às ações que devem ser realizadas durante o processo de manufatura. As ações podem ser executadas automaticamente ou através de um agente externo.

Verificação são as pré-condições necessárias para que uma ação operacional seja executada.

Fato é uma informação relevante para a execução de um processo, por exemplo em manufatura o número de máquinas, seu estado, produto(s) e sua(s) característica(s), etc.

Decisão são as condições que determinam que o próximo passo no processo possa ser executado após uma ação tenha terminado.

Situação é um conjunto definido de fatos relevantes juntamente com o contexto que descrevem o estado de processo em um determinado instante no tempo.

Contexto é o conjunto de condições inter-relacionadas para que algo exista ou ocorra [Merriam-Webster, 2004]. Logo, o contexto é modelado por objeto(s) agrupados em uma série de situações relacionadas.

O modelo da pesquisa foi implementado através de uma arquitetura cliente–servidor via Internet em um sistema de manufatura de orçamentação de lotes de tamanhos variáveis de peças diferentes. Os resultados para peças usinadas em máquinas distintas com tamanhos de lote variáveis demonstram que a diferença a favor desta abordagem, quando comparada com estimativas humanas, em geral, é positiva variando entre 3.1% e 9.9% .

Neste caso, a pesquisa fundamenta a ontologia em uma analogia entre um processo decisório e a resolução de problemas industriais, especificamente manufatura. Sob esse aspecto, esta pesquisa foi bastante útil para clarificar taxonomicamente o quanto DFM é, na verdade, um processo com o fim de melhorar um projeto de um produto, bem como o quão relevante é sua otimização através da informação. Ainda que seja consistente e válida, a taxonomia consequente difere da desta pesquisa em escopo, apesar de ser do mesmo domínio.

Ahmed, Kim e Wallace, 2007 desenvolveram uma abordagem contextual para utilização da informação em projeto de componentes físicos através de ontologias de engenharia. O objetivo da ontologia EDIT é a busca e aquisição de informações de projeto sobre montagem, componentes e sólidos característicos de um produto, ao invés de

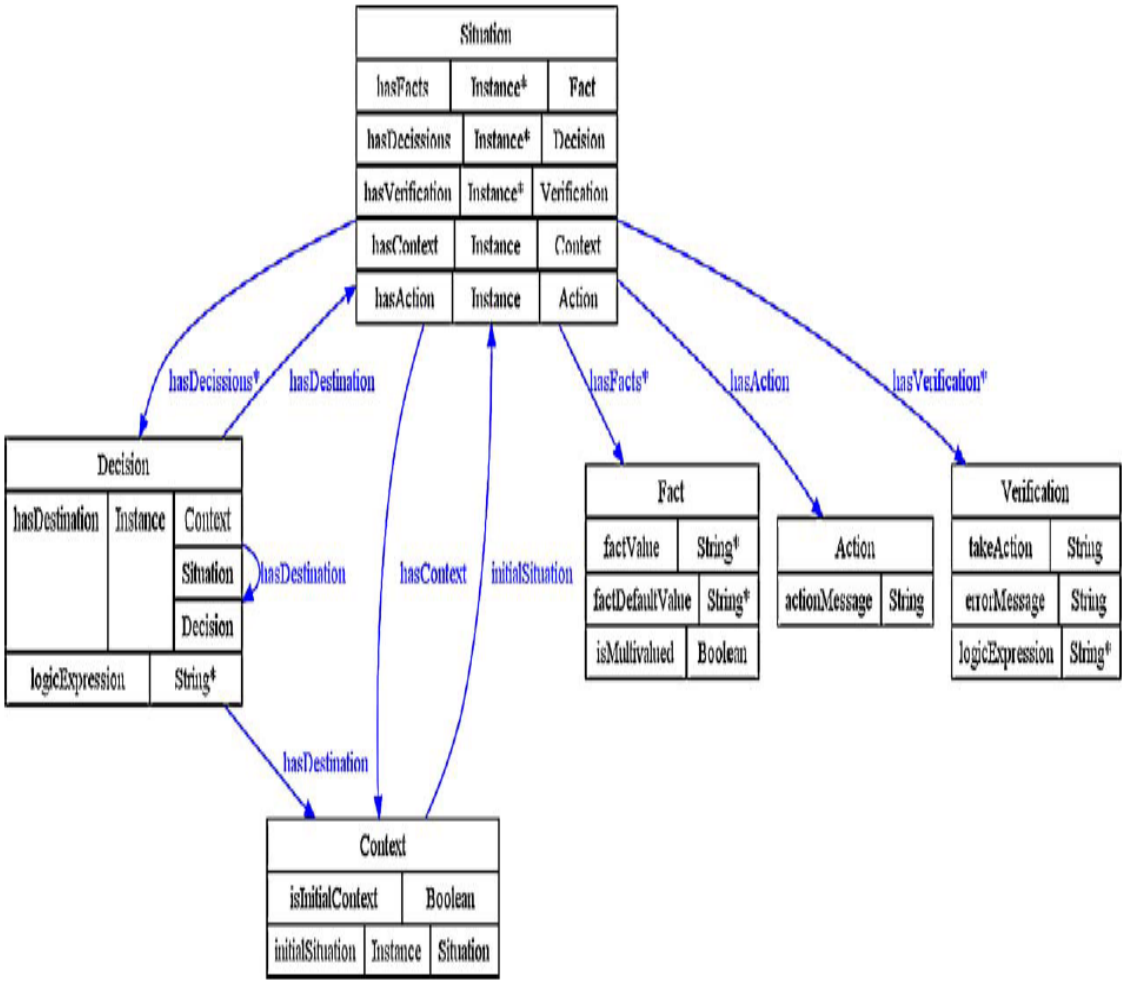


Figura 3.23 – Atributos do modelo conceitual por ontologias [Garcia-Crespo *et al.*, 2010].

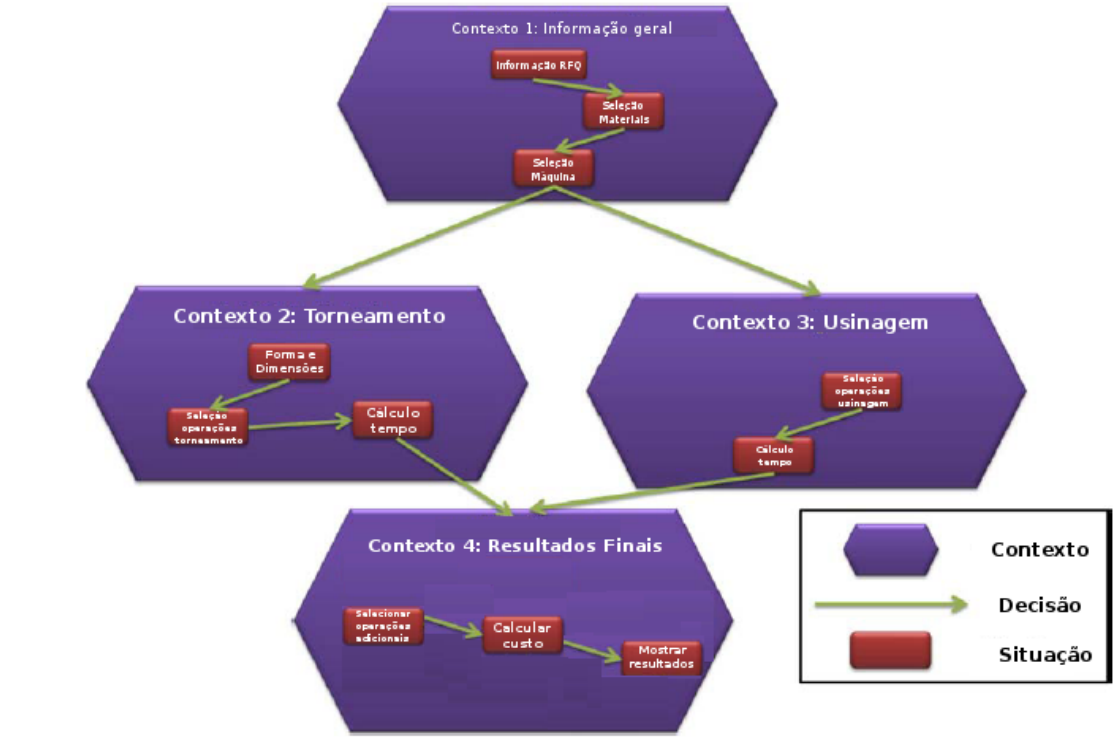
somente a definição, ou descrição, de um modelo particular, como tradicionalmente representado por arquivos (inclusive STEP na sua configuração mais simples). O método conceitualmente especifica ontologias em dois níveis e seis etapas sequenciais conforme a figura 3.25. Através desta sequência é possível identificar a taxonomia desta pesquisa; em geral ela obedece a analogia proposta por Pinto e Martinez, 2004. Resumidamente, ela possui quatro etapas, descritas abaixo, que também são utilizadas por Ahmed *et al*. Após a formalização das taxonomias, sua representação é possível através do formato de descrição de recursos (RDF), que pode ser utilizado por ferramentas computacionais de ontologias.

1. Identificação de conceitos–raiz e ontologias pré–existentes.

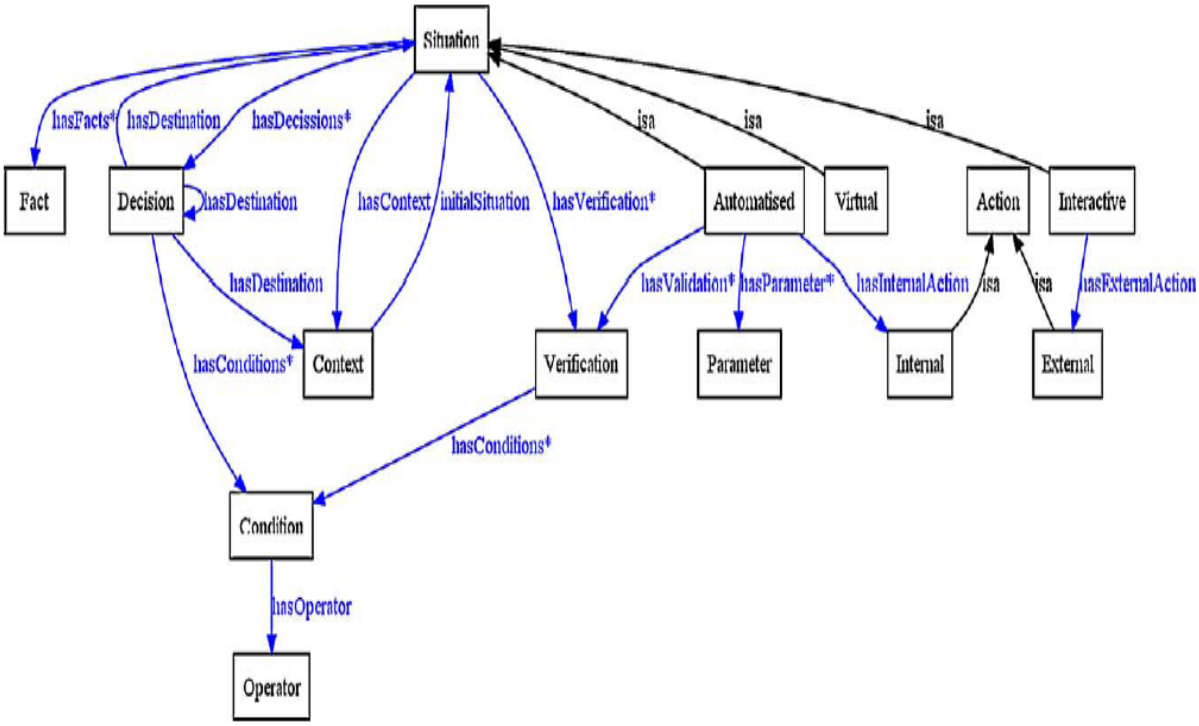
2. Criação de taxonomias.

3. Teste das ontologias e desenvolvimento do glossário.

4. Aprimoramento das ontologias.



(a) Contexto, decisão e situação em processos industriais (traduzido).



(b) Ontologia operacional do modelo de processo de manufatura.

Figura 3.24 – Integração entre processos industriais e ontologias [Garcia-Crespo *et al.*, 2010].

A definição dos conceitos básicos envolvidos em um projeto é descrita na figura 3.26a enquanto detalhes específicos de partes desta ontologia podem ser vistos nas figuras

3.28b e 3.27. Os conceitos básicos, ou raiz, da taxonomia de projeto da figura 3.26a são detalhados por,

Processo de Projetar descrição das tarefas associadas especificamente com o domínio de projeto.

Função aspectos da funcionalidade que devem ser obedecidos por um produto.

Aspecto complementos necessários para que a performance de um produto seja ela funcional, estética, resistência, térmica, ou custo seja ótima.

Produto partes de um produto projetado, por exemplo componentes, montagens e sub-montagens; inclusive componentes que compartilham uma interface física ou funcional.

O projeto–exemplo utilizado foi o local de armazenamento (*casing*) de uma turbina e a instância utilizada foi obtida através de entrevistas com projetistas da área e da literatura. O objetivo principal, além da configuração e estruturação da informação de projeto, foi servir como um repositório de informações sobre instâncias deste produto. Na pesquisa, o acesso à informação indexado de produtos reais teve resultados bastante positivos e relevantes percentualmente, demonstrando que a definição da estrutura ontológica de projeto pode ser de grande valia em geral.

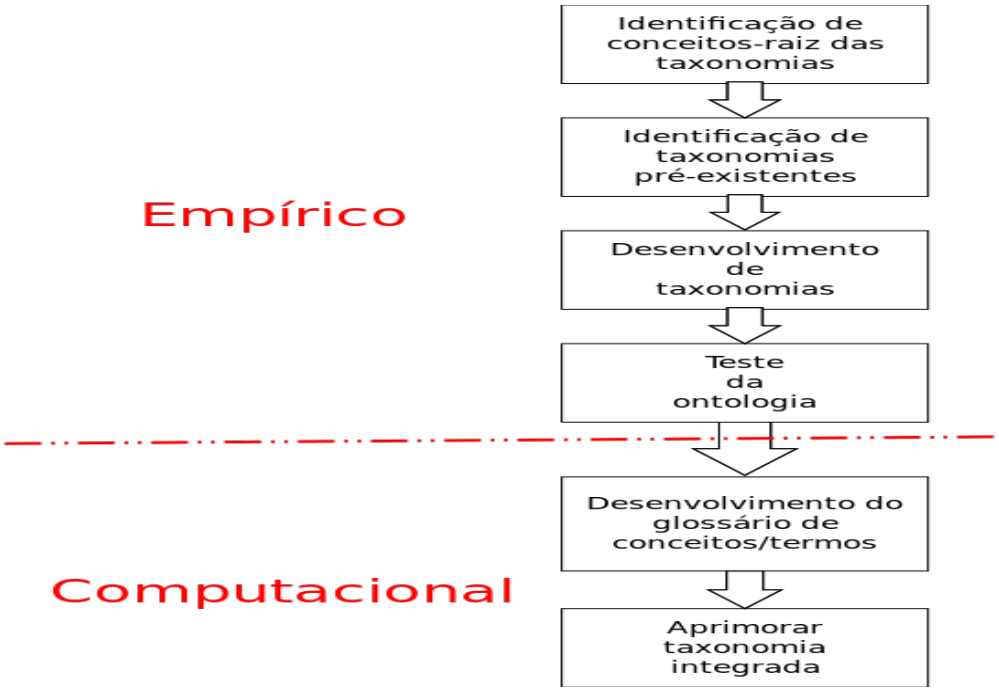
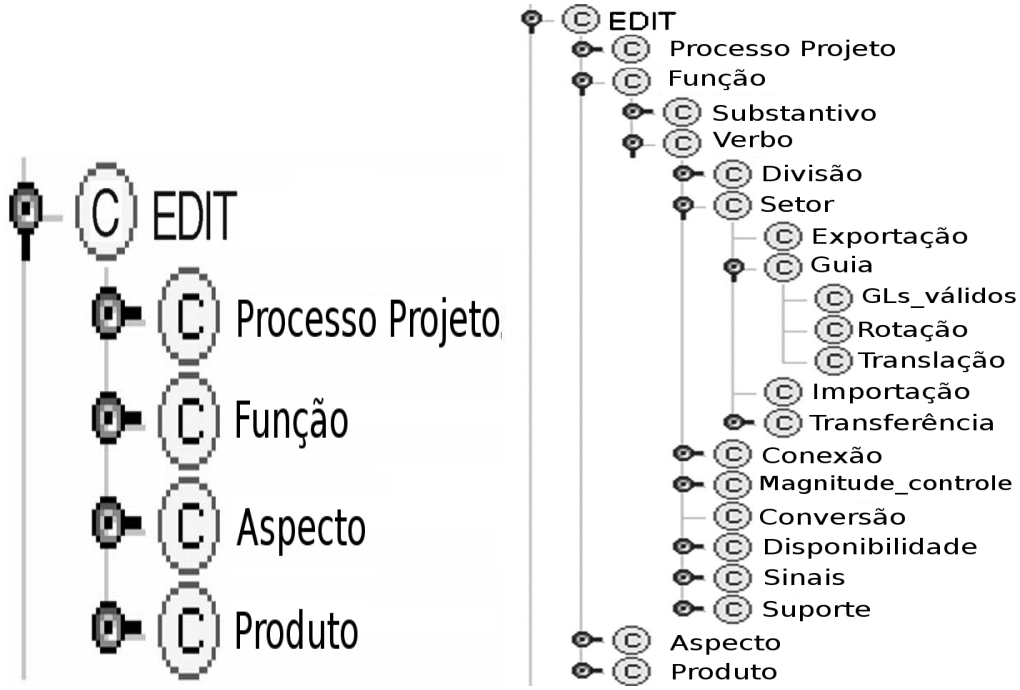


Figura 3.25 – Metodologia de desenvolvimento integrado de taxonomia de ontologias de projeto. [Ahmed, Kim e Wallace, 2007].



(a) Conceitos–raiz da taxonomia de projeto.

(b) Taxonomia da função em projeto.

Figura 3.26 – Taxonomia da atividade de projeto [Ahmed, Kim e Wallace, 2007].

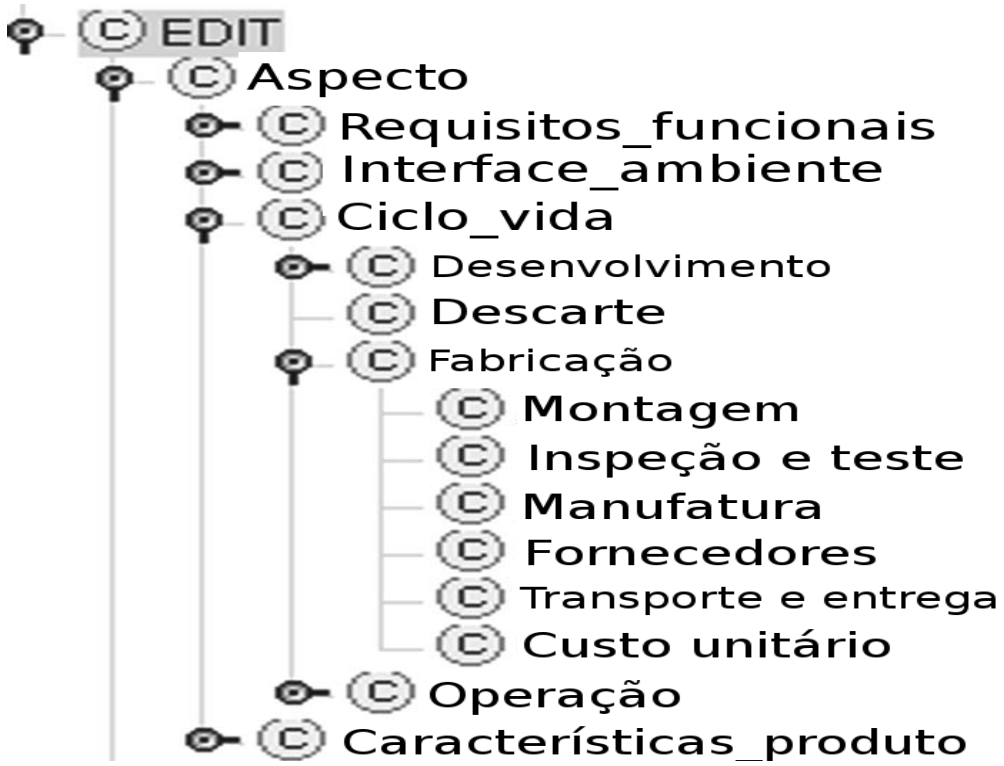


Figura 3.27 – Taxonomia do ciclo de vida integrada com manufatura em projeto [Ahmed, Kim e Wallace, 2007].

A taxonomia da ontologia EDIT possui classes que foram utilizadas analogamente no domínio e escopo desta pesquisa, particularmente relacionadas com algumas subclasses

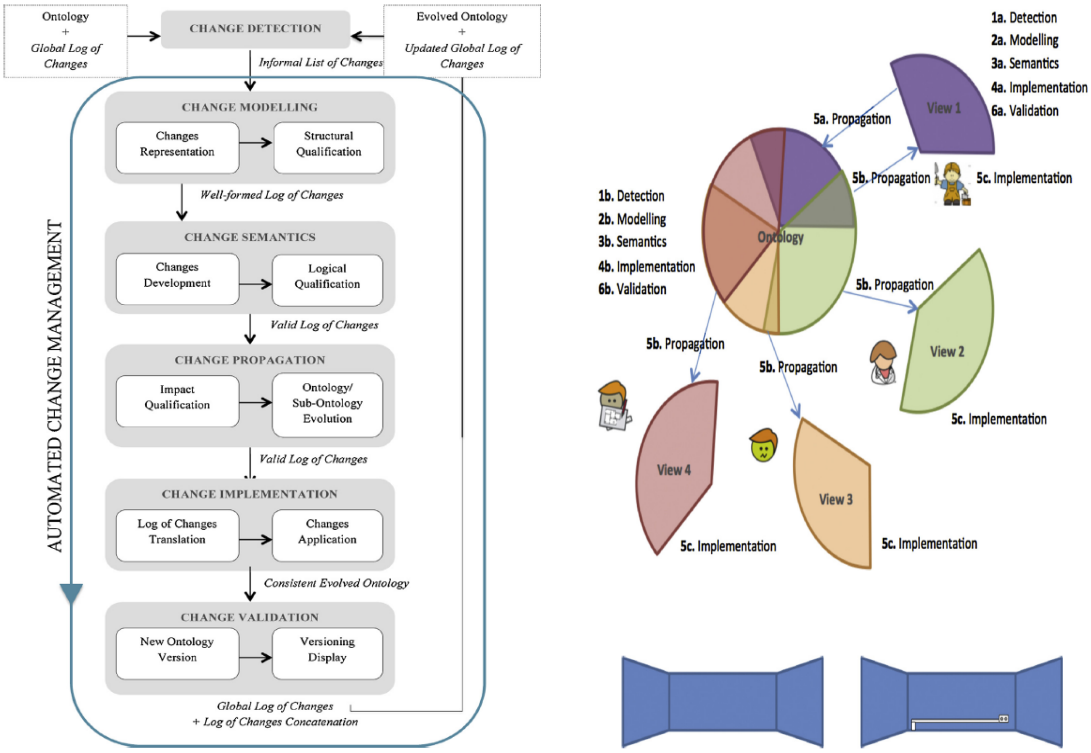
de suas classes-raiz, descritas acima, como Aspecto (que inclui montagem, manufatura, operação e características de fabricação do produto) e Produto (características internas, ou intrínsecas como geometria, do produto). Entretanto, a pesquisa tem domínio, escopo e foco distintos deste trabalho.

3.3 Estrutura e Representação Computacional de Informações Contextuais de Projeto

Historicamente, a eficaz e efetiva utilização da informação demandou considerar sua evolução sob um contexto dinâmico. Este aspecto, que é intrínseco, particularmente em projeto, foi abordado e analisado no gerenciamento de projeto de instalações industriais [Pittet, Cruz e Nicolle, 2014]. Os autores desenvolveram uma "métrica" denominada BIM com uma ferramenta embutida, *OntoVersionGraph*, visando a identificação de inconsistência ontológica através da semântica. Em geral, o objetivo de uma ontologia é conceitualizar, representar e compartilhar a informação e através disto enriquecê-la. Este objetivo é fundamental na área de projeto e, dentro do domínio da pesquisa reportada, significativamente melhorado através de versionamento.

O versionamento dinâmico, portanto, permite controlar e gerenciar políticas de acesso, perfis, contextos e segurança de dados para os usuários. A proposta desta pesquisa foi testada com sucesso em um simples problema real enfrentado por um arquiteto: a adição de um espaço vazio numa parede para colocação de uma porta. Através das taxonomias hierárquica e operacional é possível verificar dinamicamente os efeitos de alterações (log) na configuração de um colaborador (ator) associado (eletricista) através de, por exemplo: $\log_2 = ((\text{deleteRoleInstantiation}, (\text{containsPartOf}, \text{wallspace1}, \text{electricalConduit1}))$. As subfiguras 3.28a, 3.28b da figura 3.28 (página 75) descrevem mais claramente o algoritmo e visões ontológicas.

Carbonera *et al*., 2013 desenvolveram uma forma de redução da ambiguidade de posicionamento, ou localização, de um robô, aspecto fundamental, baseada em ontologias. A ontologia é parte de um projeto mais amplo de uma ontologia-base (*Core Ontology*) de Robótica e Automação [Schlenoff *et al.*, 2012, Prestes *et al.*, 2013], conforme expressa a seguir. Inicialmente, é relevante clarificar conceitualmente o que perfaz a localização de um robô, pois ela, na verdade, é constituída de três partes: topologia, posição, orientação e posicionamento (ou *pose*). Portanto, a localização de um robô constitui-se de aspectos quantitativos e qualitativos. O foco desta pesquisa é relativo a aspectos qualitativos da localização de um robô, ou seja, onde ele está no espaço pois é o que permitirá o planejamento da sua trajetória e movimento. Por exemplo, embora a localização de um robô possa ser expressa por um tríade de coordenadas (x, y, z) de um ponto no espaço em



(a) *Schema* de gerenciamento de modificações ontológicas.

(b) Visões diferentes dos atores de um projeto (arquiteto à esquerda, eletricista à direita).

Figura 3.28 – Gerenciamento de alterações ontológicas [Pittet, Cruz e Nicolle, 2014].

referência a um determinado sistema de coordenadas é igualmente relevante descrever sua localização logicamente: o robô *está na frente* de um obstáculo *C*.

A definição lógico-contextual de um robô é baseada na ontologia CORA [Prestes *et al.*, 2013]. Como um robô é um dispositivo eletromecânico que se move no tempo, existe a necessidade de representar esta característica, que é definida como uma medida, ou observação, q associada com um objeto físico o[10].

$$\forall o \, \forall q \, \mathrm{o,q} \rightarrow \mathsf{Object}(o) \wedge \mathrm{PMeasure}(q) \tag{3.14}$$

Os componentes conceituais lógico-geométricos mais típicos associados com robôs são o sistema de coordenadas, ponto, transformação e mapeamento. Um sistema de coordenadas c é uma entidade definida em relação a um objeto único de referência o, logo $\mathsf{ref}(c, o)$ é verdadeira se existir um e somente um o em c ($\exists!$) (3.15). Um ponto é sempre definido em relação a um sistema de coordenadas (CS), logo, por exemplo, um predicado $\mathsf{in}(x, y)$ é verdadeiro somente se x for um ponto no sistema de coordenadas y (3.16).

[10] A definição 3.14 é síncrona, ou seja, não é possível que existam dois objetos no mesmo ponto.

$$\forall c\, \mathrm{CS}(c) \rightarrow \exists!\, o\, \mathsf{Object}(o) \wedge \mathsf{ref}(c, o) \tag{3.15}$$

$$\forall p\, \mathsf{PPoint}(p) \rightarrow \exists!\, c\, \mathsf{CS} \wedge \mathsf{in}(p.c) \tag{3.16}$$

O mapeamento é uma operação fundamental em Robótica, pois permite representar um determinado ponto em outro sistema de coordenadas através de uma transformação, ou mapeamento, T. Se um predicado $\mathsf{maps}_{\mathsf{CS}}(c, c_r, m)$ for o mapeamento do sistema de coordenadas c em outro sistema de coordenadas c_r através de uma transformação m e o predicado (p_1, p_2, m) denotar o mapeamento de p_1 em p_2 através do mapeamento m é possível descrevê-lo por,

$$\forall c, c_r\, \forall m\, \mathsf{maps}_{\mathsf{CS}}(c, c_r, m) \rightarrow \forall p_1[\mathsf{in}(p_1, c) \rightarrow \exists!\, p_2\, [\mathsf{in}(p_2, c_r) \wedge \mathsf{maps}_{\circ}(p_1, p_2, m)]] \tag{3.17}$$

O posicionamento qualitativo de um objeto é dado pelas regiões posicionais com as quais o objeto tem interface, ou *sobrepõe-se*. Seja ext uma função mapeando um objeto o para uma região correspondente a sua extensão espacial (por exemplo, o volume ocupado pelo objeto) e pos o posicionamento, logo o posicionamento qualitativo é dado por,

$$\forall o\, \forall s\, \mathsf{Object}(o) \wedge \mathsf{PRegion}(s) \wedge \mathsf{pos}(o, s) \rightarrow \mathsf{overlaps}(\mathsf{ext}(o), s) \tag{3.18}$$

Analogamente, é possível definir o posicionamento qualitativo como relações entre objetos. Estas relações são abstrações entre classes de regiões de posições definidas por um operador. Por exemplo, $\mathsf{leftOf}(o, o_r)$ entre os objetos o e o_r descreve um operador definindo a esquerda de um objeto de referência pode ser expressa como ($\mathsf{pos}_{\mathsf{rel}}$ é o posicionamento relativo e generated a região gerada por um dado operador),

$$\forall o\, \forall o_r\, \mathsf{leftOf}(o, o_r) \rightarrow \exists s \mathsf{pos}_{\mathsf{rel}}(o, s, o_r) \wedge \mathsf{generated}(o, o_r, \mathsf{leftOfOp}) \tag{3.19}$$

A *pose* de um objeto é dada por sua posição e orientação intrínseca, conforme expresso em 3.20. Posteriormente, planeja-se, segundo os autores, a inclusão de diversos aspectos topológicos na ontologia como, por exemplo, o tempo. Esta ontologia será integrada com a ontologia mais genérica de Robótica e Automação, brevemente descrita a seguir.

$$\forall o\, \forall e\, \mathsf{pose}(o, e) \rightarrow \exists x\, \exists y\, \mathsf{Pose}(e) \wedge \mathsf{PMeasure}(x) \wedge \mathsf{hasPosition}(e, x) \wedge \mathsf{OMeasure}(y) \wedge \mathsf{hasOrientation}(e, y) \tag{3.20}$$

Na constante evolução tecnológica da área de Robótica e Automação, seja em *hardware*, controle ou *software*, gradativamente tiveram estas capacidades, embora importantes, obviamente, sob o ponto de vista operacional de um manipulador, e sua relevância aproximada com a intrínseca e requerida interação homem-robô (HRI). Um requisito importante, conforme descrito acima, dentre vários outros, é o conceito de localização espacial de um robô para que ele faça, efetivamente, suas tarefas. Portanto, uma forma comum de compreensão da informação utilizada torna-se relevante.

Jorge *et al*., 2014 utilizaram ontologias para descrição e utilização do aspecto descrito acima, associado com agentes de software, em um estudo de cooperação colaborativa entre dois robôs móveis para entrega de um objeto a um ser humano, o qual demanda especialização de conceitos genéricos. A pesquisa utiliza as ontologias preexistentes CORA, genérica de Robótica e Automação, e POS [Carbonera *et al*., 2013], de posicionamento em Robótica, que está todavia em desenvolvimento, especificamente, na parte de agentes pois são os responsáveis pelo envio e recebimento de mensagens contextuais. A experiência provou, com sucesso, a eficácia da utilização de ontologias para movimentação conjunta e colaborativa dos robôs e usuário com consequente solução do problema.

Considerando-se que, como em qualquer área relacionada com contextos, particularmente no que tange a informações, a qualidade de uma proposta solução precisa ser quantificada (qualificada); Rico *et al*., 2014 desenvolveram um arcabouço para julgamento da qualidade de ontologias (ou seja, a validade de seu propósito). Neste aspecto, torna-se importante ressaltar que analisar a qualidade de uma ontologia desconsiderando sua utilização, ou propósito, não tem sentido prático. Os autores desenvolveram uma ontologia para representar a qualidade da informação intercambiada entre sistemas de informação heterogêneos através da sua semântica. A análise da qualidade das ontologias é feita, portanto, semanticamente utilizando três tipos de requisitos (OR):

OR1 representação formal da informação intercambiada.

OR2 representação da informação estritamente necessária ao intercâmbio.

OR3 interpretação correta da informação trocada em todos contextos em análise.

Cada OR é analisado através de perguntas específicas, por exemplo em OR1 duas perguntas são: Q1.1 A linguagem é computacionalmente representável? e Q1.2 A linguagem permite a representação de entidades, seus relacionamentos e características? Cada pergunta possui valores-resposta que podem ser binários ou um intervalo válido (tipicamente 0 ou 1 e [0,1]) e cada valor-resposta possui um ótimo (em geral um). Claramente, OR3 perfaz a parte mais complexa desta ontologia, logo cinco princípios básicos foram utilizados no seu projeto. Estes princípios, que são focados nas similaridades entre entidades, relações e características, tem por proposta enriquecer a informação intercambiada [Ricoa *et al*., 2013].

A ontologia EBDO foi qualificada e testada em um ambiente EBD real de uma indústria de laticínios. Especificamente, foi testado se a qualidade da ontologia influencia os resultados do intercâmbio de informações entre a fábrica e um fornecedor de leite numa comparação entre três ontologias: $EBDO_{base}$, $EBDO_{enriched}$ e $EBDO_{dairy}$; a última ontologia é a referência. A comparação foi feita através da ferramenta AgentMaker [Cruz, Stroe e Palmonari, 2012], que comprovou que a hipótese do estudo, qual seja, a qualidade de uma ontologia é influenciada pela(s) ontologia(s) utilizada como *input*.

Borgo, 2014 utilizou ontologias relacionando, sob contexto linguístico e de polissemia, as diferenças entre a compreensão de um contexto na área industrial e sua descrição ontológica. Seja a ontologia O descrita por $\leq, C_1, \ldots, C_n, R_1^{S_1}, \ldots, R_m^{S_m}$ baseada num vocabulário onde $\leq$ denota a relação de pertencimento (*subsumption*), $C_1, \ldots, C_n$ são os predicados das categorias (classes) e $R_j^{S_j}$ os relacionamentos n-ários de s_j (propriedades, relacionamentos). De forma a restringir a utilização do vocabulário são utilizadas duas definições em lógica de primeira ordem[11]:

$$\text{Se} \quad C_h \leq C_k \text{ então } \forall x(C_h(x) \rightarrow C_k(x)) \tag{3.21}$$

$$\forall x \left(\bigvee_i C_i(x)\right) \tag{3.22}$$

O autor descreve como gerenciar a diferença na compreensão dos conceitos intrínsecos de uma ontologia O, no caso PCBA, e conceitos extrínsecos como, por exemplo, desenhos esquemáticos, denominados de entidades *extra-ontológicas*. Analogamente ao descrito, se existir uma categoria extra-ontológica D relacionada com O é possível utilizar uma relação "remendo" (*patch*) tal que, para $k \geq 2$:

$$\forall x \, \exists y_1, \ldots, y_k \left(x \in D \rightarrow \bigwedge_i C_{j_i}(y_i) \wedge Patch(x, y_1, \ldots, y_k) \right) \tag{3.23}$$

Portanto, a relação "remendo" define uma relação entre um domínio D e duas, ou mais, categorias C_{j_i} tal que, para cada elemento x de D, existem elementos y_i de C_{j_i} para os quais ela existe com $x, y_1, \ldots, y_k$. Consequentemente, O torna-se "remendada" por D ou D e C_{j_i} para cada i, logo, estão "remendadas". Adicionalmente, é relevante destacar que o "remendo" não inclui a categoria D na ontologia O, pois existe apenas a quantificação do domínio misto/conjunto O, D.

A abordagem de integração entre categorias de uma ontologia e categorias *extra-ontológicas* foi testada em dois exemplos: integração do ciclo de vida de dados de plantas industriais, particularmente em produção de gás e óleo combustível [ISO, 2003], e DOLCE [Borgo e Masolo, 2009]. A abordagem atual, que utiliza ISO15926-2, embora

[11] (3.21) denota a relação de pertencimento, (3.22) denota que categorias são ou uma especialização (conceito genérico) ou disjuntas.

forneça uma solução, é baseada em busca exaustiva e para problemas de âmbito real, onde o número de entidades cresce vertiginosamente, ela tornar-se-á proibitiva. DOLCE é uma ontologia iniciada em 2002 para domínios como modelagem de organizações e produtos, medicina. Esta ontologia apresenta limitações, fundamentalmente na divisão entre entidades físicas e de informação. O ambiente industrial é pródigo em entidades que pertencem a mais de um domínio, como no exemplo de PCBA. Esta abordagem integradora possibilita simplificar a união de conceitos logicamente incompletos ou de integração aparentemente inviável. Suas principais vantagens podem ser enumeradas,

- Sistemas "remendados" através deste tipo de relação entre entidades ontológicas e *extra-ontológicas* são aquiescentes, logicamente consistentes e coerentes com a visão organizacional e processual de uma empresa.
- Os dados da empresa permanecem, todos, acessíveis.
- As estratégia da empresa não é modificada na integração.

Hiekata e Yamato, 2009 desenvolveram um modelo de armazenamento de conhecimento DFM baseado em ontologias no domínio de projeto de componentes de navios de docas. O modelo permite que o plano de processo de produção seja acessado e alterado pelos responsáveis facilitando, consequentemente, a compreensão da fabricação de um produto e seu processo. A descrição do conhecimento é definida em RDF com todos os objetos possuindo identificadores únicos de recursos (URI)s. Um método de integração entre formas diferentes de descrição de sistemas para DFM, especificamente linguagem de descrição de ontologias RDF e linguagem de descrição formal de processos IDEF0, foi proposto.

O conhecimento dos processos é estruturado a partir de classes representam as instâncias dos componentes reais. As classes são estruturadas de forma análoga às tradicionais listas de materiais (BOM). Estas classes são utilizadas na geração do plano de processo, que é focado em operações de montagem, neste caso especificamente no tempo das operações. Juntamente com os tempos das operações, o sistema fornece o caminho crítico de montagem. A taxonomia, consequentemente o escopo no domínio descrito acima, é focada em operações de montagem diferindo, portanto, da pesquisa deste trabalho.

Grüninger, 2007, Grüninger e Menzel, 2003 desenvolveram uma linguagem–base para representação de processos denominada Linguagem de Especificação de Processo (PSL). O objetivo desta linguagem é a interoperabilidade entre componentes sistêmicos de determinado domínio. Especificamente, o foco inicial da pesquisa foram sistemas de manufatura.

PSL é definida baseada em ontologias através da utilização axiomas; todos axiomas são sentenças de primeira ordem representadas em KIF. Os axiomas são subdivididos

em dois tipos: teoremas básicos (*Core theories*) e extensões das definições. Os teoremas básicos definem e axiomatizam relações e funções e são as primitivas da linguagem, sendo utilizadas pelas extensões quando novos axiomas existem em um determinado domínio. A linguagem PSL permite definir a semântica intuitiva das primitivas de manufatura. A ontologia–base (*core ontology*) possui quatro classes disjuntas: atividades, ocorrências de atividades, pontos temporais e objetos, conforme a figura 3.29.

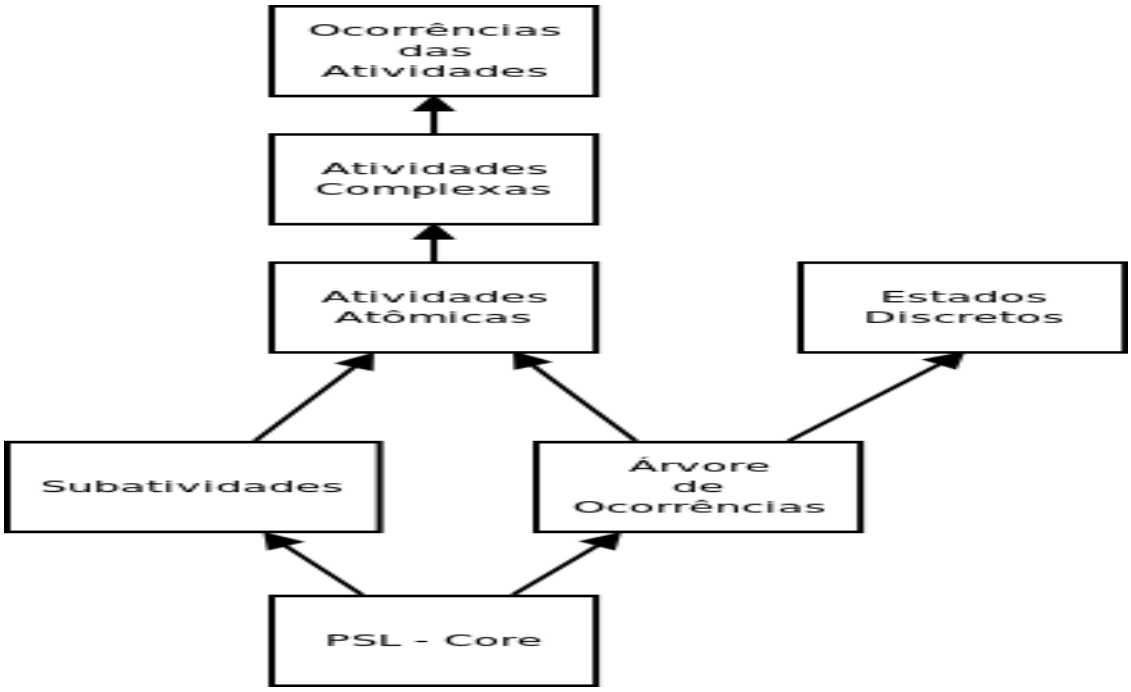


Figura 3.29 – Teoremas básicos da ontologia PSL.

As árvores de ocorrências são, em resumo, axiomas de conjuntos de sequências discretas de ocorrências de atividades; o fator que diferencia uma árvore de outra é a situação, ou estado, inicial. Estados discretos descrevem estados específicos (*fluents*) e somente podem ser modificados através de uma ação (ou atividade). Os axiomas de sub-atividades são relações isomórficas em relação a uma determinada ordenação parcial discreta. As atividades atômicas são axiomas de agregação de atividades primitivas.

Uma agregação simultânea, ou concorrente, é representada através da ocorrência de atividades simultâneas. Os axiomas de atividades complexas são representados pelas intra-relações entre as ocorrências de uma atividade e as ocorrências de suas sub-atividades. Os axiomas definidos através de relações complexas restringem, ou limitam, as ocorrências de sub-atividades. Os axiomas de ocorrência de atividades generalizam estas restrições em relação a sub-atividades complexas.

A ontologia PSL define axiomas para classes e atividades utilizados na definição de um processo. Entretanto, a utilização desta ontologia não impõe de forma restritiva a troca da definição de classes de atividades entre diferente componentes de *software*. Ao invés disto, é feita a troca de sentenças que pertençam a suas classes. Estas sentenças representam especificações de processos, em geral, ou conceitos de processos.

Em geral, a dinamicidade de um processo de manufatura, conforme seu foco, torna complexa sua utilização numa proposta, inicialmente, focada na taxonomia hierárquica e operacional em DFM. Eventualmente, inclusive conforme o objetivo da utilização de ontologias, tanto PSL quanto esta ontologia DFM poderão ser integradas.

Schlenoff, Ivester e Knutilla, 2002 identificaram que a compreensão e utilização adequada da informação é o limitador relevante na integração de sistemas de manufatura. Os autores descrevem que esta limitação deve–se à falta de semântica e sintaxe adequadas na interface entre as aplicações de um domínio, neste caso processos de manufatura, e a arquitetura–padrão atual STEP.

Zhao e Liu, 2008a, Zhao e Liu, 2008b desenvolveram uma forma de interpretar o modelo de informação STEP através de ontologias. Neste caso, especificamente, o objetivo é representar o contexto da informação através da semântica existente. Os autores reiteram que a semântica formal limitada da arquitetura STEP através de sua linguagem EXPRESS limita a utilização da informação de uma produto específico; por exemplo enquanto é possível a troca de dados como tradicionalmente é feito não existe de fato extensibilidade na representação da informação.

A representação do conhecimento através da rigidez arquitetônica STEP existe para alguns tipos de dados, entidades, etc. porém suas restrições tipicamente definidas por regras e conhecimento procedural (p.ex. funções e processos) não são utilizadas eficientemente. Desta forma, ontologias são utilizadas operacionalmente em um modelo STEP como forma de fazer inferências sobre informações de projeto.

A abordagem dos autores utiliza a fundação representacional ontológica: a linguagem XML que codifica a arquitetura STEP representada por EXPRESS em OWL e SWRL (conforme a figura 3.30). Especificamente, a linguagem OWL–DL, que é um subconjunto de da linguagem OWL, é utilizada garantindo que a solução seja computacionalmente completa e com decidibilidade.

A utilização de SWRL baseado nesta sub-linguagem permite que a parte "dedutiva" do conhecimento possa ser utilizada. A parte "dedutiva" do conhecimento tipicamente tem anexada uma cláusula do tipo WHERE nas regras que define a validade de uma expressão. O mecanismo de herança em STEP é mapeado analogamente a axiomas existentes em declarações nas classes OWL.

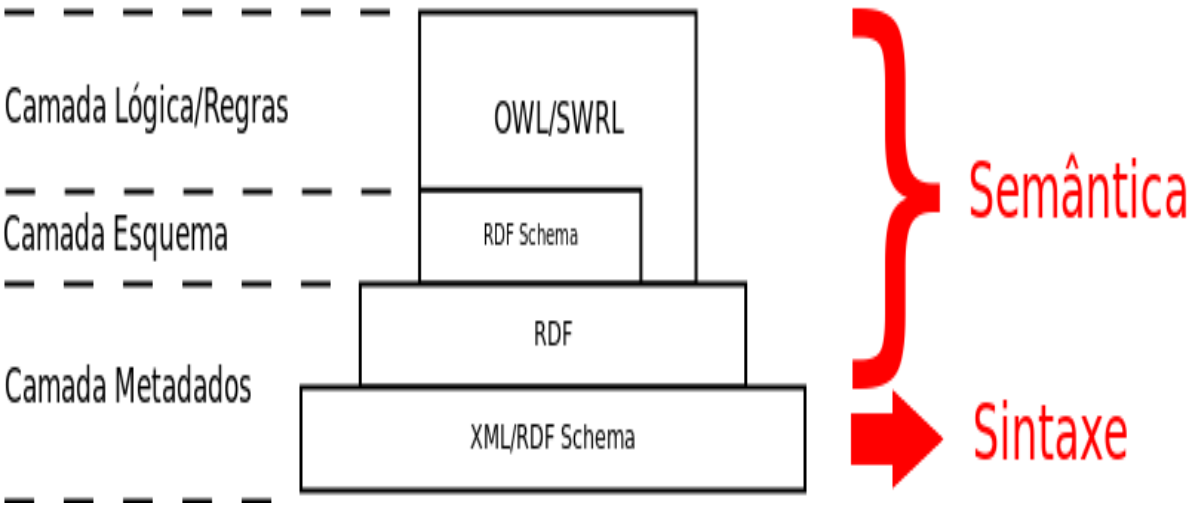


Figura 3.30 – Arquitetura OWL/SWRL sobre Web Semântica [Zhao e Liu, 2008a].

O esquema (*schema*) STEP é utilizado como um recipiente (*container*) da linguagem EXPRESS que descreve modelos de produtos. Modelos de produtos são conceitos–padrões independentes e integrados. Na pesquisa em questão, a ontologia corresponde a um *schema* STEP com o mesmo nome. Esta característica permite a união bem como adaptação com análise de adequabilidade entre esquemas distintos.

Através da base teórica descrita, os autores concomitantemente implementaram a parte prática desta abordagem como teste [Zhao e Liu, 2008b]. O foco desta parte da pesquisa é composto de dois aspectos: definição das classes, suas propriedades e instâncias conforme descrito em [Zhao e Liu, 2008a] utilizando OWL e a reinterpretação da tradução da representação ontológica EXPRESS para habilitar inferência e raciocínio utilizando regras SWRL.

A segunda parte permite validar a consistência das definições representadas utilizando o sistema especialista Jess [Friedman-Hill, 2003]. A tradução entre a semântica OWL e SWRL utiliza componentes de integração entre as diferentes formas de representação: OWL2Jess e SWRL2Jess [Mei, Bontas e Lin, 2005] utilizando *eXtensible Stylesheet Language Transformation* [World Wide Web Consortium, 1999]. A arquitetura da pesquisa, portanto, pode ser subdividida em três partes, que são:

1. Tradução do esquema e dados STEP e sua implementação em OWL e SWRL.

2. Definição da interoperação entre o modelo de informação do produto e bases de conhecimento representadas por regras.

3. Disponibilização de interfaces para aplicações que utilizem e/ou necessitam de um modelo de informação do produto.

Tabela 3.5 – Mapeamento entre EXPRESS e OWL.

EXPRESS	OWL
Esquema (Schema)	Ontologia (Ontology)
Entidade (Entity)	Classe (Class)
Subtipo de (Subtype of)	Subclasse de (Subclass of)
Atributo com tipo da entidade	PropriedadeObjeto (ObjectProperty)
Atributo com tipo de dado simples	PropriedadeDado (DataProperty)
Atributo opcional	O intervalo permissível da propriedade é restrito ao valor–membro ObjectAllValuesFrom ser igual à união entre o tipo de atributo e a classe Nothing

Os resultados desta abordagem permitiram a integração de aplicações heterogêneas com um modelo de informação contextualizada do produto representado por ontologias integradas com Jess. Entretanto existem algumas limitações como, por exemplo, a utilização de tipos de arquivos diferentes na tradução entre OWL/SWRL e Jess. Adicionalmente, Jess é um sistema especialista baseado em *forward chaining* e nem sempre esta é a forma de raciocínio utilizada em alguns aspectos de informações de produto. Um tópico a ser desenvolvido futuramente é a integração entre Jess e estratégia evento–condição–ações (*event–condition–actions* – ECA) que existe no *software* Protégé.

Krima *et al*., 2009a, Krima *et al*., 2009b descreveram como estruturar a abordagem e as hipóteses utilizadas na integração entre STEP e OWL através de seu subconjunto OWL–DL [12]. A utilização deste subconjunto de Lógica Descritiva (DL) simplifica a integração pois possui facilitadores semânticos: verificação da consistência da informação, inferência e decidibilidade. A pesquisa estrutura a interpretação semântica de um arquivo STEP em duas etapas: mapeamento dos conceitos básicos principais e mapeamento das instâncias conforme a tabela 3.5.

Barbau *et al*., 2012 desenvolveram um componente adicional (*plug-in*) ontológico para enriquecer modelos de dados de produtos, tradicionalmente representados no formato STEP, denominado OntoSTEP. A linguagem EXPRESS é fundamentada em entidades e atributos que não tem capacidade de representar a intrínseca semântica e contexto existentes em sistemas PLM. O *plug-in*, devido a ser baseado em semântica, permite a representação de conceitos geométricos e não geométricos tais como funções, comportamento, estrutura (ou forma) de projeto, etc.

A STEP–AP representada por EXPRESS em um arquivo STEP é traduzida para um esquema (*schema*) OWL formando um componente terminológico (*terminological box*) T-Box. Os dados extraídos de um arquivo STEP parte 21 obtido de um sistema CAD são traduzidos em indivíduos OWL formando um componente assertivo (*assertional box*) A-Box.

[12] *Uma das três sublinguagens OWL que garante que a semântica seja computacionalmente completa e com decidibilidade além de permitir processos de dedução da informação.*

Posteriormente, ambas representações são combinadas em uma ontologia resultante que é utilizada no Modelo Básico de Produto NIST (*Core Product Model*) [Fenves *et al*., 2008] e que também possui possibilidade de representação de operações de montagem [Baysal *et al*., 2004].

Embora o *plug-in* seja um aspecto relevante para permitir uma utilização mais completa e efetiva da informação em sistemas de manufatura, ainda não foi possível obter uma solução completa baseada em STEP incluindo todas características da linguagem EXPRESS devido a limitações da representação OWL, bem como complexidade crescente.

As pesquisas de Zhao, inicialmente, Barbau e Krima, posteriormente, foram inovadoras pois possibilitaram abordagens para o enriquecimento efetivo da informação definida de forma inequívoca e explícita em arquivos STEP num arcabouço moderno focado em informações, aspecto cada vez mais relevante atualmente. Ainda assim, ambas pesquisas possuem utilização limitada atualmente, no caso de Barbau ou Krima, ou inviável, no caso de Zhao e Liu, quando relacionadas com esta pesquisa. Enquanto OntoSTEP proporciona uma ligação efetiva e, sob certo aspecto, eficiente para mapear STEP em um arcabouço de informação, sua taxonomia, obviamente, precisa ser totalmente integrada com esta pesquisa para que seja útil. Portanto, sua perspectiva utilização eventual nesta pesquisa seria *a posteriori*, inclusive devido ao fato dela ser implementacionalmente baseada numa Java API ao invés do padrão RDF (ou SQWRL). Embora seja útil sob contexto de como definir EXPRESS em uma ontologia, a pesquisa de Zhao e Liu foca, conforme reportada, somente ao análogo da parte terminológica T–Box em STEP.

Kulon, Broomhead e Mynors, 2006 expressarem e ressaltaram o quão relevante é a utilização do conhecimento em DFM/DFA, especificamente no processo forjamento a quente integrado com a Internet. Um sistema integrado utilizando conhecimento foi desenvolvido em diversos módulos em um núcleo. Os módulos podem ser subdivididos, em geral, em 4 áreas: banco de dados (armazena regras de projeto, regras de produção, dados de materiais, dados de aplicações, dados de projetos pré–existentes e cliente de acesso ao banco de dados), núcleo (*kernel*) geométrico, projeto e análise (CAD, FEA) e Internet (interface com o usuário.).

Finalizando este capítulo, foram analisados os principais tópicos relativos à informação relacionados com esta pesquisa. Quando considerado relevante e associado diretamente com ela, uma breve análise crítica foi feita. A descrição completa de todos os aspectos relacionados com informação está fora do escopo deste trabalho, embora a bibliografia propicie referências consideradas adequadas para a sua compreensão.

4 ARCABOUÇO DE INFORMAÇÕES E CONHECIMENTO PARA INTEGRAÇÃO DE COMPONENTES DE PROJETO ORIENTADO PARA MANUFATURA (DFM)

Este capítulo descreve a pesquisa deste trabalho. Os principais aspectos relacionados com arquiteturas de integração e requisitos de informação são apresentados sob contexto de Projeto Orientado para Manufatura (DFM). A primeira seção define aspectos teóricos sobre arquiteturas (*frameworks*) de projeto com foco na estrutura de informação necessária. A estrutura proposta permite inferir analogias com os aspectos arquitetônicos de integração e estrutura de informação descritos no capítulo de Revisão da Literatura, capítulo 3. Especificamente, são descritos sistemicamente os pré–requisitos que precisam ser considerados e analisados para uma arquitetura integrada DFM e que, obviamente, também devem ser incluídos no que se refere à informação; são eles: arquitetura (seção 4.1), modelo cognitivo (seção 4.1.1) e funcionalidade (seção 4.2). Após estas considerações iniciais, o arcabouço de informação e conhecimento para DFM é descrito.

Na estruturação da informação, nesta pesquisa definida como um subconjunto do conhecimento, é necessário definir inicialmente o domínio analisado. Nesta pesquisa o domínio utilizado é Projeto Orientado à Manufatura (*Design For Manufacturing* – DFM). Objetivamente, a pesquisa tem por foco a estrutura e arquitetura necessárias às informações utilizadas por seus componentes utilizando um subconjunto de aplicações do domínio. Dentro deste contexto e considerando que DFM é um domínio muito amplo e abrangente, a especificidade da estrutura da informação é restrita a peças poliédricas sujeitas a processos de usinagem, sem perda de generalidade. Adicionalmente, uma arquitetura–protótipo de informação é desenvolvida com subconjuntos de processos de usinagem de materiais metálicos e características geométricas (*features*) para demonstrar a sua viabilidade.

Historicamente, devido a limitações relacionadas ao ferramental teórico–prático, *hardware* e *software*, originalmente a arquitetura e estrutura de informação para ambientes integrados ficou restrita a dados somente. Este aspecto é restritivo inclusive sob o ponto de vista combinatório. Por exemplo, se for considerado um ambiente integrado composto por cinco componentes, como nesta pesquisa, o conjunto de possibilidades de funções de conversão direta de dados, *apenas dados*, entre componentes (importar, exportar) é dado por $2^n = 2^5 = 32$ (no caso de existir a necessidade de intercâmbio direto). Isto demonstra claramente a necessidade de desenvolvimento de uma forma de representação comum; aspecto que já foi definido na arquitetura STEP. A representação comum, historicamente, foi focada em dados com posterior augmentação para uma arquitetura. Entretanto, conforme descrito a seguir neste capítulo, somente dados não representam a informação de forma completa, ou íntegra.

4.1 Arquitetura DFM de Informação

Genericamente, uma arquitetura pode ser descrita como uma "*estrutura de suporte ou que engloba algo*". Ela perfaz o mais alto nível de abstração de um sistema, neste caso DFM. Componentes, ou auxiliares, de projeto são responsáveis pelas atividades na arquitetura. As atividades DFM podem variar pois dependem tanto da peça quanto do aspecto de manufatura em análise. Esta variabilidade intensifica a necessidade de uma correta e adequada estrutura da informação pois a combinatoriedade entre dependências cresce exponencialmente se forem considerados somente dados, conforme descrito, além de não descrever completamente seu significado real.

Recentemente, Hoque *et al.*, 2013 demonstraram uma arquitetura CAD/CAM com foco em DFM; o sistema proposto integra características (*features*) de fabricação utilizando uma biblioteca de *features* que tem por objetivo unir geometria com manufatura. Neste caso, a solução proposta é desenvolvida utilizando uma plataforma comercial, CREO da PTC, através de orientação a objetos. Embora seja uma solução efetiva e eficiente para o domínio inicialmente proposto, ela possui elevado custo, monetário e computacional. Talvez a principal limitação desta solução em termos de flexibilidade de integração com novos componentes seja a falta de uma forma mais eficaz de gerenciar a informação que cada parte sistêmica demanda, pois ela é díspar visto que é baseada tanto em representação direta, fundamentada em dados através da utilização de um banco de dados, quanto contextualmente: seu escopo é em operações de torneamento. A limitação deve-se ao fato do dado ser o foco da representação da informação, conforme pode ser visto na figura 4.31, que mostra claramente que existe uma intersecção não clara e explícita quando aspectos heterogêneos precisam ser integrados.

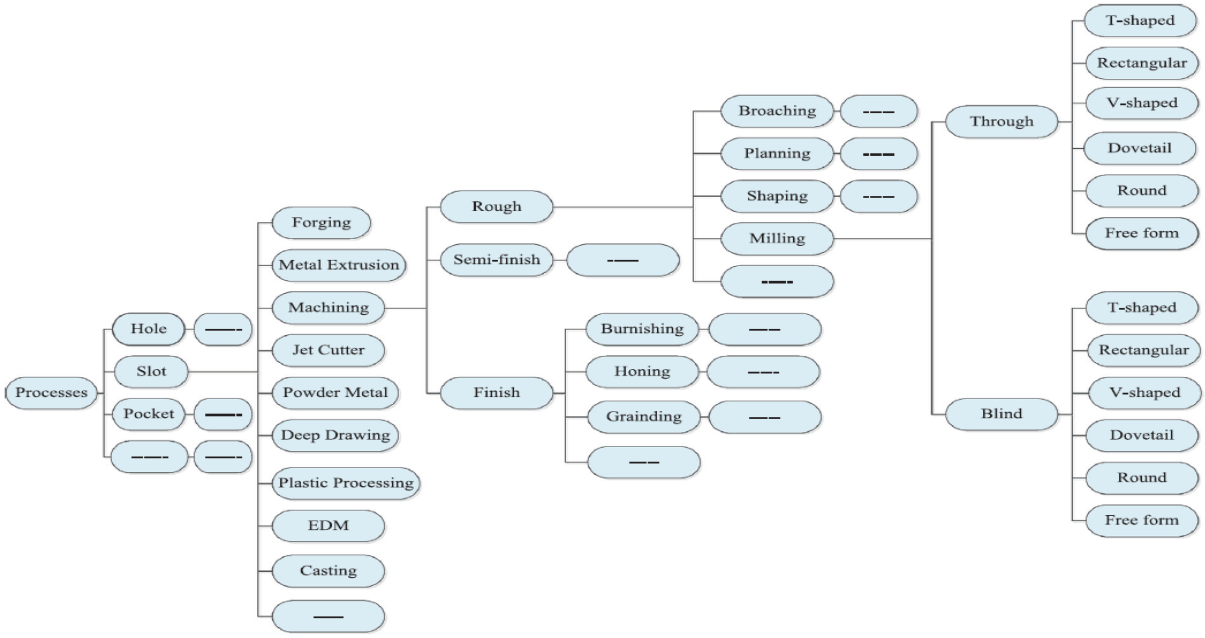


Figura 4.31 – Hierarquia combinada da informação entre processos e geometria [Hoque *et al.*, 2013].

Por princípio, portanto, é relevante destacar que somente dados não especificam a totalidade da informação necessária de forma adequada, ou seja, não é conceitualizada a partir de sua taxonomia. Ela pode variar desde, por exemplo, dados em uma sequência algorítmica fixa [Krishnan, 1995] até sistemas baseados em conhecimento. Adicionalmente, a compreensão do real significado de diferentes aspectos relacionados com a informação possui abstração diametralmente distinta de sua instanciação[1] eventual através de dados. Desta forma, é necessário que exista gerenciamento e estrutura do conhecimento, tradicionalmente formado por somente dados como uma das partes da informação.

Uma forma mais completa disponível para isto é através da utilização de ontologias, descritas na seção 4.3.1. A falta desta interconexão entre dados e processos associados, juntamente com a contextualização da informação, ou sua taxonomia, limita de forma relevante a flexibilidade de sistemas e consequentemente não captura de forma clara o conhecimento que permite resolver um determinado problema. Chang, Sahin e Terpenny, 2008 demonstraram eficácia e utilidade desta técnica no estágio de projeto conceitual de produtos. Neste caso, a arquitetura, que é baseada em M-IDEF∅[2] com uma ferramenta gráfica de modelagem, expressa os inter–relacionamentos entre um banco de dados e ontologias. Através desta integração é possível capturar o conhecimento de projeto através de consultas contextualizadas enriquecidas por um módulo de análise dos dados, que oferece ao projetista resultados semanticamente consistentes com seus requisitos de informação.

A relevância e importância da utilização da informação em ambientes, ou arquiteturas/arcabouços integrados, particularmente na área de projeto, é um aspecto conhecido há bastante tempo. Considerando este contexto, bancos de dados tem sido, em geral, uma das técnicas mais utilizada em arquiteturas de projeto [Shaw e Garlan, 1996, Howard e Lewis, 2003]. Tipicamente, em sistemas do tipo CAD/CAM um banco de dados armazena os dados fundamentais em estruturas tabulares indexadas. Woyak e Myklebust, 1998 descrevem, de forma pioneira, a relevância e importância de bancos de dados definidos de forma adequada, juntamente com a necessária troca da informação, considerando o fluxo de um projeto.

Esta característica deriva do fato que processos de projeto em engenharia são compostos por tarefas, majoritariamente, fortemente criativas e integradas com aspectos relacionados ao conhecimento. Estas atividades, por sua vez, envolvem a troca e integração de informações associadas, direta ou indiretamente [Brandt *et al.*, 2006]. O conhecimento associado com estas informações é de relevância fundamental para a competitividade de uma empresa. Este conhecimento, que é heterogêneo, precisa, adicionalmente, ser integrado em uma arquitetura sistêmica, idealmente. O conhecimento, em geral, é *implícito* e de

[1] Ação ou efeito de instanciar, fornecer a instância concreta de alguma coisa. Em programação, criar uma instância concreta, um objeto de determinada classe.

[2] Modified Integrated DEFinition ∅.

forma que possa ser utilizado de forma efetiva precisa ser tornado *explícito* e idealmente compartilhado entre os partícipes do processo de projeto.

Explicitar e utilizar conhecimento, que é expresso através informações de forma ótima equivale, em geral, a definir sua taxonomia. Uma taxonomia envolve, na verdade, um processo complexo para sua definição, pois existe um conjunto complexo de aspectos a serem levados em consideração. Por exemplo, o quão adequada ela é para descrever uma informação específica, bem como seu compromisso mínimo[3]?

Portanto, conforme descrito a partir do início deste capítulo e como pode ser inferido a partir da revisão da literatura, tanto no domínio quanto especificamente no que tange à informação *per se*, ainda falta uma solução baseada em ontologias no escopo do domínio selecionado. As próximas seções deste capítulo detalham a abordagem considerada adequada na representação de informações para resolução da heterogeneidade presente em projeto, em geral, porém, particular e especialmente, em DFM. O capítulo 5 descreve, por sua vez, como analisar a consistência das definições propostas neste capítulo para taxonomia DFM através de uma máquina de inferência.

4.1.1 Modelo Cognitivo

A definição de um modelo cognitivo de utilização da informação deve ser o primeiro passo para fundamentação teórica de sua arquitetura para utilização em um ambiente integrado, seja ele computacional ou não. O modelo cognitivo estabelece a forma através da qual o usuário define, interage e utiliza a informação. Desta forma, a abordagem proposta através de ontologias, especifica os aspectos previamente descritos de uma forma mais próxima da interpretação humana, ou seja, mais abstrata e sem perda significativa da necessária disponibilidade e interconexão com dados reais. Sob o contexto de integração em uma arquitetura computacional, no caso o domínio DFM, duas formas de interação com o usuário existem: não–assistida e assistida [Megale *et al*., 1991].

No modelo não–assistido o sistema é responsável pela maioria das decisões, ou seja, o sistema efetuará um determinado conjunto de atividades baseado no *input* do usuário. Posteriormente, se o usuário modificar suas opções de projeto definindo um novo conjunto de *inputs*, por exemplo, uma nova solução alternativa é gerada. Sistemas computacionais baseados neste modelo demandam quantidade relevante de conhecimento do domínio em questão e são mais complexos para desenvolver e manter. No modelo não–assistido, o sistema recebe o *input* do projetista e procura obter uma solução inicial; este tipo de sistema é considerado primariamente um modelo síntese–orientado.

O sistema atua como um auxiliar de projeto do projetista no modelo assistido, aconselhando modificações e sugerindo correções dinamicamente baseado nos *inputs*.

[3]O compromisso mínimo ó que define os conceitos representacionais mínimos de uma informação em análise.

Neste modelo o sistema e o usuário atuam de forma única em direção à solução final ideal. O sistema primariamente tem atividades de análise baseadas nos *inputs* do usuário neste contexto. Obviamente, as atividades de análise executadas a partir dos *inputs* dos usuários podem requerer/demandar atividades relacionadas de síntese.

As informações, entretanto, cognitivamente utilizam uma solução mista entre as duas alternativas mais comuns na estrutura proposta, quando totalmente finalizada. A estrutura utiliza, inicialmente, o modelo não–assistido devido à necessidade de obtenção da fonte de informações STEP. Adicionalmente, ela também pode ser utilizada no modelo assistido pois o contexto da informação é alterado dinamicamente, particularmente relacionando o quão específico é o estado corrente das informações.

4.2 Funcionalidade

Conforme descrito, a definição de uma arquitetura de informação para um sistema DFM demanda que suas diversas fontes e formas de representação, que são heterogêneas, sejam não apenas integráveis porém, também, efetivamente passíveis de compreensão, integração e utilização. Embora atualmente existam sistemas PDM/PLM (*Product Data Management/Product Lifecycle Management*) disponíveis, inclusive comercialmente, eles são limitados pois ou não possuem modelos de informação de produto adequados/disponíveis ou, se os possuem, são limitados/determinísticos além de, em geral, são definidos em termos de dados. Este tipo de sistemas tem como foco processos detalhados, ou micro–processos, tais como controle de versões e/ou mudanças na engenharia de um determinado produto.

A forma de aquisição e estruturação da informação dentro de uma arquitetura padronizada e flexível é, atualmente, melhor e mais efetivamente solucionável através de ontologias. Ontologias permitem a definição e utilização de *schemas* dinâmicos de informação, ao contrário dos tradicionais *schemas* fixos.

DFM é, por padrão, uma atividade baseada em conhecimento de forma deveras intensiva pois engloba várias fases de projeto, qual sejam conceitual, detalhado, montagem, processo e sua avaliação. Adicionalmente, particularmente em DFM, uma arquitetura/estrutura de integração possui demandas complexas em relação às diferentes formas de interpretação da informação associada com os requisitos de seus componentes[4]. Adicionalmente, a interpretação distinta da informação leva a problemas relacionados com sua utilização até na sua forma mais definida, que é o dado. Este aspecto é bastante claro através do exemplo de uma típica tabela simples, conforme analogamente já descrito no capítulo introdutório deste trabalho na tabela 1.1, que é relacionada com o domínio de manufatura.

[4]É necessário salientar que, neste parágrafo, apenas uma visão de informação é considerada a título de simplificação: a focada no dado.

As operações e custo descritas naquela tabela, que são expressas somente em função de dados, também possuem informações associadas relevantes, as quais tipicamente são expressas em tabelas–referência obtidas através de cálculos ou de experiência prática. *Estas informações, junto com suas abstrações, é que perfazem o conhecimento necessário para efetivamente resolver um problema geralmente e, particularmente, em DFM*. A informação e conhecimento associados nestas tabelas são *intrínsecos*, ou *embarcados*, nelas. Este aspecto é análogo na maior parte das informações em engenharia. A disjunção entre informações representadas simplesmente por dados e sua origem, bem como sua razão–fonte, torna a compreensão contextual complexa e difícil, limitando a efetividade e eficiência dos sistemas que as utilizam.

Especificamente, no que tange à automatização em domínios integrados, como DFM, o avanço disponibilizado pela arquitetura STEP, que ainda está em desenvolvimento e onde, inclusive, este aspecto é intrínseco seu, existe de forma limitada; uma solução completa ainda não é totalmente disponível. Ramos e Deisenroth, 2006 introduziram a relevância da utilização de um padrão estruturado como STEP em um ambiente DFM integrado. Maier e Stumptner, 2007 descreveram alguns aspectos limítrofes da arquitetura STEP, especificamente no relacionado a uma representação mais completa englobando intenções de um projeto (*design intents*) através de parâmetros, restrições e história dos dados de um modelo, bem como propõem que ontologias podem ser utilizadas para melhorar este aspecto.

A informação portanto, conforme já expresso, demanda uma estrutura mais complexa que apenas a simplificada sequência típica de aquisição, leitura, operacionalização e escrita para ser efetiva. Logo, ela idealmente deve ser definida em uma arquitetura que possibilite a estruturação de seus distintos e diversos aspectos. Neste contexto, a definição sobre como a informação é utilizada é relevante bem como são sua estrutura, compreensão e utilização ótima. Em geral, a informação pode ser conceitualmente definida como a organização, de forma estruturada e compreensível, do conhecimento, conforme a figura 4.32 descreve simplificadamente.

Adicionalmente, a necessidade de um arcabouço global de modelo de informações em CAD/CAE é ressaltada por Monticolo *et al*., 2015, que propuseram e desenvolveram uma solução consistente de encapsulamento dos *dados* de parâmetros e restrições no processo de projeto. A validade, no caso, é mantida através de quatro aspectos, ou visões: identificação, rastreabilidade, reutilização e consistência. Os modelos de conhecimento, unidos em um sistema denominado KCModel, armazenam as informações em sua parte intrínseca ("interna"). Sua meta interface com informações análogas são definidas com três *focos*: produto, processo e organização estruturados através de UML-2. O sistema proposto é útil dentro do seu objetivo, escopo e inclusive está sendo utilizado na indústria. Entretanto, o arcabouço (arquitetura) baseado em UML-2 claramente limita sua completeza e flexibilidade, bem como a possibilidade de ampliação do seu escopo, como será demonstrado pela utilização

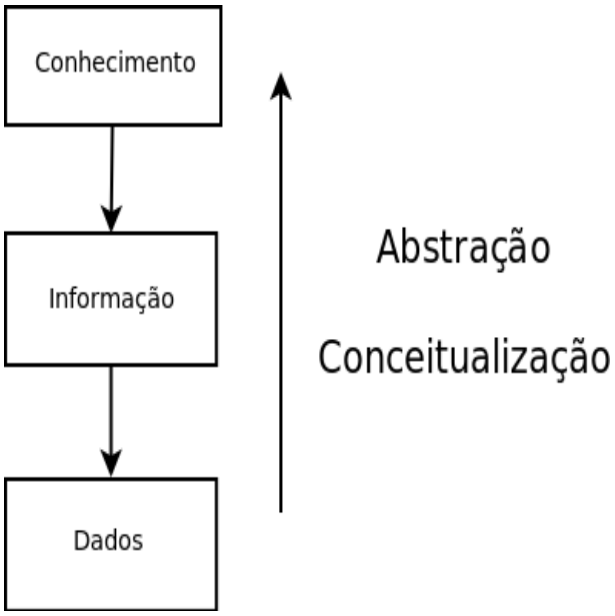


Figura 4.32 – Arquitetura de conhecimento, informação e dados.

de ontologias a seguir[5].

Desta forma, Newell, 1981 expressou, inicialmente, que para estruturar o conhecimento é necessário compreender como representá–lo. As formas de representação variam em função do ato ou processo da aquisição do conhecimento (ou cognição), segundo Platão, e do estudo do significado (ou semântica) utilizadas. A teoria mais comum associada com este tópico é que o pensamento humano funciona em termos de estruturas representacionais da mente enquanto processos computacionais (ou algorítmicos) operam sobre estas estruturas [Hofstadter, 1994]. A hipótese mais comum e relevante é que a mente humana possui representações mentais associáveis com estruturas computacionais de dados. Associadas com os requisitos de cada abordagem cognitiva, existem diferentes técnicas representacionais do ato, ou efeito, de conhecer, ou, dentro do contexto matemático, formas válidas de inferência de uma linguagem formal (ou lógica formal). A lógica formal possui três aspectos fundamentais, que são:

- Sintaxe definida: princípios que definem como sentenças[6] devem ser estruturadas.
- Semântica: define o sentido das sentenças.
- Inferência: processo analítico, sob diferentes formas, que permite definir relacionamentos entre sentenças, por exemplo como uma sentença (ou asserção) é derivada de outra.

[5]É relevante descrever que neste momento existe disponibilização ontológica em uma ferramenta UML-2 comercial.

[6]Uma expressão matemática pode ser analogamente interpretada como uma sentença.

As técnicas utilizadas para garantir estes tópicos, descritas a seguir, possuem várias características que as tornam um meio consistente para representação do conhecimento. As lógicas formais mais comuns são:

- Tríade (*triplet*) Objeto–Atributo–Valor: representação de fatos (asserções) sobre objetos com seus atributos[7].
- Fatos incertos: é a inclusão de incerteza nos fatos através, mais tradicionalmente, de fatores de certeza, ou valores numéricos, denotando o quão confiável é um fato.
- Fatos difusos: representação de incerteza nos fatos através de termos, ou vocábulos, utilizados em linguagem natural. A incerteza é quantificada através funções de pertinência relacionadas com cada termo que denota um conjunto difuso.
- Regras: é a representação do conhecimento através de estruturas que relacionam premissas, ou condições, com conclusões, ou ações/atos.
- Redes semânticas, ou mapas conceituais: são uma forma de representação do conhecimento que busca interpretar a cognição humana. Redes semânticas perfazem um modelo psicológico da memória associativa humana. Estas redes são definidas através de grafos constituídos por nós (objetos, conceitos e *stati*) conectados por arcos/ligações nomeados *não padronizados* que representam as inter–relações.
- Quadros (ou *Frames*): é uma forma de representação estereotípica do conhecimento sobre um conceito de forma similar a redes semânticas, porém com a diferença que possuem abstração mais definida em termos de dados, ou *slots*, e procedimentos (funções), ou *facets*.

Um aspecto relevante em lógica formal é que todas sentenças são ***asserções****, ou seja, sua capacidade de inferência é limitada à obtenção de valores–verdade e provas de suas asserções* [Gašević, Djurić e Devedžić, 1998]. Ademais, o conhecimento no domínio deste trabalho, e em geral, precisa modelar aspectos adicionais, tais como pesquisa, suposição, semelhança, crença, dúvida, desejo, etc. Esta coleção de asserções, com seu variado nível de interpretação contextual, tipicamente é armazenada em uma *base de conhecimento*[8]. *Em geral a obtenção uma base de conhecimento, especificamente no domínio DFM desta pesquisa, é complexa pois seu desenvolvimento e versão estável normalmente possuem diferenças cognitivas sobre o conhecimento*. Adicionalmente, existem problemas relacionados com a transformação do conhecimento a partir de sua fonte bem como sua manutenção pois ele é dinâmico.

Inicialmente, devido a estes aspectos, portanto, a utilização destas técnicas para representar conhecimento precisou efetivamente ser desenvolvida. Isto só foi possível devido ao desenvolvimento do *Nível de Conhecimento* [Newell, 1981]. O nível de conhecimento

[7] Ontologias utilizam a representação do conhecimento/informação por *triplets*.
[8] As formas de implementação da base de conhecimento podem variar bastante.

permite distinguir diferentes níveis, ou formas, de abstração em sistemas inteligentes. As formas de abstração são tipicamente classificadas como nível de implementação, nível lógico e nível do conhecimento (ou epistemológico). *Esta forma de estruturação permite distinguir o conhecimento de sua representação*.

4.3 Arquitetura sistêmica contextual de informação

O intercâmbio de informações é uma parte intrínseca em arquiteturas computacionais, particularmente DFM. A especificação de uma arquitetura genérica, ou ampla, de um domínio demanda que seu fundamento básico, que é o conhecimento, seja representado de forma adequada. Portanto, é necessário que os requisitos de conhecimento, ou sua definição em termos de informação estruturada, sejam definidos inequivocamente. Este tipo de estrutura permite a generalização, ou otimização, das abordagens de integração e compreensão entre componentes. A maioria das arquiteturas de projeto não considera de forma completa este requisito e, consequentemente, tornam-se limitadas em relação à sua flexibilidade. Esta limitação está diretamente ligada a uma característica bastante comum entre as diversas arquiteturas: dependência e foco primário em *somente dados*, geralmente. Dados pertencem um nível de especificidade que difere da compreensão, ou do conhecimento humano, conforme a figura 4.32 à página 91 denota.

Esta limitação é intrinsecamente relacionada ao fato que este detalhamento da abstração não é eficaz para representar conhecimento/informação. Em geral, conclusões viáveis para utilização da informação de forma ampla em arquiteturas computacionais atuais são restritas contextualmente, consequentemente sua flexibilidade é consideravelmente menor. Esta restrição contextual em relação à informação impede que arquiteturas sejam genéricas no *stricto sensu*.

4.3.1 Ontologias

Esta pesquisa possui um domínio heterogêneo e diversificado em termos representacionais. Tradicionalmente, na verdade historicamente pois já foi introduzida pela civilização da Grécia antiga, a utilização de ontologias[9,10] é tradicionalmente aceita como a abordagem teórica mais eficaz e eficiente. Ontologias são utilizadas no domínio DFM conforme descrito a seguir. Uma ontologia de um domínio contém sua terminologia (voca-

[9] A palavra ontologia deriva da língua grega: *ontos*, ou "ser", e *logos*, ou "palavra".

[10] A descrição do significado de uma onntologia varia segundo o autor; neste documento são utilizadas as consideradas mais abrangentes e completas.

bulário), conceitos básicos, hierarquia, restrições, taxonomia e axiomas[11]. Uma ontologia[12] é um catálogo de conceitos existentes em um domínio *D* descrito através de uma linguagem *L*. Seus componentes, ou conceitos, são descritos em termos de relações e predicados na linguagem *L*. Gruber, 1993a definiu ontologias como a especificação, ou representação formal explícita, de uma contextualização, ou visão simplificada e abstrata do mundo, mais especificamente de um domínio.

Logo, juntamente com os aspectos descritos nas seções anteriores, é possível utilizar ontologias para assegurar, ou definir, contextos da informação/conhecimento (analogamente à figura 4.32). Este é um dos principais aspectos limítrofes relacionados com a interoperação e integração entre componentes, sejam eles puramente teóricos ou computacionais. Sob o ponto de vista histórico, o vertiginoso aumento de poder computacional, seja *hardware* ou *software*, disponibilizado nas últimas três décadas não ocorreu com a mesma velocidade no que tange a formas e estruturas de informação. O foco, em geral, tem sido em termos da interoperação e integração baseada em dados, com algumas exceções, como descrito na seção 3.1 do capítulo 3, até o desenvolvimento da arquitetura STEP. Neste contexto é relevante destacar que mesmo dentro da arquitetura STEP, embora tenha havido grande evolução em relação aos formatos anteriores, todavia persistem determinadas características não explicitamente definidas. Embora não especificamente, Hendler, 2001 descreve-as nos seguintes aspectos:

Interconexões semânticas uma ontologia especifica o significado das relações entre os conceitos utilizados. Esta parte é ausente em arquiteturas de informação tradicionais, ou seja, o significado, apesar de existir, é intrínseco e disponível apenas a partir de análise do padrão em consideração.

Lógica e inferência a existência deste *dual* permite que uma ontologia tenha capacidade de ser analisada em uma asserção sob diferentes formas, ou seja, analogamente à interpretação de diferentes possibilidades, ou significados, de um dado. Esta característica não é possível em arquiteturas estáticas como STEP.

A primeira parte da definição de Hendler especifica que existe um significado (ou razão, fundamento) nas relações entre conceitos utilizados. Este aspecto é inexistente, ou intrínseco, nas representações focadas em dados. A arquitetura STEP, embora seja

[11] Princípio evidente, que não precisa ser demonstrado; máxima, sentença; norma admitida como princípio [Michaelis. . . , 2011]. Na lógica tradicional, um axioma, ou postulado, é uma sentença, ou proposição, que não é provada ou demonstrada porém é considerada como óbvia, ou como um consenso inicial necessário para a construção, ou aceitação, de uma teoria.

[12] Ontologias são fundamentadas em dois aspectos sistêmico-operacionais: definição e inferência. Componentes definidos, com suas propriedades e relações, descrevem como o escopo de um domínio é aceito pelo projetista da ontologia. Componentes inferidos, com suas propriedades, e relações descrevem como o escopo de um domínio é *interpretado* após a utilização de um motor de raciocínio, ou inferência. Esta inferência é que realmente demonstra como uma ontologia representa o escopo de um domínio.

sistêmica, ainda possui foco-objetivo final em dados. Ontologias aprimoram, ou refinam, a representatividade de informações em domínios heterogêneos. Isto é disponibilizado através de três aspectos, descritos abaixo, para que seja possível o compartilhamento e reutilização da informação, especificamente no domínio de engenharia. É relevante salientar que, mesmo que ontologias proponham uma abordagem mais eficaz para gerenciamento e armazenamento da informação, este é um aspecto todavia bastante complexo e, efetivamente, é onde o padrão STEP se destaca: a forma, ou *estrutura do dado*, da informação descrita.

Idealmente, uma ontologia define um domínio de forma completa. A união entre vocabulário e taxonomia forma, portanto, uma arquitetura conceitual para utilização da informação em um domínio, ou seja, seu conteúdo. A existência de uma linguagem permite verificar a consistência da informação entre componentes do domínio além de permitir, através deste aspecto, habilitar uma característica fundamental em arquiteturas integradas: a estrutura de seu contexto. Os principais componentes de ontologias são o vocabulário, a taxonomia e o conteúdo.

Vocabulário uma ontologia possui a definição de um vocabulário controlado (ou finito) em um domínio selecionado. Adicionalmente, o vocabulário tem associado definições lógicas sobre a validade de cada vocábulo bem como regras que definem interconexões entre eles. O vocabulário possui significado único com semântica independente do contexto. Esta independência permite que o significado seja intercambiado entre usuários e aplicações.

Taxonomia a ontologia hierarquiza de forma categorizada o vocabulário possibilitando o agrupamento de suas entidades conforme características comuns. A taxonomia é definida e possui suas entidades agrupadas de forma mutuamente exclusiva em grupos e subgrupos. Ponzetto e Stube, 2011 descrevem como a relevância da utilização da taxonomia em ambientes heterogêneos e diversificados demanda análise de sua qualidade em relação a seus conceitos associados. No caso, os autores propõem um método baseado na concorrência de significado, ou na existência de um mesmo significado (similaridade semântica) entre termos existentes na taxonomia e conjuntos–padrão de dados com resultados comparativamente favoráveis a métodos manuais de classificação de conceitos.

Conteúdo A característica principal de uma ontologia é identificar objetos e suas relações hierarquizadas em um domínio. Adicionalmente, através disto, ela possui uma linguagem para representar estes aspectos. Desta forma, essencialmente, ontologias perfazem o que as pessoas produzem para esclarecer fatores que motivam os indivíduos e o comportamento humano, ou teoria do conteúdo [Chandrasekaran, Josephson e Benjamins, 1999, Bylander e Chandrasekaran, 1988].

4.3.2 Modelagem da informação

A modelagem da informação demanda que, inicialmente, suas características sejam compreendidas de forma adequada e que sua representação seja modelada de forma que sua utilização ótima seja possível, neste caso por componentes auxiliares de projeto. Selic, 2003 identificou os aspectos fundamentais, válidos e necessários para modelar a informação em engenharia.

Abstração um modelo é sempre uma representação reduzida, ou simplificada, de um sistema por ele representado.

Compreensibilidade é um requisito básico representar um modelo da forma mais análoga à intuição.

Precisão um modelo necessita representar de forma real, ou exata, as características importantes e relevantes de um sistema.

Preditibilidade um modelo necessita ter a capacidade de, adequadamente, "prever" os aspectos não óbvios de interesse de um sistema.

Acessibilidade/economia um modelo precisa ser mais fácil de construir e analisar que um sistema modelado diretamente.

A análise destes aspectos, associada com a demanda por abstração da informação descrita anteriormente, impõe um tipo distinto de estrutura denominado meta–modelo. O meta–modelo possui classes de um sistema que são autoidentificadas em uma determinada linguagem de modelagem. O meta–modelo define as formas de expressão, inferência e raciocínio, ou racionalidade, em modelos válidos de uma linguagem [Seidwitz, 2003]. Especificamente, o meta–modelo é uma representação clara e explícita dos relacionamentos e bases representacionais com as regras necessárias para sua definição em um determinado domínio.

Analogamente a modelos, existe o nível meta relativo a dados, que é denominado meta–dado. Em resumo, meta-dados são dados que referem-se, ou melhor descrevem, o significado dos próprios dados. Tipicamente, a descrição de um dado contém pelo menos um meta-dado embutido. A descrição de um determinado comando em uma linguagem como HTML, por exemplo, embute determinada formatação de um dado.

A representação de modelos de informação está relacionada, tipicamente, com uma arquitetura em camadas associada com modelos genéricos, denominada *Meta–Object Facility* (MOF), conforme a figura 4.33. A MOF utiliza formatos descritivos internos embutidos, tais como UML e XMI. Os meta-dados consequentes de um meta-modelo são definidos pelo modelador, ou seja, dependem de uma determinada abstração de um

domínio. As camadas descritas que permitem modelar a informação e estão detalhadas na figura 4.33.

Figura 4.33 – Arquitetura Orientada a Modelos (MOF) [Gašević, Djurić e Devedžić, 1998].

Portanto, conforme descrito introdutoriamente nesta seção e na anterior, é fundamental corretamente modelar a informação para que ela seja útil de forma eficaz e eficiente. Embora ainda não exista um método determinístico para modelagem da informação, de fato ele existe intrinsecamente devido a suas características semânticas. Um típico processo sequencial com este objetivo e adaptado para esta pesquisa [Noy e McGuinness, 2004] é descrito abaixo de forma condensada na figura 4.34. Fernández-López *et al*., 1999 descrevem este aspecto mais genericamente.

1. Definição do domínio e escopo da ontologia: perfaz a objetiva clarificação do que está sendo modelado. Neste caso, o domínio é DFM e seu escopo restrito a peças poliédricas metálicas sujeitas a processos de remoção de material, ou usinagem.

2. Reutilização de ontologias pré–existentes: se for considerado o domínio completo, na literatura não foram identificadas ontologias desenvolvidas, ou um escopo análogo neste domínio devido à definição de um escopo deveras restritivo. Desta forma, esta etapa não foi utilizada.

3. Enumeração dos tópicos, ou termos, relevantes: os termos relacionados com esta pesquisa estão descritos nas seção 4.3.3.3 e suas subseções.

4. Definição de classes e sua taxonomia: a estrutura taxonômica é definida baseada na integração dos conceitos da informação em função dos princípios do topo para a base

(*top-down*), ou mais genérica para mais detalhada, e do meio para as extremidades (*middle-out*).

5. Definição das propriedades (*slots*) das classes: *slots* detalham a estrutura operacional da informação de cada classe do domínio, especificamente *slots* internos e externos bem como sua relação com o nível mais inferior de detalhe da informação que é o dado em *instâncias*.

6. Caracterização dos *slots*, denominados *facets*: são detalhes como permissibilidade de determinado conceito em um *slot* e/ou sua cardinalidade.

7. Instanciação de dados (valores) das *facets* dos *slots*.

Esta sequência descrita demonstra como a partir de conceitos humanos, sob certo aspecto abstratos, é possível obter, de forma concisa e objetiva, o dado. Desta forma, o dado, portanto, se analisado diretamente, não representa a totalidade do significado embutido na sua representação, seja ela numérica ou através de vocábulos.

4.3.3 Modelagem da Informação em Projeto para Manufatura

A relevância do gerenciamento adequado da informação em DFM, sua modelagem e, idealmente, representação arquitetônica computacional foi ressaltada por Chang, Rai e Terpenny, 2010, que arguiram que existe a possibilidade de redução de erros e inconsistência dos dados através da utilização de ontologias; ademais, a base de conhecimento do domínio é ampliada. Consequentemente, decisões mais complexas de projeto podem ser feitas de forma mais eficiente reduzindo custos. Esta hipótese foi provada através de casos no domínio de soldagem. Adicionalmente, embora soluções consistentes tenham sido desenvolvidas como, por exemplo, Xiao *et al*., 2015 com um arcabouço CAD/CAM/CNC eficaz e eficiente, ela é todavia focada primariamente apenas na representação direta da informação através de dados pois segue partes do padrão STEP que não incluem, explicitamente, a completude da informação.

Mais recentemente, inclusive, podem ser identificadas pesquisas nas quais a inclusão de ontologias poderia otimizar a abordagem de resolução do problema. Vide, por exemplo, Madhusudanan e Chakrabarti, 2014 com a utilização de um sistema especialista para diagnóstico de erros em operações de montagem e especialmente Cambaa *et al*., 2014, que baseiam-se em uma ligação, através de XML, entre notas, ou anotações, sobre aspectos geométricos 3D para representar intenções de projeto.

4.3.3.1 Definição taxonômica e seu detalhamento

A necessidade de representar a informação, e eventualmente conhecimento, em um domínio demanda que seja definido o significado das palavras, que descrevem conceitos

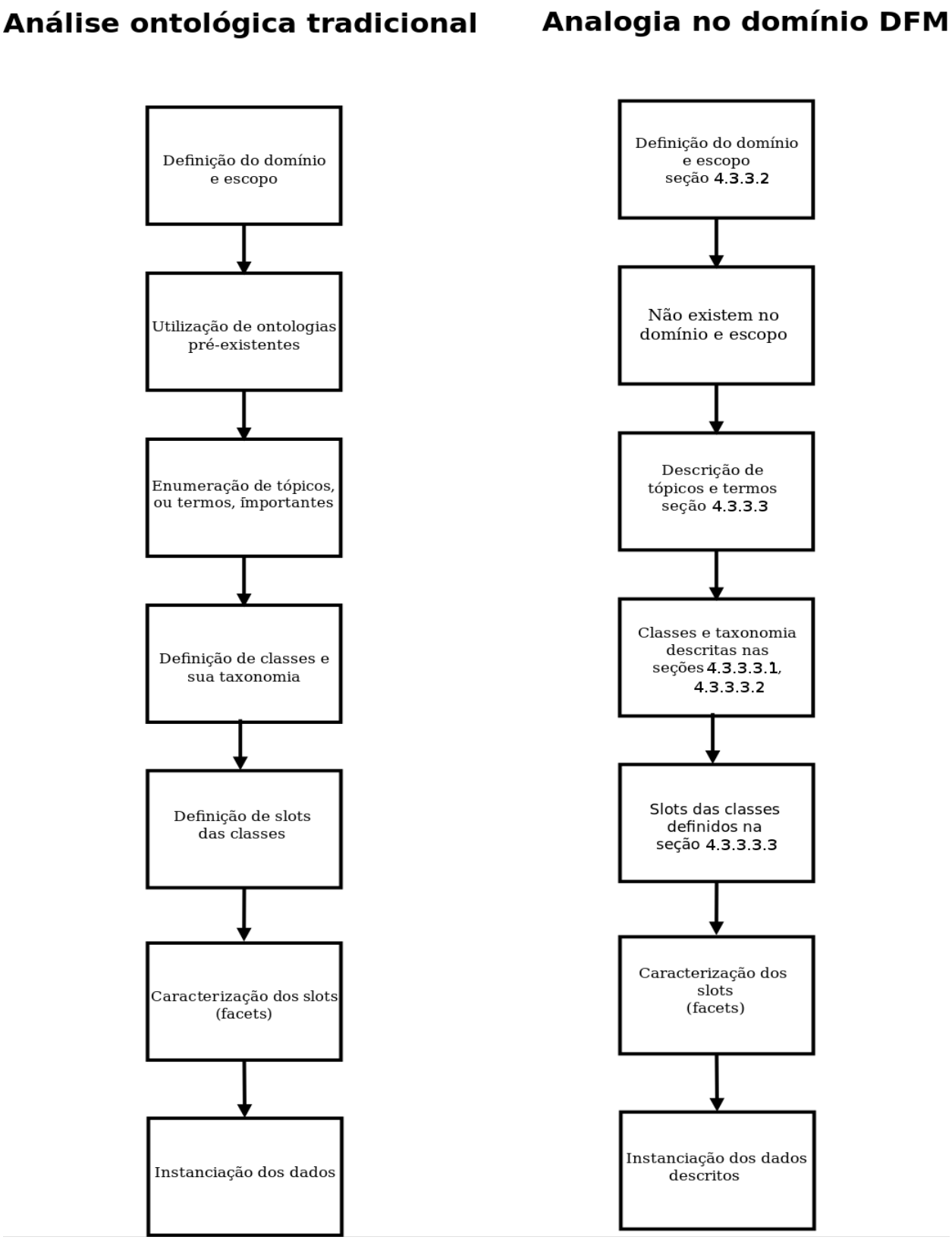


Figura 4.34 – Arquitetura de desenvolvimento de uma ontologia e sua utilização no domínio DFM desta pesquisa.

associados, utilizadas. Seu significado é o contexto da informação, ou seja, como ela é compreendida e utilizada pelos usuários ou componentes sistêmicos. Efetivamente, este é o primeiro aspecto necessário para modelar a informação nesta pesquisa: a taxonomia DFM. A utilização de ontologias permite definir uma hierarquia de conceitos ou, mais especificamente, a compreensão de um domínio. Tipicamente, uma ontologia é estruturada em categorias e subcategorias (ou classes e subclasses).

Ontologias definem uma ***arquitetura conceitual de informação***. Esta arquitetura de informação define as inter–relações hierárquicas entre classes/subclasses de forma estrita, ou seja, inclui até as inter–relações existentes entre suas instâncias[13] formais. A definição até o nível de detalhe da informação, que são instâncias, permite que sua consistência seja garantida. A definição da arquitetura conceitual de informação com objetos classificados em classes e suas inter–relações permite que ontologias sejam também conhecidas como teorias de conteúdo [Chandrasekaran, Josephson e Benjamins, 1999]. Desta forma, uma ontologia permite "discernir", ou interpretar/compreender, o contexto da informação representada.

Efetivamente, ontologias permitem representar a dependência entre determinados aspectos da informação através de correlações, ou "referências cruzadas". Através disto, as intercorrelações permitem compreender o significado real da informação, que é oposto ao tradicional foco puramente em dados. *Na abordagem tradicional toda interpretação contextual é implícita e, geralmente, não permite realizar inferências sobre sua validade*.

A interpretação do significado real da informação, portanto, implica que seu contexto seja conhecido e, idealmente, estruturado em um arcabouço compreensível por seus usuários. O domínio é, desta forma, o primeiro aspecto que precisa ser definido. O domínio selecionado é Projeto Orientado à Manufatura (DFM), conforme expresso inicialmente na seção 4.1. Borgo, 2014 enfatiza a relevância da estruturação adequada da informação para o domínio industrial, particularmente relacionando aspectos taxonômicos e sua intrínseca interdependência entre conceitos. Neste caso, sua abordagem propõe uma taxonomia mais abstrata e genérica neste domínio, que é uma arquitetura díspar desta pesquisa, bem como possui um escopo distinto.

O domínio desta pesquisa possui uma quantidade relevante de diferentes tipos de informação inter–relacionados entre si variando desde dos dados "*per se*" até como eles são utilizados. A forma de sua utilização é, efetivamente, um fator limitador para o desenvolvimento de sistemas tanto teóricos quanto computacionais deste tipo. Assim sendo, a hipótese básica fundamental desta pesquisa é que a representação/estrutura da informação é mais eficiente e eficazmente obtida através da utilização de ontologias.

Esta hipótese é suportada por diversos autores, por exemplo Gruber, 1994a, que

[13]Uma instância define a unidade, ou forma básica, de representação da informação. Ela pode, por exemplo, variar desde a descrição de uma unidade de informação, como metro, ou milímetro, até um número.

inclusive associa esta necessidade com o desenvolvimento de produto [Gruber, Teinembaum e Weber, 1992]. Esta hipótese é enfatizada por Gruber, 1994b, Gruber, 1993a e por Gašević, Djurić e Devedžić, 1998. A utilização de ontologias perfaz tanto sua aquisição, ou extração, quanto sua obtenção [Wei, Qin-yi e Tian-yi, 2009].

Efetivamente, portanto, um arcabouço mais efetivo de informação com sua eventual implementação lida com o conhecimento e informação. Inclusive, este aspecto é ressaltado inicialmente por Brandt *et al.*, 2006 na arquitetura PDW e, de forma temporalmente coincidente, mais especificamente, por Lemaignan *et al.*, 2006. Estes autores propõem uma ontologia DFM (MASON) em um escopo distinto desta pesquisa, além de ser mais abstrata, pois seus componentes são entidades, operações e recursos de manufatura. O objetivo é disponibilizar o substrato de informação para estimação automática do custo e uma ligação de alto nível entre a ontologia e uma arquitetura de agentes para simulação da manufatura.

Giovannini *et al.*, 2012 enfatizam a usabilidade de ontologias no domínio de manufatura; seu escopo, entretanto, é díspar desta pesquisa, além de sua implementação ser baseada em um *dual* de tipos de regras SWRL e SQWRL. Neste caso, o foco é a sustentatibilidade do processo de manufatura *per se* centrado no produto. Consequentemente, existe a necessidade de representar de forma otimizada a informação, aspecto que, neste caso, utiliza mapeamento dos relacionamentos entre os contextos através funções, ou seja, simplificando, onde e como, taxonomicamente, cada componente de manufatura é utilizado no projeto de um produto. Embora todavia incompleta devido à falta de integração com processos organizacionais, a pesquisa mostra, com sucesso, a possibilidade de integração de funções, aspecto conceitualmente bastante relacionado com o ser humano, numa ontologia deste domínio.

4.3.3.2 Definição do domínio e escopo da ontologia

O domínio–objeto desta pesquisa: DFM é amplo e abrangente possuindo, consequentemente, heterogeneidade relevante. Esta abrangência, embora aparentemente limitante, pode ser avaliado de forma mais eficiente através de ontologias. Levando–se em consideração o escopo do domínio, é necessário restringi-lo para que os aspectos básicos e fundamentais desta pesquisa sejam analisados a contento. Desta forma, a informação no domínio DFM é estruturada segundo o escopo selecionado: processos de usinagem de peças metálicas prismáticas, sem prejuízo da generalidade.

A flexibilidade dos fundamentos teóricos de ontologias, descritos na seção 3.2, permite representar sua variabilidade de forma objetiva e sucinta. Fundamentada nisto, a definição da arquitetura da informação é feita a partir de sua parte mais abstrata até a mais detalhada, analogamente à figura 4.34, a qual é baseada, sob aspecto genérico, no descrito na figura 4.32. Wei, Qin-yi e Tian-yi, 2009 utilizaram ontologias para definir

aspectos implícitos, ou embutidos, para otimização da busca da informação no domínio DFM.

4.3.3.3 Caracterização dos detalhes do domínio da informação

O domínio DFM intrinsecamente engloba diversos aspectos que possuem dependência entre si desde o dado, por exemplo valores numéricos, até sua interpretação e posterior utilização (perfazendo sua interpretação). A utilização de conceitos associados, e necessários, aos dados torna este aspecto ainda mais complexo. Em geral, a manufatura possui conceitos fundamentalmente teóricos, que possuem fundamento tanto em sintaxe quanto em gramática, e que descrevem boas práticas, que são as regras otimizadas de fabricação.

Estas regras definem interdependências e relacionamentos entre os diversos e diferentes conceitos necessários para estruturar a compreensão, e eventual utilização, da informação disponível para solucionar um problema específico. Assim sendo, somente o dado não permite inferir seu significado diretamente a partir de todos seus conceitos associados. Isto acarreta uma sobrecarga relevante, e muitas vezes limitante, sobre sistemas de projeto.

4.3.3.3.1 Taxonomia

A breve introdução anterior descreveu o quão relevante é interpretar globalmente como a informação é compreendida. Desta forma, fica claro como existe a necessidade de estruturar a informação para que sua compreensão, posterior efetiva e eficiente utilização, seja a mais fidedigna possível. A estrutura da informação, ou taxonomia, é a definição de uma hierarquia conceitual dos termos, ou conceitos, utilizados dentro de um domínio determinado bem como suas inter-relações e dependências.

A taxonomia permite organizar a informação, que é baseada em conceitos, através de sua hierarquização, ou, de forma simplista, através de categorias denotando dependências com suas relações e correlações. Adicionalmente, a taxonomia pode incluir regras que permitem a validação[14] de uma determinada estrutura da informação, inclusive do conhecimento, de um determinado domínio.

A definição do domínio e seus componentes é o primeiro passo para definição da estrutura da informação. Entretanto, é relevante destacar, novamente, que o domínio DFM é bastante abrangente pois inclui todos os partícipes de um processo de manufatura. Assim sendo, levando em consideração esta abrangência, o processo de definição da taxonomia da informação nesta pesquisa possui escopo menor que todo domínio, sem perda de

[14] A validação pode englobar apenas conceitualização pura e/ou sua operacionalização, pois depende do objetivo da modelagem da informação.

generalidade. Consequentemente, ele pode, e muitas vezes demanda, ser customizado para especificidades de outro escopo.

A relevância de uma arquitetura de informação para ambientes integrados, no caso utilizando STEP através de OntoSTEP, foi proposta por Zha e Du, 2002. O domínio DFM possui grande variabilidade da informação utilizada e inclui desde aspectos pessoais ao maquinário e ferramental sujeitos à demanda de um determinado projeto específico, o qual, por sua vez, possui tanto informações geométricas quanto de fabricação como, por exemplo, material, resistência e tolerância associadas. Ele tem por objetivo utilizar aspectos *específicos* considerados relevantes de DFM. Excetuando a parte relacionada diretamente com pessoal, estes aspectos, brevemente descritos a seguir, englobam contextualmente DFM.

Materiais toda manufatura tem por base a modificação de características dos materiais, seja geométrica ou composição estrutural. Desta forma, torna–se necessário que suas características conceituais fundamentais estejam presentes em uma arquitetura de informação.

Disponibilidade de máquinas e ferramentas para que uma determinada peça seja fabricada, obviamente é necessário que o conjunto de maquinário, ferramental e elementos de fixação esteja disponível e seja adequado. Nesta pesquisa, entretanto, apenas a disponibilidade do *dual* máquina–ferramenta é utilizada, sem perda de generalidade.

Acessibilidade da ferramenta enquanto os dois tópicos expressos acima lidam com a peça e a capacidade estrutural de fabricação, em DFM é necessário otimizar o projeto levando em consideração a integração dentre estas duas partes. Adicionalmente, é necessário integrar aspectos geométricos de uma determinada peça. Logo, por consequência, é um requisito conhecer se a ferramenta pode usinar, no caso do processo deste escopo, as características geométricas (*features*) de uma peça. Sob contexto relativamente recente, a arquitetura STEP [Pratt, 2001, Liang *et al.*, 1999, Loffredo, 1999] tornou–se o padrão praticamente inconteste nesta área apesar de todavia ser incompleta, por definição inclusive.

Tolerância por definição prática, todo projeto de peça, ou produto, é feito para ser eventualmente fabricado. Concomitantemente, portanto, é necessário considerar os efeitos do processo de fabricação, o qual possui limitações físicas que resultam em dimensões diferentes das projetadas. Tradicionalmente, estas variabilidades são expressas por tolerâncias. Tsai, Chuang e Guo, 1998 descreveram a complexidade inerente com o desenvolvimento de um modelo de informação de dimensões e tolerâncias na arquitetura STEP. Esta complexidade é relacionada com o fato de a arquitetura STEP, atualmente, ser majoritariamente focada em dados, tornando complexa a operacionalização da utilização da informação representada. Considerando este aspecto, Maier

e Stumptner, 2007 explicitam a necessidade de um ambiente, ou arquitetura, mais completa que o padrão STEP utilizando ontologias.

Custo no ambiente competitivo de mercado no sistema capitalista o preço, na maior parte da vezes, é fator determinante na seleção de uma das alternativas de manufatura disponíveis. Sabidamente, o aspecto fundamental na determinação do preço possui sua base no custo de um produto. A formação do custo tem diversas variáveis relacionadas com os diferentes componentes de um produto. Elas perfazem aspectos da infra–estrutura de manufatura, incluindo a mão de obra (técnica, administrativa e operacional) existente nas distintas fases do desenvolvimento, comercialização, manutenção e suporte além dos materiais necessários e maquinário com seu ferramental. Sob o ponto de vista especificamente de manufatura, entretanto, o custo pode ser subdividido genericamente em duas partes: custo de investimento e custo de processamento.

Considerando a parte operacional e, adicionalmente, incluindo a informação contextualizada, Mikos, 2008 definiu um modelo para analisar um relevante problema existente nos sistemas de fabricação atuais: a falta de conformidade entre componentes de manufatura. Neste caso, a análise sistêmica é feita através de analogia com o processo e problema real através de agentes de *software*, enquanto ontologias são utilizadas de forma a contextualizar os requisitos da informação e as consequências de inferência de seus resultados para o caso de conformidade. Enfim, a utilização da associação entre ontologias e agentes permite resolver este complexo problema de forma eficaz e efetiva.

Conforme descrito por Posada *et al*., 2005-2006, a informação em ambientes integrados heterogêneos, como DFM, demanda uma arquitetura mais completa exatamente devido a este aspecto [Krima *et al*., 2009a, Krima *et al*., 2009c]. *Esta heterogeneidade implica em dependência, que pode, inclusive, incluir semântica, entre os seus componentes*. A falta de capacidade de representação semântica consistente e flexível, direta ou relativa, nos formatos atuais, particularmente STEP, dificulta bastante a estruturação de uma arquitetura de informação flexível; este é um aspecto a ser ressaltado, inclusive já descrito previamente neste documento.

Este fato resulta em redução da flexibilidade sistêmica tanto na parte de fundamento teórico da informação quanto em uma eventual parte de sua implementação computacional. Desta forma, Barbau *et al*., 2012 procuram reduzir o vão entre formas de representação, ou melhor expresso, estrutura e arquitetura da informação através um uma "ligação" entre elas denominada OntoSTEP. Essa pesquisa procura permitir a inclusão de componentes STEP, com suas partes terminológica e assertiva, numa ontologia mais genérica no domínio. Desta forma, é possível incluir a contextualização que, tradicionalmente e equivocadamente, é associada através de mapeamento direto com dados em uma arquitetura de informação. A inexistência deste aspecto ocasiona dois relevantes problemas:

1. A falta de uma taxonomia e descrição de seu inter–relacionamento limita, tipicamente muitas vezes impede, a compreensão adequada do real significado da informação descrita pelos dados.

2. A falta da taxonomia impõe uma sobrecarga adicional, muitas vezes inclusive limitadora, no desenvolvimento de sistemas, especificamente DFM.

Nesta pesquisa a definição da estrutura da informação utiliza a ferramenta Protégé [PROTÉGÉ TEAM, 2014] que permite, de forma relativamente flexível, sua definição bem como customização. Adicionalmente, ela permite que sejam incorporadas novas características, tais como o motor de raciocínio customizável e interfaces com *software* procedural ou algorítmico. As próximas subseções, portanto, utilizam a descrição da informação baseada nesta ferramenta, que segue os princípios genéricos descritos previamente. Desta forma, conforme já expresso, existe a necessidade de definir e, de certa forma, restringir o domínio para que o conceito de como estruturar a informação para o domínio DFM seja demonstrável factivelmente, sem perda da generalidade.

Os componentes do domínio precisam ser analisados em suas relações e correlações de forma que seus requisitos de informação sejam compreendidos. A hierarquia conceitual, além de estruturar melhor a informação, permite que a já inerente complexidade deste aspecto, que incorre em redução da flexibilidade sob o ponto de vista sistêmico-implementacional, seja reduzida significativamente. A taxonomia é analisada através de asserções lógicas, tradicionalmente expressas através de premissas válidas que sempre resultam em conclusões válidas, ou lógica de primeiro grau, ou, inclusive, conforme necessário, através da utilização de interpretações sobre conceitos - lógica modal (vide o Apêndice A). Inicialmente, é necessário definir os conceitos próprios que serão analisados bem como suas inter-relações para que suas propriedades, ou relacionamentos, e hierarquia possam ser utilizadas. O domínio e escopo DFM desta pesquisa possuem, em geral, alguns conceitos básicos que são utilizados para estruturação da informação, correspondentes a:

Material (Mat)	$\forall m$	$\in$	$\mathsf{Mat}(m)$: m é um material.
Peça (Pe)	$\forall p$	$\in$	$\mathsf{Pe}(p)$: p é uma peça.
Feature (Fe)	$\forall fe$	$\in$	$\mathsf{Fe}(fe)$: fe é uma *feature*.
Dimensão (Dim)	$\forall d$	$\in$	$\mathsf{Dim}(d)$: d é uma *dimensão*.
Tolerância (Tol)	$\forall t$	$\in$	$\mathsf{Tol}(t)$: t é uma tolerância.

Máquina (Maq)	$\forall ma$	$\in$	Maq(ma): ma é uma máquina.
Ferramenta (Ferr)	$\forall fe$	$\in$	Ferr(fe): fe é uma ferramenta de corte.
Operação (Oper)	$\forall o$	$\in$	Oper(o): o é uma operação.
Usinagem (Usin)	$\forall u$	$\in$	Oper(u): $u \in$ Usin $\forall$ Usin $\subseteq$ Oper usinagem é um subconjunto das operações possíveis.
Custo (C)	$\forall c$	$\in$	C(c): c é uma unidade monetária.

A partir da definição dos conceitos básicos, a próxima etapa é estruturar suas inter–relações, ou seja, como um tipo de informação se relaciona com outro. Sob forma simplificada, é a taxonomia de inter–relacionamento da informação. Estas inter-relações, entretanto, não são limitadas à hierarquização entre conceitos, porém incluem desde como as informações são relacionadas entre si até sua representação integrada definida explícita, que é o dado, além de regras e condições cardinais–lógicas.

Utilizando a estrutura mais abstrata, e com especificidade em relação aos limites da arquitetura, é possível e necessário analisar e definir como e de que forma os conceitos se inter-relacionam, bem como a forma de interfaceamento da informação. Esta especificação permite conhecer qual a forma de utilização de conceitos e associa o conhecimento com a finalidade descritiva da informação. Efetivamente, sua análise e estruturação permitem desagregar a informação de seu conteúdo.

Sob contexto de fabricação: $\forall \text{Dim} \Rightarrow \text{Tol}$

Disp. de máquinas e ferram. (Disp) $\forall di \in \text{Disp}: di$ conj. de máquinas e ferram.

Acessibilidade da ferramenta (Acess) $\text{Fe} \wedge (\text{Dim} \wedge \text{Tol}) \wedge \text{Ferr} \wedge \text{Disp} \Rightarrow \text{Acess}$

$$
\begin{aligned}
\text{Disp} &\Rightarrow \text{Maq} \\
\text{Disp} &\Rightarrow \text{Ferr} \\
\text{Disp} &\Leftrightarrow \text{Mat} \\
\text{Disp} &\Leftrightarrow \text{Dim} \\
\text{Disp} &\Leftrightarrow \text{Tol}
\end{aligned}
$$

Custo (C)

$$
\begin{aligned}
\text{Mat} &\Rightarrow \text{C} \\
\text{Tol} &\Rightarrow \text{C} \\
\text{Acess} &\Rightarrow \text{C} \\
\text{Disp} &\Rightarrow \text{C} \\
\text{Tol} \wedge \text{Acess} \wedge \text{Disp} \wedge \text{Mat} &\Rightarrow \text{C}
\end{aligned}
$$

Idealmente, uma arquitetura de informação deve ter flexibilidade para ser integrada com componentes sistêmicos, onde cada um deles é tradicionalmente focado em um tópico específico. Ademais, este aspecto é extremamente relevante em sistemas CAD/CAM e DFx devido à tradicional e típica situação de *software* legado. Neste contexto, e sob o aspecto histórico, a arquitetura baseada em agentes de *software* mostrou ser a mais flexível disponível para sistemas integrados [Frost e Cutkosky, 1996, Nyulas *et al.*, 2008, Bellifemine, Caire e Greenwood, 2007]. A arquitetura de agentes, inicialmente proposta por Bradshaw, 1997, *fundamenta–se no desacoplamento entre solução de um problema específico e um algoritmo fixo com este fim*, ou seja, ela é obtida *iterativamente* de forma automática segundo regras de dependência.

Conforme descrito a partir do início deste capítulo e no capítulo anterior, é relevante que em um ambiente heterogêneo e dinâmico como DFM existam dois aspectos principais: compreensão adequada da abstração intrinsecamente envolvida nas informações e gerenciamento ótimo das dependências operacionais entre seus componentes. Neste aspecto, Aart *et al.*, 2002 demonstraram a possibilidade da sua união de forma a gerenciar otimamente tanto a arquitetura de integração entre componentes quanto a compreensão adequada das informações que habilitam a sua operacionalização ótima.

Portanto, é relevante incluir este tipo de conceito também sob o contexto de informação. Na arquitetura de informação, desta forma, as atividades básicas associadas com agentes de *software*, bem como sua hierarquização e associações, são definidas intrinsecamente na ontologia. Consequentemente, ocorrerá a simplificação da operacionalização de sistemas.

Componente (Ag)	$\forall co \leq \mathsf{Ag}(co)$: *co* agente-componente.
DFM (DFM)	$\exists co \leq \mathsf{DFM}(co)$: *co* componente DFM.
Dispon. de máquinas e ferr. (Disp)	$\forall di \in \mathsf{Disp}$: *di* conj. de máquinas e ferram.
Acessib. da ferramenta (Acess)	$\mathsf{Fe} \wedge (\mathsf{Dim} \wedge \mathsf{Tol}) \wedge \mathsf{Ferr} \wedge \mathsf{Disp} \Rightarrow \mathsf{Acess}$

4.3.3.3.2 Hierarquia Taxonômica (Classes)

A partir dos conceitos, ainda que genéricos, é necessário definir sua hierarquia com o objetivo de estruturar o arquétipo conceitual de uma informação específica. Neste caso, é relevante ressaltar que a taxonomia, tipicamente descrita por classes, não necessariamente é uma analogia computacional *ipsis–literis*, pois o foco é o relacionamento entre os conceitos de uma base de conhecimento através de seu vocabulário. Portanto, as classes mais genéricas definidas consideram os seguintes aspectos,

- A estrutura da informação tem por objetivo ser utilizada numa arquitetura de agentes de *software*.

- As classes são subdivididas em nove áreas, ou conceitos, descritos a seguir, juntamente com sua hierarquia gráfica em alguns deles: DFM, Análise, Processo, Operação, Produto, Material, Dimensão, Pessoal e Custo. Esta hierarquia taxonômica perfaz conceitualmente a interpretação mais aceita em termos práticos, bem como advinda de referências na literatura.

1. **DFM** Projeto Orientado para Manufatura essencialmente possui um tipo de informação, com seu conhecimento associado, heterogêneo e por esta razão tem, na sua maioria, inter relações entre as informações de seus componentes. Desta forma, apenas aspectos taxonômicos mais genéricos são descritos nesta classe enquanto seu detalhamento, ou especificidade, está disperso associado nos relacionamentos entre classes e propriedades. Seu detalhamento descreve a parte operacional, isto é, como são inter-relacionados os conceitos de forma não hierárquica.

   Os princípios detalham tanto a hierarquia conceitual quanto de relacionamentos, ou "operacional", sobre a informação em DFM. As atividades são compostas pela informação associada com a ação de cada componente. As figuras 4.35[15] e 4.36 descrevem parte da taxonomia hierárquica definida de atividades e princípios e a taxonomia como representada no Protégé, respectivamente.

   A descrição–base de aspectos DFM–específicos claramente não totaliza as informações deste domínio, conforme expresso nas tabelas 4.6 e 4.8. Assim sendo, é necessário complementar suas informações com relações práticas, ou "princípios metodológicos", que perfazem sua representação intrínseca; esta pesquisa, apesar de contemplar este requisito, não o possui em todo seu escopo. A representação deste tipo de informação em sistemas tradicionais é intrínseca na solução, limitando a sua flexibilidade.

   Especificamente, a taxonomia associada com princípios e regras DFM é que efetivamente descreve este domínio é interpretado por seres humanos, ou seja, é o que perfaz este contexto. As figuras seguintes[16] descrevem, sucintamente, como parte da taxonomia, ou compreensão hierárquica e operacional, é definida. A taxonomia, por sua vez, está associada com um aspecto intrínseco embutido na análise ontológica: um motor de inferência, que depende de dois aspectos:

   - Ontologia definida (*asserted*) - é a forma, ou estrutura, como o conceito é compreendido/definido pelo usuário/desenvolvedor.
   - Ontologia inferida (*inferred*) - é a forma, ou estrutura, como a ontologia *resultante* de um processo realizado por uma *máquina de inferência* deduz a taxonomia. Os

[15] A figura mostra graficamente em termos de sua coloração: classes em laranja são classificadas como tendo *condições necessárias e suficientes*, ou **definidas**, (qualquer indivíduo que satisfaça às definições desta classe pertence a este tipo) enquanto componentes em amarelo são classes **primitivas**, ou seja, possuem condições necessárias.

[16] Figuras 4.37, 4.38, 4.39 e 4.40.

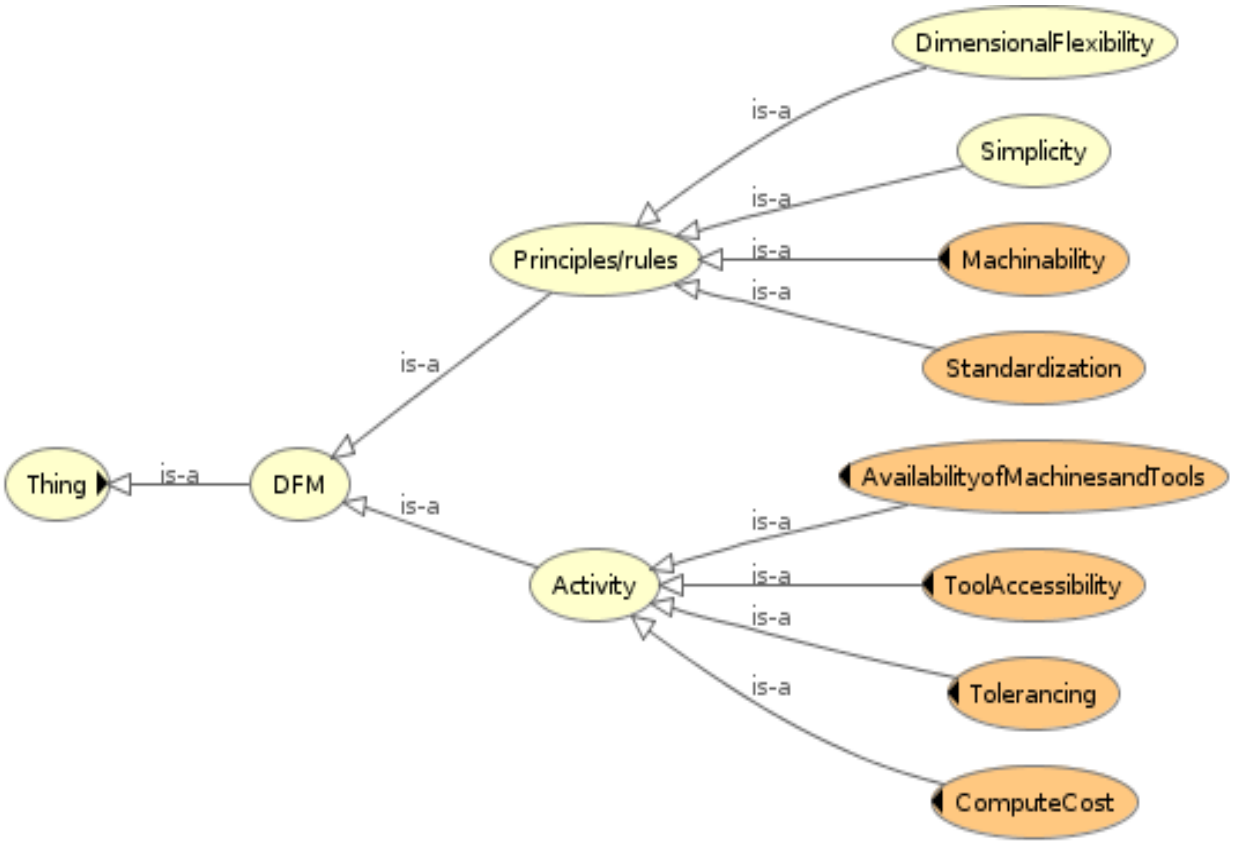


Figura 4.35 – Taxonomia definida relacionada ao contexto DFM.

> resultados de uma máquina de inferência podem variar pois existem diversas disponíveis e, consequentemente, a taxonomia resultante também pode ser diferente[17].

A análise de uma taxonomia perfaz, portanto, tanto como ela é definida quanto como ela é inferida. Em geral, a taxonomia inferida é mais abrangente que a definida e define o significado verdadeiro de uma ontologia. Este é um aspecto inexistente na definição e utilização da informação em sistemas tradicionais, ocasionando diminuição da sua flexibilidade e generalidade. Este aspecto é muito importante pois a inferência, ou compreensão, das informações utilizadas é intrínseca, ou faz parte, à solução de um dado problema.

A partir da definição dos relacionamentos entre os componentes do domínio DFM, também é necessário explicitar suas incompatibilidades. Este fato deve-se a um fundamento da utilização de ontologias denominado Suposição de Mundo Aberto (OWA), que impõe que qualquer inferência sobre um conceito é válida até prova em contrário. Desta forma, a especificação da definição de um contexto é considerada completa apenas quando todas as possibilidades de diferentes interpretações alternativas forem definidas. Tradicionalmente, isto é feito através da definição da exclusão,

[17] Analogamente, este aspecto é relacionado, sob contexto humano, como diferentes interpretações de uma asserção.

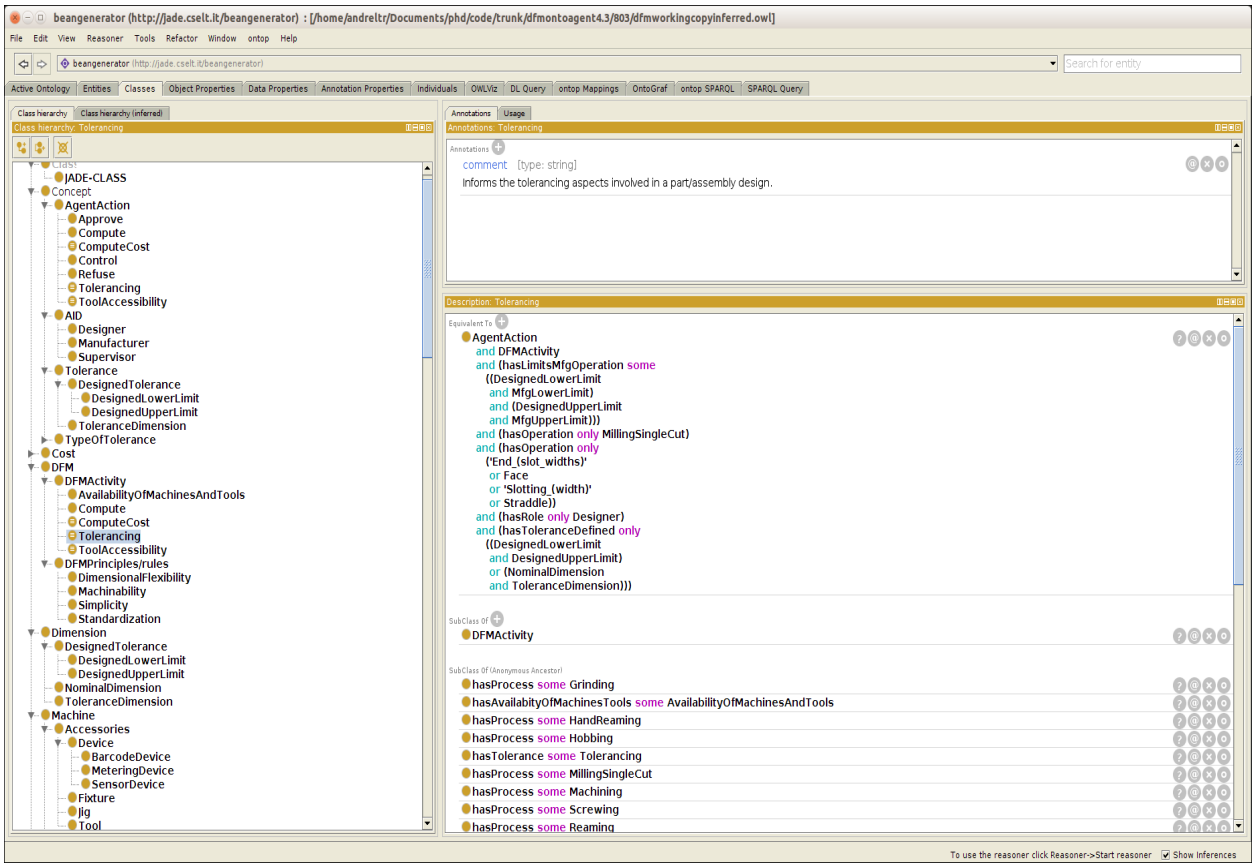


Figura 4.36 – Taxonomia definida relacionada ao contexto DFM (parte).

ou disjunção, entre conceitos ontológicos (classes e operadores adicionais, tais como *slots*, *facets* e instâncias).

Efetivamente, esta característica permite diferenciar a parte relacionada diretamente com a informação, como propriamente definida, de sua utilização. Isto é relevante para permitir que componentes DFM, no caso, sejam focados na resolução de seu problema específico ao invés da necessidade de embutir aspectos não relacionados com seu objetivo. A inexistência disto pode, inclusive, limitar a capacidade computacional de um componente. As atividades DFM-específicas estão nas tabelas a seguir. Faddoul, 2011 descreve explicitamente como associar raciocínio algébrico com Lógica Descritiva (DL), simplificando a compreensão mais efetiva e completa desta pesquisa.

As tabelas descritivas a seguir, que são relacionadas especificamente com DFM, são fundamentadas na sequência descrita na figura 4.34. Existe, portanto, um aspecto relacionado com a informação que pode ser resumido em duas partes: a hierarquização conceitual, que tradicionalmente representa dependência, e como utilizar os conceitos expressos, ou sua operacionalização, descrita por suas propriedades associadas.

Logo, é relevante descrever o significado de cada conceito, seja ele puro ou operacional, pois sem isto o próprio processo identificatório tornar-se-ia um limitador. Devido à quantidade relativamente elevada de identificadores de conceitos informacionais, o

Tabela 4.6 – Relações lógico-abstratas das atividades DFM-específicas do domínio desta pesquisa com classes definidas em **negrito**.

Atividade	Equivalência[18]	Superclasses[19]
AvailabilityOfMachinesAndTools	∃ hasPersonnel (Supervisor ∨ Manufacturer) ∃ hasMachineAccessory (Tool ∧ Fixture ∧ Jig) ∃ hasActivity (Activity ∨ Analysis) ∃ isRole (Supervisor ∨ Manufacturer)	Activity ∧ Analysis ∀ hasMachine Machine ∀ hasTool Tool Compute AgentAction
ToolAccessibility	∃ hasLimitsMfgOperation ((MfgUpperLimit ∧ **UpperLimit**) ∧ (MfgLowerLimit ∧ **LowerLimit**)) ∃ isRole (Designer ∨ Manufacturer)	Activity ∧ Analysis AgentAction Compute ∀ hasMfgProcessDefined (End ∨ Face ∨ Slotting ∨ Straddle) ∀ hasOperation Milling ∀ hasToleranceDefined ((MfgUpperLimit ∧ **UpperLimit**) ∨ (NominalDimension ∧ Tolerance)) ∃ hasToolAccessibility (GeometryFeature ∧ ManufacturingFeature) ∃ hasTypeOfFeature (GeometryFeature ∧ ManufacturingFeature)
Tolerancing	∃ hasLimitsMfgOperation ((MfgUpperLimit ∧ **UpperLimit**) ∧ (MfgLowerLimit ∧ **LowerLimit**))	Activity ∧ Analysis Compute AgentAction ∀ hasOperation (End ∨ Face ∨ Slotting ∨ Straddle) ∀ hasOperation Milling ∀ hasToleranceDefined ((**LowerLimit** ∧ **UpperLimit**) ∨ (NominalDimension ∧ Tolerance)) ∃ isRole (Designer ∨ Manufacturer)
ComputeCost	∃ hasCostDefined (InitialCost ∧ ProcessingCost) ∀ hasProcessingCostDefined ProcessingCost ∀ hasInitialCostDefined InitialCost ∃ hasPersonnel Personnel	Activity ∧ Analysis AgentAction

Tabela 4.7 – Relacionamentos disjuntos das relações lógicas abstratas das atividades DFM no domínio da pesquisa com classes definidas em **negrito**.

Atividade	Disjunta
AvailabilityOfMachinesAndTools	¬ **Tolerancing** ¬ **ToolAcessibility** ¬ **ComputeCost**
Tolerancing	¬ **ToolAccessibility** ¬ **AvailabilityOfMachinesAndTools**
ToolAccessibility **ComputeCost**	¬ **Tolerancing** ¬ **AvailabilityOfMachinesAndTools**

Apêndice B descreve em mais detalhes seus significados, os quais são subdivididos em duas partes: hierarquização conceitual, operacional e de indivíduos.

Levando em consideração o conceito OWA, a disjunção das atividades é descrita segundo a tabela 4.7.

Após a definição dos relacionamentos hierárquico-taxonômicos é relevante e necessário especificar como a informação DFM é descrita em termos de sua tipificação. A tipificação inclui duas partes: operacionalização de suas intra-relações, parcialmente expressa na tabela 4.6, e a especificação da descrição de como eventuais dados e informações são representados. Os membros perfazem o último nível de abstração da representação conceitual da informação. A tabela 4.8 descreve este tipo de inter-relacionamento taxonômico da tabela 4.6.

[18] A equivalência é uma condição *necessária e suficiente*.

[19] Uma superclasse denota uma condição *necessária*. O vocábulo superclasse possui uma amplitude maior que no tradicional conceito computacional.

Tabela 4.8 – Relações lógicas abstratas dos membros, ou tipos dos dados, DFM no domínio da pesquisa.

Atividade	Membros (identificadores do tipo do *dado*)
AvailabilityOfMachinesAndTools	AgentAction
ToolAccessibility	Axis
	AgentAction
Tolerancing	AgentAction
	millimiter
ComputeCost	AgentAction
	Real

2. **Análise** DFM é, essencialmente, uma atividade de análise seja comparação, conformidade ou verificação. Desta forma, as atividades relacionadas com análise perfazem, em termos conceituais da informação, uma classe. As classes, ou conceitos, pertencem a diferentes categorias: análise e DFM concomitantemente. Este aspecto, dentre outros, efetivamente limita e sobrecarrega a parte teórico-sistêmica e implementacional em soluções tradicionais além de ocorrer analogamente em outras classes, de forma geral, nesta pesquisa. Portanto, a análise demanda uma arquitetura *operacional* eficiente e efetiva, como a fundamentada em agentes informada previamente, é descrita na tabela 4.9[20,21].

Tabela 4.9 – Taxonomia operacional DFM com classes definidas em **negrito**. Todas classes são subclasses da classe Conceito, ou Concept.

AID	*AgentAction*	Subclasses
Designer	Compute	**ToolAccessibility**
		Tolerancing
		ComputeCost
Manufacturer	Compute	**ToolAccessibility**
		ComputeCost
		AvailabilityOfMachinesAndTools
Supervisor	Approve, Refuse	**ToolAccessibility**
		Tolerancing
	Compute	**AvailabilityOfMachinesAndTools**
		ComputeCost

Tradicionalmente, DFM é um conceito intrinsecamente relacionado com regras e/ou princípios que o definem de forma menos abstrata o vocábulo *per-se*. Estas regras/princípios são que embutem o conhecimento, humano ou não, associado com um conceito. As regras/princípios escolhidos nesta pesquisa, a seguir, procuram descrever efetivamente o escopo do domínio. Analogamente à generalidade, quando ontologias são utilizadas estes conceitos possuem classe(s) disjuntas, que são tópicos

[20] *Agent IDentification* - são os agentes das atividades de projeto. Neste caso, analogamente, equivalem às atividades dos humanos envolvidos.

[21] Uma ação de um agente corresponde, neste caso, às atividades efetuadas por humanos.

Tabela 4.10a – Classes de princípio, ou regra, DFM–específico Flexibilidade Dimensional.

Atividade	Equivalência	Superclasses
DimensionalFlexibility		Principles/rules ∃ hasDimension (NominalDimension ∧ Tolerance) ∃ hasFeature Feature

Tabela 4.10b – Membros do princípio/regra DFM Flexibilidade Dimensional.

Princípios/regras	Membros
DimensionalFlexibility	millimiter

logicamente distintos em termos vocabulares. Especificamente, elas estão brevemente descritas a seguir, pois seus detalhes serão expressos adiante neste documento.

- Flexibilidade Dimensional - esta é uma regra que relaciona a dimensão nominal com a tolerância bem como o quão intrínseca esta união é com sólidos característicos (*features*). Considerando que o objeto desta pesquisa é DFM, por consequência existe a necessidade de integrar aspectos de fabricação como, por exemplo, maquinário e seus limites.

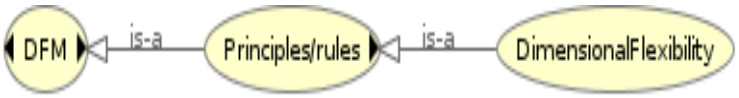


Figura 4.37 – Regra DFM: Flexibilidade Dimensional definida.

- Usinabilidade - descreve as relações entre conceitos de fresagem, desde o processo *per se*, seus componentes até suas inter-relações com outro conceito, que é resultante deste processo. No caso desta pesquisa e seu escopo, portanto, são os sólidos característicos, tolerâncias, limites do maquinário com seus acessórios e materiais.

Figura 4.38 – Regra DFM: Usinabilidade definida.

Tabela 4.11 – Classes de princípio, ou regra, DFM–específico: Usinabilidade.

Atividade	Equivalência	Superclasses
Machinability	Tolerance ∧ TypeOfTolerance	Principles/rules ∀ hasMachine Milling ∀ hasMachineAccessory (Fixture ∧ Jig ∧ Tool) ∀ hasTypeOfToleranceDefined TypeOfTolerance

Tabela 4.12a – Classes e propriedades de informação herdada de Usinabilidade.

Atividade	Classe, Propriedade herdada
Machinability	∃ hasToleranceDefined ((**LowerLimit** ∧ **UpperLimit**) ∧ (NominalDimension ∧ Tolerance)) ∃ hasTypeOfToleranceDefined **LocationTolerance** ∨ **OrientationTolerance** ∨ **ProfileTolerance** ∨ **Run-outTolerance** ∨ **ShapeTolerance** ∨ **SizeTolerance**)

Tabela 4.12b – Membros do princípio/regra DFM Usinabilidade.

Princípios/regras	Membros
Machinability	millimiter

- Padronização - considerando tanto aspectos puramente produtivos e de manufatura quanto mercadológicos, padronizar produtos o máximo possível, e viável, bem como processos de manufatura é uma demanda obrigatória se qualquer empresa que desejar continuar a ser competitiva no mercado atual. DFM, portanto, impõe indiretamente que padronização seja utilizada. Em geral, padronização, na sua forma mais simples, refere-se a obedecer um conjunto de regras organizacionais/corporativas que, em geral, tem por foco principal a redução do custo operacional, seja de manufatura, energia, segurança ou humano.

  Em DFM, padronização é relacionada com todos aspectos de projeto associados com produção de produtos. Por exemplo, a implementação da padronização no escopo específico de fabricação desta pesquisa demanda que o maquinário, ferramental e materiais sejam coerentes entre si e tecnicamente viáveis. Além do suporte de infraestrutura de fabricação associado com aspectos produtivos de qualidade, existe necessidade que ele seja compatível com um projeto específico em questão. Tradicionalmente, por exemplo, além dos materiais utilizados, existe o requisito que a infraestrutura seja capaz de fabricar, produtos segundo determinados requisitos de tolerâncias dimensionais e com determinado acabamento desejado.

  A padronização não é um processo simples de realizar efetivamente, porém atualmente e, na verdade, há mais de três décadas, é uma demanda imposta sobre as empresas de forma a se manterem competitivas no mercado. Sua efetivação e eficácia impõe comprometimento de todos os aspectos de uma organização: colaboradores, infraestrutura técnica e operacional, além dos fornecedores e suporte pós-comercialização. Portanto, para padronização efetiva de um processo para um dado fim, ele deve ser definido e implementado.

  Considerando o foco desta pesquisa, apenas ligações entre os conceitos de informação do escopo no domínio são feitas. Portanto, a padronização conside-

rada refere-se apenas ao Material e suas propriedades pois ele é, geralmente, a primeira etapa e o guia inicial de qualquer projeto.

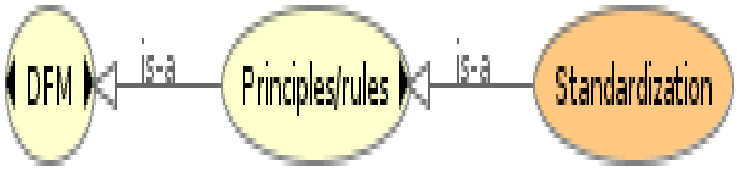


Figura 4.39 – Regra DFM: Padronização definida.

Tabela 4.13a – Classes de princípio, ou regra, DFM–específico: Padronização - foco em Materiais.

Atividade	Equivalência	Superclasses
Standardization	∃ hasMaterialProperty (Appearance ∧ CorrorionResistance ∧ ElectricalConductivity ∧ Formability ∧ Stiffness ∧ Strength ∧ SurfaceRoughness) ∃ hasMachineAccessory (Tool ∧ Fixture ∧ Jig)	Principles/rules ∀ hasMaterial Material *Material* *Product*

- Simplicidade - apesar de sua relativa obviedade conceitual, a simplicidade é provavelmente o mais abstrato destes quatro conceitos e, portanto, o que demanda maior complexidade de definição devido ao número de interconexões conceituais existentes, as quais abrangem todos os aspectos neste domínio. Desta forma, ele é expresso apenas como uma direção eventual de pesquisa fundamentada neste trabalho a ser desenvolvida posteriormente.

  Como um exemplo de simplicidade, é possível definir que determinados materiais com características específicas demandam sua fabricação em máquinas específicas objetivando um processo otimizado segundo determinada demanda. Analogamente, algumas *features* impõem que apenas ferramental e maquinário de certa categoria seja utilizado para obter a funcionalidade projetada. As duas descrições acima mostram claramente como as regras DFM de simplicidade variam segundo o objetivo do processo e englobam aspectos informacionais distintos que, em geral, são efetiva e eficazmente integrados através da utilização de ontologias.

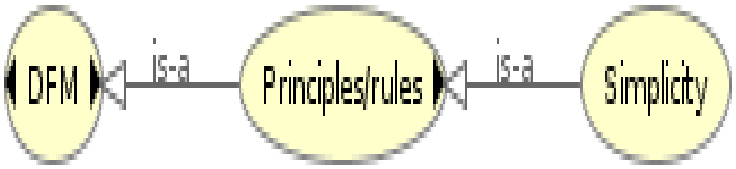


Figura 4.40 – Regra DFM: Simplicidade definida.

Tabela 4.14 – Classes de princípio, ou regra, DFM–específico Simplicidade.

Atividade	Equivalência	Superclasses
Simplicity	∃ hasTypeOfFeature (GeometryFeature ∧ ManufacturingFeature) ∃ hasLimitsMfgOperation (Dimension ∧ Milling ∧ MfgLimit)	Principles/rules

3. ∀ **Processo Manufatura**[23] Um processo de manufatura é composto por operações de fabricação bastante variáveis que, em geral, são díspares. Nesta pesquisa, sem perda de generalidade, o domínio é restrito à usinagem de peças poliédricas metálicas, conforme previamente expresso. As operações são, portanto, restritas às principais operações de fresamento, tais como de topo, faceamento, etc. e são sub-conceitos de um processo genérico, expresso na tabela 4.15 a seguir.

Tabela 4.15 – Relações lógicas abstratas do conceito DFM do Processo.

Atividade	Equivalência	Superclasses
Process	MIN hasOperation 1 **Operation**	Thing
Operation	MIN hasOperation 1 (Milling ∨ Drilling ∨ Turning)	∀ hasOperationType (Primary ∨ Secondary) **Process**

4. **Máquina** Obviamente, um material é transformado em um produto através de um determinado maquinário. Entretanto, por princípio, a informação relacionada com uma máquina possui três partes nos aspectos relacionados com informação: equipamento, seus acessórios (ferramental e elementos de fixação) bem como suas inter-relações. Esta característica específica diferencia uma estrutura ontológica de abordagens tradicionais pois possibilita expressar claramente o significado correto da informação.

   No caso deste conceito específico, por exemplo, é possível inter relacionar máquinas com princípios de análise DFM, conforme descrito na tabela 4.16a, que é relacionada com classes e propriedades específicas de máquina[24].

[23] Esta expressão equivale logicamente a uma asserção na qual todos os processos são de manufatura. Fundamentos lógicos de OWL estão disponíveis no Apêndice A.

[24] A indentação, que existe em outras tabelas, analogamente, é espressa apenas para retratar conceitos herdados adicionais de, no caso, Machine, ou seja, uma simplificação gráfica, somente, de relacionamentos de hereditariedade.

Tabela 4.16a – Relações lógicas abstratas do conceito DFM de Máquina e classes DFM–específicas: Máquina.

Atividade	Equivalência	Superclasses
Machine		Thing ∃ hasMachine (MillingMachine ∨ TurningMachine ∨ DrillingMachine) ∀ hasMachineAccessory (Tool ∧ Jig ∧ Fixture) ∃ hasManufacturingToleranceDefined (LowerLimit ∧ UpperLimit)
Tool[24]	∀ hasMachineAccessory **Tool**	Machine
Jig[24]	∀ hasMachineAccessory **Jig**	Machine
Fixture[24]	∀ hasMachineAccessory **Fixture**	Machine
MillingMachine[24]	∀ hasMachineAccessory (**Tool** ∧ **Jig** ∧ **Fixture**)	Machine
TurningMachine[24]	∀ hasMachineAccessory (**Tool** ∧ **Jig** ∧ **Fixture**)	Machine
DrillingMachine[24]	∀ hasMachineAccessory (**Tool** ∧ **Jig** ∧ **Fixture**)	Machine
MfgLimit[24]		Machine **Operation** ∀ hasMaterial Material ∃ hasMaterialProperty (Appearance ∧ CorrosionResistance ∧ ElectricalConductivity ∧ Formability ∧ Stiffness ∧ Strength ∧ SurfaceRoughness) ∃ hasToleranceDefined (LowerLimit ∨ UpperLimit) ∀ *hasOperationType (Primary ∨ Secondary)* MIN *hasOperation 1 (Milling ∨ Turning ∨ Drilling)* ∀ *hasMachineAccessory (Fixture ∨ Jig ∨ Tool)* ∃ *hasManufacturingToleranceDefined (LowerLimit ∧ UpperLimit)* ∃ *hasMachine (MillingMachine ∨ TurningMachine ∨ DrillingMachine)* MIN *hasOperation 1 Operation*
MfgLowerLimit[24]		MfgLimit e todas classes herdadas de Machine e MfgLimit
MfgUpperLimit[24]		MfgLimit e todas classes herdadas de Machine e MfgLimit

Tabela 4.16b – Membros do conceito Máquina.

Conceito	Membros
Machine	DFMIndex

5. **Operação** Sob o contexto de informação, produção e manufatura é necessário definir mais além do que o processo, por exemplo seu tipo em um plano de processo ou as suas limitações intrínsecas relacionadas com o maquinário. A tabela 4.17a descreve conceitos informacionais relacionados com atividades de fresagem, escopo desta pesquisa. Os detalhes sobre operações de Torneamento e Furação são análogos com Fresagem.

   Também é relevante especificar um aspecto ressaltado desde a descrição das atividades do domínio, ou seja, o requisito de "operacionalização" da informação. A "operacionalização" perfaz a descrição de como utilizar um conceito que, na verdade, permite definir como um conceito (informação) se inter-relaciona com outro(s). Em resumo, portanto, para utilização de forma completa dos conceitos de um domínio são necessários dois aspectos: hierarquização e inter-relacionamentos. A seção 4.3.3.3.3 descreve as relações cruzadas entre conceitos do domínio.

6. **Produto** Um produto é composto por uma, ou várias, peças. Quando um produto possui diversas peças ele é uma montagem. Cada peça possui descrições geométricas

[24]Componentes herdados adicionais de Machine.
[25]Componentes herdados em *itálico*.

Tabela 4.17a – Classes DFM–específicas: Operação.

Conceito	Equivalência	Superclasses[25]
Operation	MIN hasOperation 1 (Milling ∨ Drilling ∨ Turning)	∀ hasOperationType (Primary ∨ Secondary) MIN *hasOperation 1 Operation* **Process** Thing
OperationType		**Operation**
MfgLimit		Machine Operation ∀ hasMaterial Material ∃ hasMaterialProperty (Appearance ∧ CorrosionResistance ∧ ElectricalConductivity ∧ Formability ∧ Stiffness ∧ Strength ∧ SurfaceRoughness)
Milling		**Operation** MIN hasOperation 1 Milling ∀ *hasOperationType (Primary ∨ Secondary)* MIN *hasOperation 1 (Milling ∨ Turning ∨ Drilling)* MIN *hasOperation 1 Operation*
End		Milling MIN hasOperation 1 End MIN *hasOperation 1 Milling* e componentes herdados de Milling
Face		Milling MIN hasOperation 1 Face MIN *hasOperation 1 Milling* e componentes herdados de Milling
Slotting		Milling MIN hasOperation 1 Slotting e componentes herdados de Milling
Stradle		Milling MIN hasOperation 1 Straddle e componentes herdados de Milling

Tabela 4.17b – Membros da atividade Operação.

Atividade	Membros
Operation	DFMIndex
OperationType	DFMIndex
Milling	DFMIndex
Turning	DFMIndex
Drilling	DFMIndex

representadas através de sólidos característicos (ou "*features*"). Com esta representação taxonômica genérica, é possível associar classes que representam mais especificamente geometria, suas dimensões, e seus detalhes de fabricação como, por exemplo, tolerâncias, que explicitam o conceito de modificação da parte geométrica devido à manufatura nesta pesquisa.

Os sólidos característicos serão manufaturados e, consequentemente, terão sólidos característicos resultantes deste processo. Estas "*features*" de manufatura possuem conceitos informacionais análogos às "*features*" geométricas logo também precisam ser incluídas na arquitetura de informação. Desta forma, existem tanto "*features*" geométricas quanto de manufatura. Esta estrutura permite, inclusive, a expressão taxonômica comum a diferentes conceitos que, tradicionalmente, são fonte de proble-

mas em projeto pois não são interligados em DFM. Novamente, este problema está relacionado com o foco no dado. A tabela 4.18 descreve a taxonomia do produto.

Tabela 4.18 – Classes DFM–específicas: Produto.

Atividade	Equivalência	Superclasses
Produto		Thing
Assembly	MIN hasAssembly 2 **Part**	**Produto**
Part	MIN hasFeature 1 Feature	**Produto** ∃ hasFeature Feature
Feature		**Part** ∃ *hasFeature Feature* MIN *hasFeature 1 Feature*
TypeOfFeature		Feature ∀ hasFeature (GeometryFeature ∨ ManufacturingFeature) ∃ *hasFeature Feature* MIN *hasFeature 1 Feature*
GeometryFeature[26]	(∃ hasDimension NominalDimension) ∧ (∀ hasTypeOfFeature GeometryFeature) ∧ ∀ (hasFeature (Prismatic ∨ Rotational)) ∧ EXACTLY hasFeature 1 Feature	TypeOfFeature ∀ *hasFeature (GeometryFeature ∨ ManufacturingFeature)* ∃ *hasFeature Feature* MIN *hasFeature 1 Feature*
Hole[27]	(∀ hasTypeOfFeature GeometryFeature) ∧ (∀ hasFeature **Hole**) ∧ (EXACTLY hasFeature 1 Feature)	**GeometryFeature** ∀ hasTypeOfFeature **Hole** (∃ *hasDimension NominalDimension*) ∧ (∀ *hasFeature (Prismatic ∨ Rotational)*) ∧ (∀ *hasTypeOfFeature GeometryFeature*) ∧ (EXACTLY *hasFeature 1 Feature*) ∀ *hasFeature (GeometryFeature ∧ ManufacturingFeature)* ∃ *hasFeature Feature* MIN *hasFeature 1 Feature*
Prismatic[27]	(∀ hasTypeOfFeature GeometryFeature) ∧ (∀ hasFeature **Prismatic**) ∧ (EXACTLY hasFeature 1 Feature)	**GeometryFeature** ∀ hasTypeOfFeature **Prismatic**
Rotational[27]	(∀ hasTypeOfFeature GeometryFeature) ∧ (∀ hasFeature **Rotational**) ∧ (EXACTLY hasFeature 1 Feature)	**GeometryFeature** ∀ hasTypeOfFeature **Rotational**
ManufacturingFeature[26]	(∀ hasTypeOfFeature ManufacturingFeature) ∧ (∃ hasToleranceDefined Tolerance) ∧ ∀ (hasFeature (**Hole** ∨ **Prismatic** ∨ **Rotational**)) ∧ EXACTLY hasFeature 1 Feature	TypeOfFeature ∀ *hasFeature (GeometryFeature ∨ ManufacturingFeature)* ∃ *hasFeature Feature* MIN *hasFeature 1 Feature*

Tabela 4.19 – Membros do conceito Produto.

Conceito	Membros
Produto	
Part	millimiter
Feature	millimiter
GeometryFeature	millimiter
Prismatic	millimiter
	Prismatic
Rotational	millimiter
	Rotational
Hole	millimiter
	Hole

7. **Dimensão** Uma dimensão é um conceito relacionado a outro conceito: **Produto**. Em termos de fabricação, as dimensões são alteradas devido às consequências de um processo de fabricação, tanto em termos térmicos quanto em termos da modificação de seu tamanho pelas operações. Consequentemente, a definição conceitual de uma dimensão é descrita, geralmente, em termos nominais e da tolerância, definida e aceitável, resultante de um processo.

   Conforme descritas no tópico de Operação, é um requisito a integração da dimensão e sua consequente fabricação em DFM. Ao mesmo tempo e considerando o expresso,

[26] Conceitos herdados de Feature e TypeOfFeature.

[27] Conceitos herdados de Feature, TypeOfFeature e GeometryFeature. A *feature* **Hole** é utilizada apenas como referência.

uma tolerância possui sub-conceitos heterogêneos intrínsecos tais como os limites físicos do maquinário, que são numerais, e o tipo de tolerância além de suas inter-relações com outros aspectos de DFM, por exemplo Usinabilidade (*Machinability*).

Neste caso, também fica claro que esta característica, ou propriedade (vide seção 4.3.3.3.3), também embute diferentes conceitos tanto na tolerância *per se* quanto em relação a seu tipo. Considerando somente esta diferença conceitual, por exemplo, ocorrerá um excesso óbvio, tanto em nível sistêmico-algorítmico, quanto em sua eventual implementação, se for utilizada a abordagem tradicional em relação à utilização e gerenciamento da informação. A tabela 4.20a descreve a taxonomia da tolerância.

Tabela 4.20a – Classes DFM–específicas: Dimensão.

Atividade	Equivalência	Superclasses
Dimension		Thing
NominalDimension		Dimension
GeometryFeature	NominalDimension ∧ TypeOfFeature ∧ ∀ hasFeature (Prismatic ∨ Rotational) ∧ EXACTLY hasFeature 1 Feature	∀ *hasDimension NominalDimension* ∀ *hasFeature (GeometryFeature ∨ ManufacturingFeature)* ∃ *hasFeature Feature* MIN *hasFeature 1 Feature*
Tolerance	∃ hasToleranceDefined ((**LowerLimit** ∧ **UpperLimit**) ∧ (NominalDimension ∧ **Tolerance**))	Dimension
TypeOfTolerance		**Tolerance** ∃ hasTypeOfToleranceDefined (**LocationTolerance** ∨ **OrientationTolerance** ∨ **ProfileTolerance** ∨ **Run-outTolerance** ∨ **ShapeTolerance** ∨ **SizeTolerance**)
LocationTolerance[28]	∀ hasTypeOfToleranceDefined **LocationTolerance**	TypeOfTolerance ∃ *hasToleranceDefined ((**LowerLimit** ∧ **UpperLimit**) ∧ (NominalDimension ∧ Tolerance)*) ∃ hasTypeOfToleranceDefined (***LocationTolerance*** ∨ ***OrientationTolerance*** ∨ ***ProfileTolerance*** ∨ ***Run-outTolerance*** ∨ ***ShapeTolerance*** ∨ ***SizeTolerance***)
OrientationTolerance[28]	∀ hasTypeOfToleranceDefined **OrientationTolerance**	TypeOfTolerance
ProfileTolerance[28]	∀ hasTypeOfToleranceDefined **ProfileTolerance**	TypeOfTolerance
Run-outTolerance[28]	∀ hasTypeOfToleranceDefined **Run-outTolerance**	TypeOfTolerance
ShapeTolerance[28]	∀ hasTypeOfToleranceDefined **ShapeTolerance**	TypeOfTolerance
SizeTolerance[28]	∀ hasTypeOfToleranceDefined **SizeTolerance**	TypeOfTolerance
UpperLimit		
LowerLimit		
Machinability		

Tabela 4.20b – Membros da atividade Dimensão.

Atividade	Membros
Dimension	DFMIndex
	millimiter

8. **Material** um produto, por princípio, é composto de um material, que é variável em função de sua forma e composição físico–química. Estas caraterísticas influirão ou, até inclusive, podem determinar seu modo de fabricação. Portanto, é necessária a disponibilização de aspectos relacionados, ou intrínsecos, que definam um material

9. **Pessoal** em qualquer organização o papel dos colaboradores é fundamental. Em um fábrica, tipicamente, eles são subdivididos por setores ou áreas correlatas com

[28]Os membros herdados são os mesmos; **LocationTolerance** é descrita apenas como referência.

Tabela 4.21a – Classes DFM–específicas: Material.

Atividade	Equivalência	Superclasses
Material		Thing
MaterialProperty		Material
Appearance	MaterialProperty ∧ (∀ hasProperty Appearance)	
CorrosionResistance	MaterialProperty ∧ (∀ hasProperty **CorrosionResistance**)	
ElectricalConductivity	MaterialProperty ∧ (∀ hasProperty **ElectricalConductivity**)	
Formability	MaterialProperty ∧ (∀ hasProperty **Formability**)	
Stiffness	MaterialProperty ∧ (∀ hasProperty **Stiffness**)	
Strength	MaterialProperty ∧ (∀ hasProperty **Strength**)	
SurfaceRoughness	MaterialProperty ∧ (∀ hasProperty **SurfaceRoughness**)	

Tabela 4.21b – Membros da atividade Material.

Atividade	Membros
Material	DFMIndex

sua especialidade[29]. Portanto, nesta pesquisa a informação sobre os colaboradores é subdividida em três papéis, ou "*roles*": projetista, responsável pela manufatura e supervisor, sem perda de generalidade.

Seu relacionamento informacional é descrito na tabela 4.22a e pode, inclusive, ser alterado para que esteja conforme a dinamicidade do processo de fabricação. Esta situação, que é habilitada apenas pela utilização de ontologias, é analogamente válida para todos outros conceitos (classes e seus componentes) desta pesquisa. Isto possibilita que a definição do dado seja feita conforme sua instanciação seja requerida.

Tabela 4.22a – Classes DFM–específicas: Pessoal.

Atividade	Equivalência	Superclasses
Personnel		Thing
Designer	∃ isRole **Designer**	AID ∧ Compute ∧ Personnel ∃ hasActivity (Tolerancing ∨ ToolAccessibility)
Manufacturer	∃ isRole **Manufacturer**	AID ∧ Compute ∧ Personnel ∃ hasActivity (AvailabilityOfMachinesAndTools ∨ ComputeCost ∨ ToolAccessibility)
Supervisor	∃ isRole **Supervisor**	AID ∧ (Approve ∨ Refuse) ∧ Personnel ∃ hasActivity (AvailabilityOfMachinesAndTools ∨ ComputeCost)

Tabela 4.22b – Membros do conceito Pessoal.

Conceito	Membros
Designer	AgentAction

10. **Custo**: o custo é, obviamente, uma parte relevante e intrinsecamente associada com o sistema capitalista vigente, portanto demanda que sua utilização exista em uma

[29] É relevante destacar que, entretanto, de forma crescente e sob certo aspecto até vertiginosa, existe o requisito que um colaborador tenha capacidade de atuar de forma multitarefa nas empresas devido a razões, inclusive mais relevantes, como, por exemplo: flexibilidade e qualificação para efetuar novas tarefas oriundas do progresso tecnológico, aliadas à competição existente no sistema capitalista.

arquitetura de informação para que ela seja viável sob o aspecto de sua utilização prática. Embora existam diversos aspectos que demandam custo sendo considerados, para uma análise completa sob o foco financeiro, esta pesquisa é restrita a demonstrar o quão relevante é a inserção deste tópico numa arquitetura de informação DFM. Portanto, apenas três aspectos iniciais, abaixo, são considerados. As tabelas 4.23a, 4.23b, 4.23c, 4.23e e 4.23f descrevem a taxonomia do custo do domínio DFM desta pesquisa.

Investimento inicial, ou de capital é o investimento necessário para capacitar uma organização à produção, ou fabricação, de determinado bem.

Investimento de processo, ou fabricação é o investimento necessário para operacionalizar um determinado sistema; pode variar desde materiais e ferramental até demanda de energia e custos associados com viabilidade ambiental, bem como custos com a posterior reciclagem dos produtos fabricados.

Investimento em mão de obra a mão de obra é o fundamento operacional básico de qualquer organização, portanto seu custo, obrigatoriamente, precisa ser incluído. A informação sobre o custo de mão de obra nesta pesquisa é restrita a três funções especificamente relacionadas com DFM: Projetista (*Designer*), Operador (*Manufacturer*) e Supervisor.

Tabela 4.23a – Custo.

Atividade	Equivalência	Superclasses
Custo		**Thing**

Tabela 4.23b – Custo inicial, ou de investimento.

Equivalência	Superclasses
∃ hasInitialCostDefined Machine	Cost Machine

Tabela 4.23c – Custo de fabricação.

Equivalência	Superclasses
∃ hasProcessingCostDefined (**Tool** ∧ **Fixture** ∧ **Jig**) ∃ hasInitialCostDefined Machine	(MIN hasAssembly 2 Part) ∧ (MIN hasFeature 2 Feature) Cost **Tool** **Jig** **Fixture** ∀ *hasMachineAccessory Tool*[30] ∀ *hasMachineAccessory Fixture*[30] ∀ *hasMachineAccessory Jig*[30]

Tabela 4.23d – Custo de fabricação: propriedades de informação herdada.

Propriedade herdada
∀ hasMachineAccessory Tool
∀ hasMachineAccessory Jig
∀ hasMachineAccessory Fixture

Tabela 4.23e – Custo de mão de obra.

Equivalência	Superclasses
∃ hasPersonnelCostDefined (Designer ∧ Manufacturer ∧ Supervisor)	Cost

Tabela 4.23f – Membros do Custo.

Conceito	Membros
Custo	Real, $
InitialCost	Real, $
ProcessingCost	Real, $
PersonnelCost	Real, $

4.3.3.3.3 Detalhes, ou Propriedades, (*Slots*) das Classes

Conforme descrito brevemente no tópico **DFM**, existe necessidade de contextualizar a informação, ou seja, definir qual seu significado em relação aos outros conceitos expressos na ontologia. Esta contextualização, além de inter–relacionar os conceitos hierarquicamente expressos, permite que o real significado de um determinado aspecto seja compreendido e utilizado.

Apenas a hierarquização conceitual direta *não é suficiente* para definir a totalidade dos requisitos necessários para compreensão integral da informação. Assim sendo, também é necessário representar como conceitos são *utilizados* na taxonomia. Este tipo de representação permite obter e completar, quase totalmente, a descrição de um conceito. Esta descrição é efetuada através da definição de propriedades, descritas por *"slots"*, associadas às classes.

As propriedades possuem as mesmas características gerais das classes, porém "reduzem" sua abstração conceitual "pura". Esta redução é feita através da definição de domínios (*domain*) e faixas (*range*) de validade, ou não, de uma determinada classe. Os domínios e faixas, analogamente a classes, podem possuir disjunções levando em consideração o fundamento OWA.

[30] Conceitos herdados.

Este aspecto inclui a necessidade de definição de propriedades intrínsecas/inerentes aos conceitos expressos, permitindo operacionalizar sua utilização cardinal levando-os em consideração. Em resumo, é um contexto operacional prático que permite a definição da dinamicidade do processo DFM. As tabelas a seguir e a descrição do significado das propriedades (apêndice B) permitem que a taxonomia seja compreendida de forma mais íntegra.

As propriedades da tabelas 4.24a e 4.24b referem-se apenas a conceitos genéricos diretamente relacionados com um produto numa arquitetura DFM. Neste caso, é possível perceber claramente como a taxonomia operacional permite interpretar corretamente o significado de um conceito; basta analisar, por exemplo, a reificação[31] isAssembly → hasPart → hasFeature → hasTypeOfFeature. Esta reificação é, inclusive, bastante clara ao explicitar os detalhes conceituais em hasFeature → hasTypeOfFeature pois, a partir de uma *feature* que é apenas dimensionada, ou definida, torna–se possível interpretá-la através desta sua definição real. Esta definição engloba a parte geométrica bem como sua consequente manufatura através dois conceitos distintos: GeometryFeature e ManufacturingFeature.

Tabela 4.24a – Assembly.

Propriedade	Domínio	Faixas
isAssembly	Product	MIN isAssembly 2 Part ∀ isAssembly Part

Tabela 4.24b – Part.

Propriedade	Domínio	Faixas
hasPart	Product ∧ Part	**Part** ∀ isAssembly Part
hasFeature	Feature	∀ hasFeature (Feature ∧ Part)
	Part	∃ hasDimensionDefined Dimension
hasTypeOfFeature	Feature	Prismatic ∨ Rotational ∨ Hole
	Part	GeometryFeature ∧ ManufacturingFeature

Levando em consideração que DFM é um processo dinâmico, torna–se importante que sua informação possa representar este aspecto. Portanto, em termos do processo DFM desta pesquisa existem dois vocábulos, ou conceitos operacionais: isDFM (conceito intrínseco) e isDFMOf (conceito representando como uma determinada informação é obtida/relacionada). Adicionalmente, é relevante ressaltar que estas tabelas perfazem um conceito sinônimo, ou associado. As tabelas 4.25b e 4.26a descrevem apenas parte do conjunto completo de propriedades desta pesquisa, que não está descrito, pois o objetivo, nesta etapa, é sintetizar a abordagem utilizada. A tabela 4.25a descreve claramente o quão interligados são os conceitos e, consequentemente, suas informações embutidas em DFM.

[31] Reificação: concepção do indivíduo como objeto. Coisificação. Transformar um conceito abstrato em realidade concreta.

Tabela 4.25a – Propriedades associadas com o conceito de o que, efetivamente, perfaz DFM, ou seja isDFM, ou sua operacionalização: hasActivity e needsComponent.

Propriedade	Domínio	Faixas
hasActivity	DFM Activity ∧ Analysis	Activity ∨ Analysis AvailabilityOfMachinesAndTools ∨ ComputeCost ∨ Tolerancing ∨ ToolAccessibility

Tabela 4.25b – needsComponent.

Propriedade	Domínio	Faixas
needsComponent	DFM	Machine **Tool** **Jig** **Fixture** **AvailabilityOfMachinesAndTools** **Tolerancing** **ToolAccessibility** **ComputeCost** Compute Approve Refuse

Um aspecto adicional relevante da utilização contextual habilitada através de ontologias é sua capacidade para definir *referências cruzadas*, diretas ou inversas, na informação. Desta forma, por exemplo, é relativamente simples inter-relacionar detalhes sobre componentes, como na tabela 4.26a. No caso desta pesquisa, uma questão bastante típica é conhecer a origem do resultado de uma inferência, ou qual sua fundamentação.

Especificamente, a propriedade needsComponent, tabela 4.25b, possui sua inversa descrita na tabela 4.26b. Estas propriedades, como, em geral, qualquer propriedade, possuem relações lógicas com as categorias, ou classes. Desta forma, é possível obter uma definição clara e completa sobre como a informação é requerida e utilizada pelos componentes DFM. No caso do domínio desta pesquisa, por opção de projeto, existe uma sub-propriedade hasActivity que descreve a inter-relação entre componentes DFM, que é comum, em duas super-propriedades: isDFM (tabela 4.25a) e isDFMOf (tabela 4.26a) - fato que prova que os conceitos são sinônimos.

A redução da abstração da informação representada por classes através propriedades com domínios e faixas permite especificar uma abordagem para definir como obter a representação final de uma determinada unidade, ou dado. A capacidade de configuração da utilização de propriedades como, por exemplo, propriedade inversa permite definir uma determinada operacionalização da informação bem como sua origem. Além disto, existe a possibilidade de contextualizar conceitos de não facilmente representáveis, pois este tipo de semântica, enquanto real, é muito complexa de ser sistematizada em termos

computacionais, particularmente na área de projeto. Portanto, as propriedades a seguir são associadas com o conceito de o que, efetivamente, é a origem de um determinado processo DFM, ou seja isDFMOf, ou a origem de sua operacionalização.

Tabela 4.26a – hasActivity (isDFMOf).

Propriedade	Domínio	Faixas
hasActivity	DFM Activity ∧ Analysis	Activity ∨ Analysis AvailabilityOfMachinesAndTools ∨ ComputeCost ∨ Tolerancing ∨ ToolAccessibility

Tabela 4.26b – isNeededComponentOf.

Propriedade	Domínio	Faixas
isNeededComponentOf	DFM	**AvailabilityOfMachinesAndTools** **ComputeCost** **ToolAccessibility** **Tolerancing**

Tabela 4.26c – isCostDefinedBy.

Propriedade	Domínio	Faixas
isCostDefinedBy	DFM	Cost
	Activity	**InitialCost**
	xPrinciples/rules	**ProcessingCost**
	AvailabilityOfMachinesAndTools	**PersonnelCost**
	ComputeCost	∃ to_decimal real
	ToolAccessibility	∃ to_decimal double
	Tolerancing	∃ to_decimal decimal
	DimensionalFlexibility	∃ to_decimal float
	Machinability	
	Simplicity	
	Standardization	

Analogamente aos relacionamentos definidos com propriedades para os componentes mais genéricos, ou seja, que definem as atividades da arquitetura DFM, a mesma estratégia é utilizada para o restante da ontologia. A seguir são descritos exemplos para as propriedades de Tolerância (tabela 4.27a), hasToleranceDefined (tabela 4.27c), e Accessibilidade da ferramenta, hasToolAccessibility (tabela 4.28), respectivamente.

Após as propriedades estarem definidas a nível dos componentes da arquitetura, torna-se necessário, sob certo aspecto, associar DFM a um "quantificador global". Isto pode ser feito através da inserção de um super-conceito quantificador associado com DFM. Tipicamente, esta quantidade, que considera todos os componentes associados com este

Tabela 4.27a – isTolerancingFrom.

Propriedade	Domínio	Faixas
isTolerancingFrom	**Tolerancing**	Dimension NominalDimension **Tolerance** TypeOfTolerance ∀ hasToleranceDefined ((LowerLimit ∧ UpperLimit) ∨ (NominalDimension ∧ Tolerance))

Tabela 4.27b – isToolAccessibilityFrom.

Propriedade	Domínio	Faixas
isToolAccessibilityFrom	Feature	GeometryFeature ManufacturingFeature

Tabela 4.27c – Propriedade Tolerância definida.

Propriedade	Domínio	Faixas	Super propriedade
hasToleranceDefined	Dimension	NominalDimension ∧ Tolerance	isDFM

conceito, é heterogênea. Logo sua representação final é, igualmente, outro conceito. Porém, em geral, é possível associar a ela um valor percentual.

Neste caso é fundamental explicitar claramente a descrição do último nível da informação antes do dado: o indivíduo, conforme descrito na subseção a seguir. O conceito-objetivo desta pesquisa é DFM, logo através desta arquitetura de informação torna-se possível associar e representar um "quantificador": DFMindex. Uma propriedade deste conceito que pode ser utilizada como, eventualmente, inclusive uma "métrica".

Atrelamento dos dados através de Indivíduos

Com a definição dos relacionamentos intrínsecos à taxonomia, também é necessário descrever como seus conceitos abstratos, ou "puros", representam, e que, na verdade, precisam e inclusive demandam, o nível mais definido, ou "baixo", da arquitetura da informação, que é o dado. Esta representação é feita através da identificação de indivíduos. Os indivíduos são membros da taxonomia, assim como classes e propriedades. Um indivíduo é utilizado analogamente a classes e propriedades para definir o tipo de informação descrita. Estes relacionamentos são do tipo de descrição do dado, ou seja, seu valor é tipicamente expresso em OWL/DL: "value, hasValue" (∋).

Tabela 4.28 – Propriedade Acessibilidade da ferramenta.

Propriedade	Domínio	Faixas	Super propriedade	Propriedade inversa
hasToolAccessibility	Feature	**GeometryFeature** ManufacturingFeature	isDFM hasTooling	isToolAccessibilityFrom

Tabela 4.29 – Relações lógicas abstratas do princípio DFM global.

Atividade	Equivalência	Superclasses
DFM		Thing ∋ hasDFMRate DFMindex

Os dados são representações de instâncias de indivíduos. Por sua vez, os indivíduos finalizam a interligação entre as abstrações (classes), que representam conceitos "puros", e suas interconexões dadas por propriedades[32]. As tabelas 4.31 a 4.34 denotam apenas um subconjunto de todas tabelas especificamente relacionadas a atividades DFM (tabela 4.8), porém é possível fazer, de forma relativamente simples, analogias para todas classes e propriedades que contenham indivíduos.

A definição de como propriedades de *dados* são embutidas em *conceitos* é feita através de operadores, ou propriedades. A tabela 4.30 descreve como os indivíduos são definidos nesta pesquisa. É relevante observar como um indivíduo é integrado através de asserções de propriedades na ontologia. Levando em consideração que as propriedades dos dados descrevem instâncias de indivíduos, é possível a especificação de um dado com contextualização. A contextualização perfaz a compreensão do significado dos dados, a qual possibilita a otimização da sua utilização, particularmente em ambientes heterogêneos como DFM.

Adicionalmente, é relevante descrever que toda taxonomia de conceitos, atividades, propriedades e indivíduos DFM é definida juntamente com sua inserção numa arquitetura operacional de agentes de *software* JADE [Bellifemine, Caire e Greenwood, 2007] através de duas classes: Class e Concept. A classe Class é relacionada especificamente com a plataforma JADE e possui apenas uma subclasse: JADE-CLASS. A classe Concept é subdividida em duas categorias: AID[32] (identificadores de agentes) e AgentAction[33] (atividades dos agentes); estas atividades podem ser relacionadas com agentes direta ou indiretamente (analogamente a uma subrotina em sistemas tradicionais).

As tabelas 4.35, 4.36 [35] (páginas 131 e 132, respectivamente) descrevem como são

[32] As entidades são restritas a asserções específicas correntes, ou seja, a integração com STEP está restrita ao acesso à taxonomia OntoSTEP. A integração efetiva com STEP será feita *a posteriori* no prosseguimento da pesquisa pois as taxonomias desta pesquisa e do OntoSTEP necessitam ser adaptadas através, em geral, de relacionamentos definidos em propriedades de objetos e/ou dados; também pode ser possível o requisito de definir novos indivíduos. Tais relacionamentos serão a ligação entre as taxonomias. Além disto, dentro do âmbito ontológico mais amplo, estas ontologias podem, idealmente devem, ser integradas de fontes de informação diferentes.

[32] AID: *Agent IDentification* - são os agentes das atividades de projeto. Neste caso, analogamente, equivalem aos humanos envolvidos.

[33] Uma ação de um agente corresponde, neste caso, às atividades efetuadas (neste caso seja por componentes de projeto ou humanos).

[35] É relevante salientar que diversos indivíduos destas tabelas, especificamente a tabela 4.36, já foram expressos anteriormente e o são novamente apenas de forma a contextualizar adequadamente a abordagem de solução.

Tabela 4.30 – Definição dos indivíduos.

Indivíduo	Description ou Type	Property assertions
Concept	JADE-CLASS	
AID	JADE-CLASS	
AgentAction	JADE-CLASS	
	Approve	
	Compute	
	Refuse	
	Designer	
	Manufacturer	
	Supervisor	
	AvailabilityOfMachinesAndTools	
	ComputeCost	
	Tolerancing	
	ToolAccessibility	
Predicate	JADE-CLASS	
millimiter	Dimension	isToleranceDefinedBy millimiter
	DimensionalFlexibility	isTolerancingDefinedBy millimiter
	Feature	
	GeometryFeature	
	Hole	
	Part	
	Prismatic	
	Product	
	Rotational	
	Tolerancing	
$	**ComputeCost**	hasInitialCostDefined $ Real
	Cost	hasProcessingCostDefined $ Real
	InitialCost	hasCostDefined $ Real
	ProcessingCost	
	PersonnelCost	
	Approve	
	Compute	
	Refuse	
	Designer	
	Manufacturer	
	Supervisor	

Tabela 4.31 – Indivíduos DFM e seus relacionamentos.

Atividade	Indivíduos
AvailabilityOfMachinesAndTools	AgentAction
ToolAccessibility	Axis
	AgentAction
Tolerancing	AgentAction
	millimiter
ComputeCost	AgentAction
	Real

Tabela 4.32 – Indivíduos da taxonomia operacional DFM com classes definidas em **negrito**. Todas classes são subclasses da classe Conceito, ou Concept.

AID	*AgentAction*	Subclasses
Designer	Compute	**ToolAccessibility**
		Tolerancing
		ComputeCost
Manufacturer	Compute	**ToolAccessibility**
		ComputeCost
		AvailabilityOfMachinesAndTools
Supervisor	Approve, Refuse	**ToolAccessibility**
		Tolerancing
	Compute	**AvailabilityOfMachinesAndTools**
		ComputeCost

Tabela 4.33 – Classes de princípio, ou regra, DFM–específico Flexibilidade Dimensional.

Atividade	Equivalência	Superclasses
DimensionalFlexibility		Principles/rules
		∃ hasDimension (NominalDimension ∧ Tolerance)
		∃ hasFeature Feature

Tabela 4.34 – Membros do princípio/regra DFM Flexibilidade Dimensional.

Princípios/regras	Membros
DimensionalFlexibility	millimiter

interconectados os conceitos, propriedades e indivíduos neste tipo de arquitetura de informação. Esta abordagem integrada e globalizada da informação que, por sua vez, embute todas interconexões existentes desde o nível mais abstrato, que é o conceito "puro" relacionado com um dado, até o nível específico de representação de seu tipo é o que permite utilizar de forma efetiva, eficiente e válida uma determinada informação. A tabela 4.35 descreve, sob contexto genérico, como é parte da taxonomia considerando agentes, que é implementada através de JADE. Os agentes estão associados, analogamente, com os humanos envolvidos em projeto, enquanto suas atividades perfazem atividades associadas.

Uma vez que a estrutura taxonômica, ou seja, classes, propriedades e indivíduos é considerada definida, ou conforme as asserções consideradas pelo projetista de análise do domínio DFM define, torna-se relevante validá-las. A validação de uma taxonomia é realizada através de uma máquina de inferência, que definirá se a taxonomia é válida[36] bem como suas consequências lógicas, ou "conclusões".

[36]Neste caso, conceito de validade restringe-se apenas à validade lógica da ontologia.

Tabela 4.35 – Classes, propriedades, indivíduos e seus relacionamentos (componentes herdados em *itálico*).

Conceito/Atividade	Equivalência	Superclasses	Indivíduos
Class			
JADE-CLASS		Class	AgentAction AID Concept Predicate
Concept			
AID		Concept	
✓1 **Designer**	∃ isRole **Designer**	AID ∧ Compute ∧ Personnel ∃ hasActivity (Tolerancing ∨ ToolAccessibility) ∃ *isRole (**Designer** ∨ **Manufacturer**)*	AgentAction
✓2 **Manufacturer**	∃ isRole **Manufacturer**	AID ∧ Compute ∧ Personnel ∃ hasActivity (AvailabilityOfMachinesAndTools ∨ ToolAccessibility ∨ ComputeCost) ∃ *isRole (**Designer** ∨ **Manufacturer**)*	AgentAction
✓3 **Supervisor**	∃ isRole **Supervisor**	AID ∧ Approve ∧ Refuse ∧ Personnel ∃ hasActivity (AvailabilityOfMachinesAndTools ∨ ComputeCost) ∃ isRole Supervisor	AgentAction
AgentAction		Concept	
Approve	∃ isRole **Supervisor**	AgentAction	AgentAction
Supervisor	conforme ✓3		AgentAction
Compute	∃ isRole (**Designer** ∨ **Manufacturer**)	AgentAction	AgentAction
Designer	conforme ✓1		AgentAction
Manufacturer	conforme ✓2		AgentAction
Refuse	∃ isRole **Supervisor**	AgentAction	AgentAction
Supervisor	conforme ✓3		AgentAction

4.4 Informação de suporte DFM

DFM é uma atividade que engloba, na grande maioria, aspectos essencialmente contextuais que utilizam entradas, ou "*inputs*", que são resultados de outros processos e/ou métodos de cálculo. Isto é compreensível e pode ser inferido tanto nas atividades definidas acima quanto nas asserções inferidas, presentes no capítulo 5. Este tipo de informação é, neste trabalho, denominada como *informação-suporte* DFM. A informação-suporte, portanto, engloba aspectos diversos que são, muitas vezes, separados. Desta forma, nos objetivos deste documento, porém sem limitá-lo, apenas conceitos fundamentais no domínio e contexto são utilizados como *informação-suporte*. Analogamente, os outros componentes podem ser definidos.

No caso, os componentes fundamentais selecionados são Materiais e Dimensão. Adicionalmente, é relevante ressaltar que, embora seja um número teoricamente reduzido quantitativamente, tornar-se-á clara a flexibilidade, poder de abstração e completeza da utilização de ontologias conquanto analisando sua abrangência na situação corrente do domínio DFM. As classes de Materiais e Dimensão foram selecionadas pois descrevem de forma inequívoca o requisito pela integração das informações de aspectos díspares em DFM.

Fundamentalmente, devido ao domínio ser relacionado com projeto orientado à manufatura enquanto, em geral, projeto é baseado em "features" torna-se coerente utilizar um tipo distinto em uma classe adicional, porém análogo, à classe genérica de sólidos característicos subdividindo-a em duas subclasses: GeometryFeature e ManufacturingFea-

Tabela 4.36 – Classes, propriedades, indivíduos e seus relacionamentos. (continuação da tabela 4.35: apenas subclasses de AgentAction, que perfazem a segunda parte, e são atividades específicas de DFM são expressas; componentes herdados em *itálico*).

Conceito/Atividade	Equivalência	Superclasses	Indivíduos
Class			
Concept			
AID		Concept	
AgentAction		Concept	
AvailabilityOfMachinesAndTools	∃ hasMachineAccessory (Fixture ∧ Jig ∧ Tool) ∃ hasActivity (Activity ∨ Analysis) ∃ hasPersonnel (Manufacturer ∧ Supervisor) ∃ isRole (Manufacturer ∨ Supervisor)	AgentAction ∀ hasMachine Machine ∀ hasTooling Tool ∋ *hasDFMRate DFMindex*	AgentAction DFMindex
ComputeCost	∃ hasCostDefined (InitialCost ∧ ProcessingCost) ∃ hasProcessingCostDefined ProcessingCost ∃ hasInitialCostDefined InitialCost ∃ hasActivity (Activity ∨ Analysis) ∃ hasPersonnel Personnel	AgentAction Activity ∧ Analysis ∋ *hasDFMRate DFMindex*	AgentAction $ Real DFMindex
Tolerancing	∃ hasActivity (Activity ∨ Analysis) ∃ hasLimitsMfgOperation ((LowerLimit ∧ MfgLowerLimit) ∧ (MfgUpperLimit ∧ UpperLimit))	AgentAction Activity ∧ Analysis ∀ hasOperation (End ∨ Face ∨ Slotting ∨ Straddle) ∀ hasOperation Milling ∀ hasToleranceDefined ((LowerLimit ∧ UpperLimit) ∨ (NominalDimension ∧ Tolerance)) ∋ *hasDFMRate DFMindex*	AgentAction millimiter DFMindex
ToolAccessibility	∃ hasActivity (Activity ∨ Analysis) ∃ hasLimitsMfgOperation ((LowerLimit ∧ MfgLowerLimit) ∧ (MfgUpperLimit ∧ UpperLimit)) ∃ hasPersonnel (Designer ∨ Manufacturer) ∃ isRole (Designer ∨ Manufacturer)	Activity ∧ Analysis AgentAction ∀ hasOperation Milling ∀ hasMfgProcessDefined (End ∨ Face ∨ Slotting ∨ Straddle) ∀ hasToleranceDefined ((LowerLimit ∧ UpperLimit) ∨ (NominalDimension and Tolerance)) ∃ hasToolAccessibility (GeometryFeature ∧ ManufacturingFeature) ∃ hasTypeOfFeature (GeometryFeature ∧ ManufacturingFeature) ∋ *hasDFMRate DFMindex*	AgentAction Axis DFMindex

ture. Portanto, a classe GeometryFeature *per-se* possui inter-relacionamentos com aspectos puramente geométricos, conforme pode ser inferido na figura 4.41. A classe ManufacturingFeature possui inter-relacionamentos adicionais com aspectos de manufatura, como tolerância e maquinário.

Como exemplo, tipicamente, um foco relevante e importante em DFM é considerar o material que será utilizado para fabricar um produto e como o mesmo ou, neste caso, sua informação, interage no processo de projeto. Desta forma, as características das informações dos materiais são utilizadas como parte das ManufacturingFeatures, apesar de existirem tópicos adicionais que também influenciam. Obviamente, existem aspectos associados adicionais nesta classe, tais como as máquinas (Machine) e seus limites, processos (Process) e operações (Operation). Esta capacidade de integração entre informações é tradicionalmente considerada englobando aspectos díspares e de integração muito complexa, tanto em termos conceituais quanto computacionais. Através da taxonomização das informações, é demonstrada a grande flexibilidade e completeza adicional na representação da informação através da utilização de ontologias.

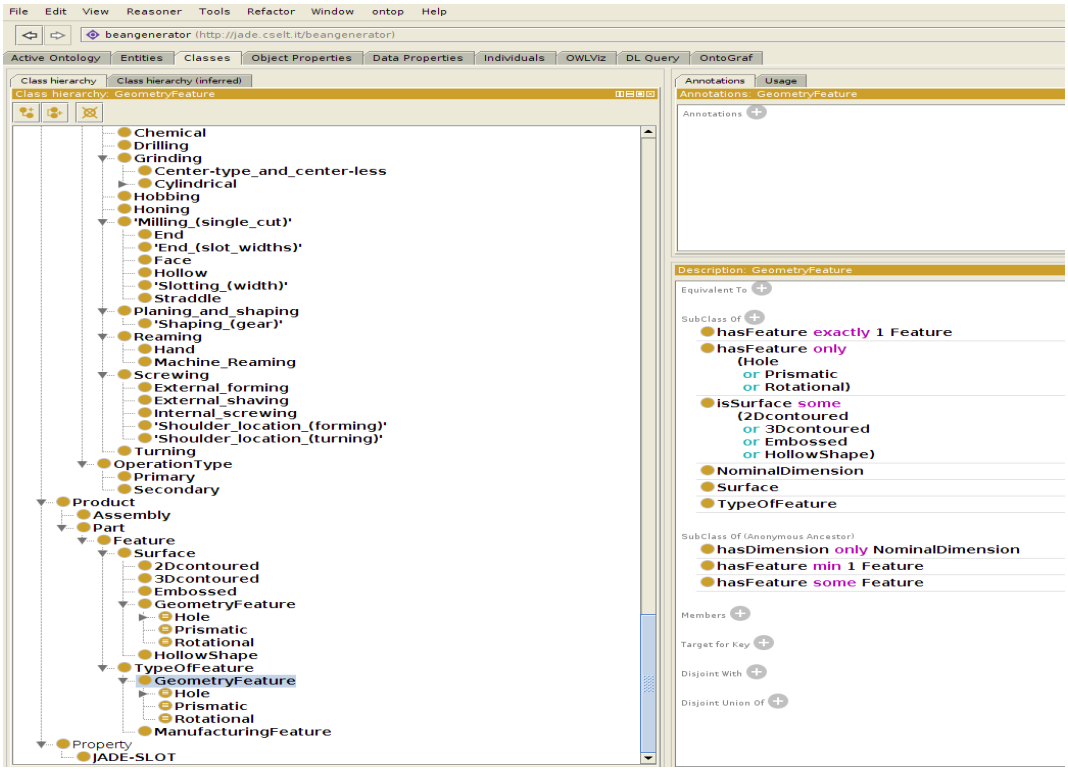


Figura 4.41 – GeometryFeature, ManufaturingFeature e sua integração com aspectos tradicionalmente díspares através de ontologias.

4.4.1 Materiais

Sob contexto de informação, que é o foco desta pesquisa, materiais, em geral, possuem suas características definidas em termos de descrições informacionais, ou seja,

aspectos que perfazem o *tipo do dado* (número, texto, sequência ou intervalo, por exemplo). Este tipo de definição em ontologias é feita através de propriedades de dados, que é o penúltimo nível conceitual antes dos dados *per-se*. Posteriormente, a descrição destas propriedades de dados é incluída nas classes associadas relevantes.

Considerando a grande quantidade de tabelas de características de materiais presentes na literatura, particularmente Bralla, 1999, apenas os tipos de dados (propriedades) presentes nelas são utilizados. As propriedades dos dados possuem, além da descrição informacional de seu tipo, associação com componentes taxonômicos (classes e/ou propriedades de objetos), conforme descrito na tabela 4.38. Logo, desta forma, é possível "reduzir" adicionalmente o nível abstração sobre como a informação DFM é representada. Abaixo, estão descritas as propriedades de tipos de dados de Materiais e como são representadas.

Tabela 4.37 – Hierarquia de Propriedades de Dados – Materiais

Propriedade de dado	Subpropriedade de	Domínio (*domain*)	Faixa (*range*)
is_array			
is_numeric_array	is_array	Precautions	double ∨ float
		Approximate_ratio	double ∨ float
		is_numeric_array ∀ (double ∨ float)	double ∨ float
is_string_array	is_array	Cutting_nature	string
		Expected_finish	string
		is_string_array ∀ string	string

4.4.2 Dimensão

Analogamente às propriedades de dados de Materiais, a propriedade de dados da classe Dimensão possui interconexão, ou descrição, sobre como estes dados são representados. Em geral, a representação dos dados é direta com uma primitiva: float ou double. Entretanto, considerando o domínio e seu escopo, é necessário considerar e analisar que o ideal é representar/conceitualizar uma característica intrínseca associada ao dado: tolerância. Uma dimensão nominal determinística resultante de um processo de manufatura é, na prática, impossível. Os tipos de dados restringem-se, portanto, neste caso, ao tipo de dado primitivo e seu consequente intervalo real.

Tabela 4.38 – Hierarquia de Propriedades de Dados – Dimensão

Propriedade de dado	Subpropriedade de	Domínio (*domain*)	Faixa (*range*)
is_float		Dimension	double ∨ float
is_tol_mfg_interval	is_float	Tolerance	double ∨ float
		UpperLimit ∧ LowerLimit	double ∨ float
		Dimension	double ∨ float
		DesignTolerance	double ∨ float
		is_numeric_array ∀ (double ∨ float)	double ∨ float

4.5 Metainformação no domínio DFM

Conforme expresso no início deste capítulo, a figura 4.32 descreve como a abstração aproxima o conhecimento[37] dos dados através da estrutura da informação. *Per se*, DFM é um conceito onde, de forma clara e inequívoca, existe um componente íntegro relacionado com conhecimento. Este componente de conhecimento está, por sua vez, em um nível de abstração superior à informação. Quanto mais próxima do conhecimento, a informação torna-se mais associada com o nível meta.

Se for levado em consideração o fato que a característica da informação possui um nível meta, torna-se, relativamente, claro em termos sistêmicos que os vocábulos sejam utilizados com anotações (descrições) de seu significado. A meta informação (ou meta representação), sob aspecto geral, não inclui asserções lógicas pois seu objetivo é uma descrição contextual do domínio em questão; no caso-objeto desta pesquisa DFM. A ligação entre o nível meta e a taxonomia é feita através de intersecção de classes relacionadas, que são elementos do domínio (*domain*), e de suas faixas (*range*) da informação, representados pelos tipos dos dados (*datatype*) nas anotações. Através desta associação entre o nível mais abstrato da informação, ou *meta*-nível, com a representação dos dados é possível, inequivocamente, compreender o significado da informação. Uma vez que esta ligação tenha sido definida, um determinado dado *per se* torna-se apenas uma instância de um conceito informacional.

A taxonomia hierárquica é utilizada como uma ligação entre a informação e o conhecimento, o qual é descrito através de um **vocabulário** representado através de propriedades de anotações (*Annotation Properties*). Em DFM, o vocabulário, portanto, é adicionalmente enriquecido conforme a figura 4.42, que descreve como é feita a associação entre o conhecimento e as informações[38].

Esta pesquisa utiliza um vocabulário meta pré-definido, denominado Dublin Core Metadata Initiative, 2014 (DCMI). Este vocabulário é customizado para utilização nesta pesquisa, seja através de anotações ou de inserção de vocábulos, os quais, inclusive, podem ser impossíveis de serem integrados. Conforme justificado acima, um vocabulário pré-definido é utilizado porém sua semântica necessita ser modificada redefinindo os óbvios diferentes significados dos vocábulos. Somente os vocábulos considerados relevantes, descritos na tabela 4.39, são utilizados.

Portanto, as definições de propriedades de anotações de dados perfazem, sob contexto sistêmico de processo de produção, aspectos relacionados com responsabilidades e atividades dos partícipes, que foram resumidos a três: Supervisor, Designer e Manufacturer. Estes componentes equivalem aos seres humanos envolvidos no processo

[37] Ressalta-se, entretanto, que conhecimento engloba e embute diversos aspectos adicionais; sua utilização nesta pesquisa restringe-se à parte descritiva em DFM.

[38] É relevante ressaltar que existem diferentes aspectos meta dos descritos nesta figura; ela é restrita às formas relacionadas com esta pesquisa.

de DFM e são sistemicamente representados como agentes na arquitetura JADE, reportada anteriormente, integrada com a ferramenta Protégé. Agentes possuem atividades, que são classes, que também perfazem as análises associadas com DFM nesta pesquisa: AvailabilityOfMachinesAndTools, ComputeCost, Tolerancing e ToolAccessibility.

Utilizando vocábulos do DCMI através de analogias como sinônimos torna-se possível descrever a taxonomia das propriedades de anotações. Elas funcionam como "descritores" abstratos do significado dos processos DFM envolvidos em projeto. Através desta abordagem torna-se possível representar o nível meta da informação.

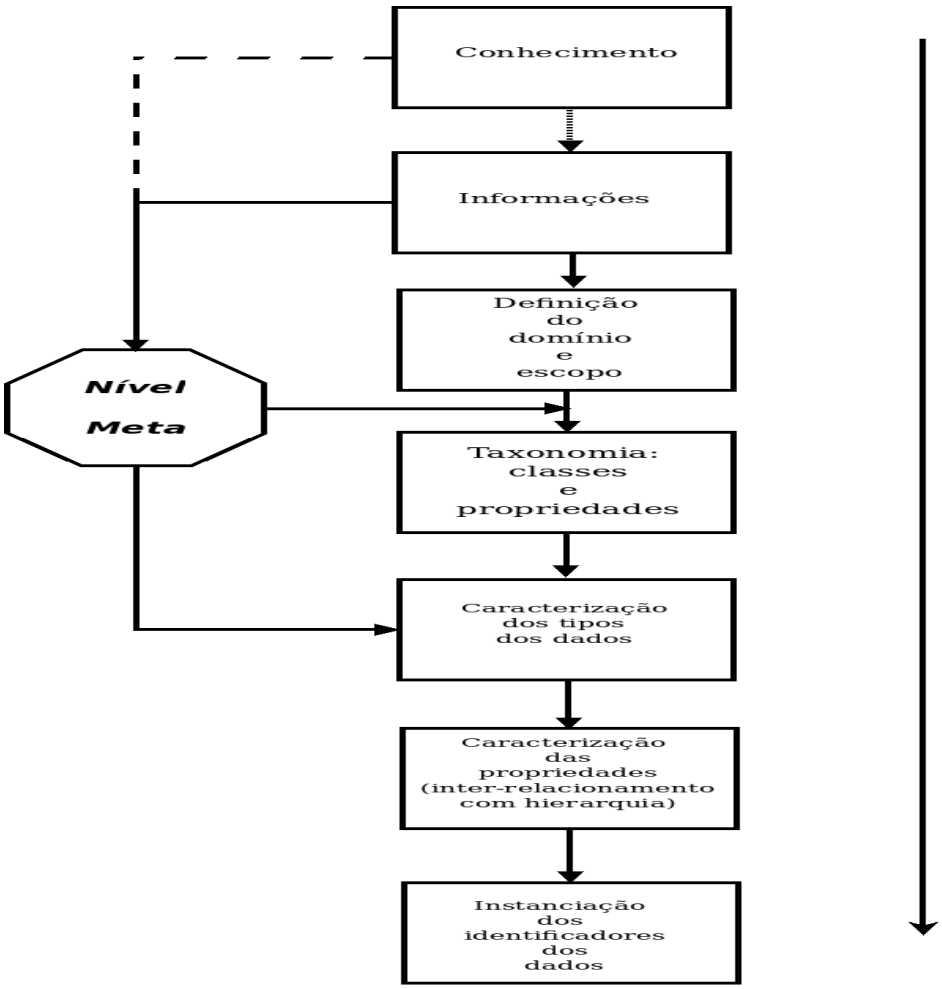


Figura 4.42 – Representação meta da informação e conhecimento através, no caso, do vocabulário.

Desta forma, conforme expresso acima, é possível, muitas vezes necessário, utilizar propriedades de anotações para representar aspectos meta que servem, fundamentalmente, para restringir a amplitude da inferência, bem como direcioná-la a contento do domínio e escopo de uma pesquisa. Em resumo, as anotações de propriedades que podem ser úteis para compreensão da parte operacional do arcabouço proposto. Utilizando DCMI genericamente e considerando as atividades DFM selecionadas (AvailabilityOfMachinesAndTools, ComputeCost, Tolerancing, ToolAccessibility) e os princípios DFM (DimensionalFlexibility,

Machinability, Simplicity, Standardization) é relativamente simples obter as meta–relações abaixo em um dos vocábulos DCMI: Agent [39] [40]. Reutilização e adaptação são partes intrínsecas dos fundamentos de ontologias. Assim sendo, a tabela 4.39 e a figura 4.43 demonstram como o vocabulário DCMI é utilizado neste arcabouço DFM.

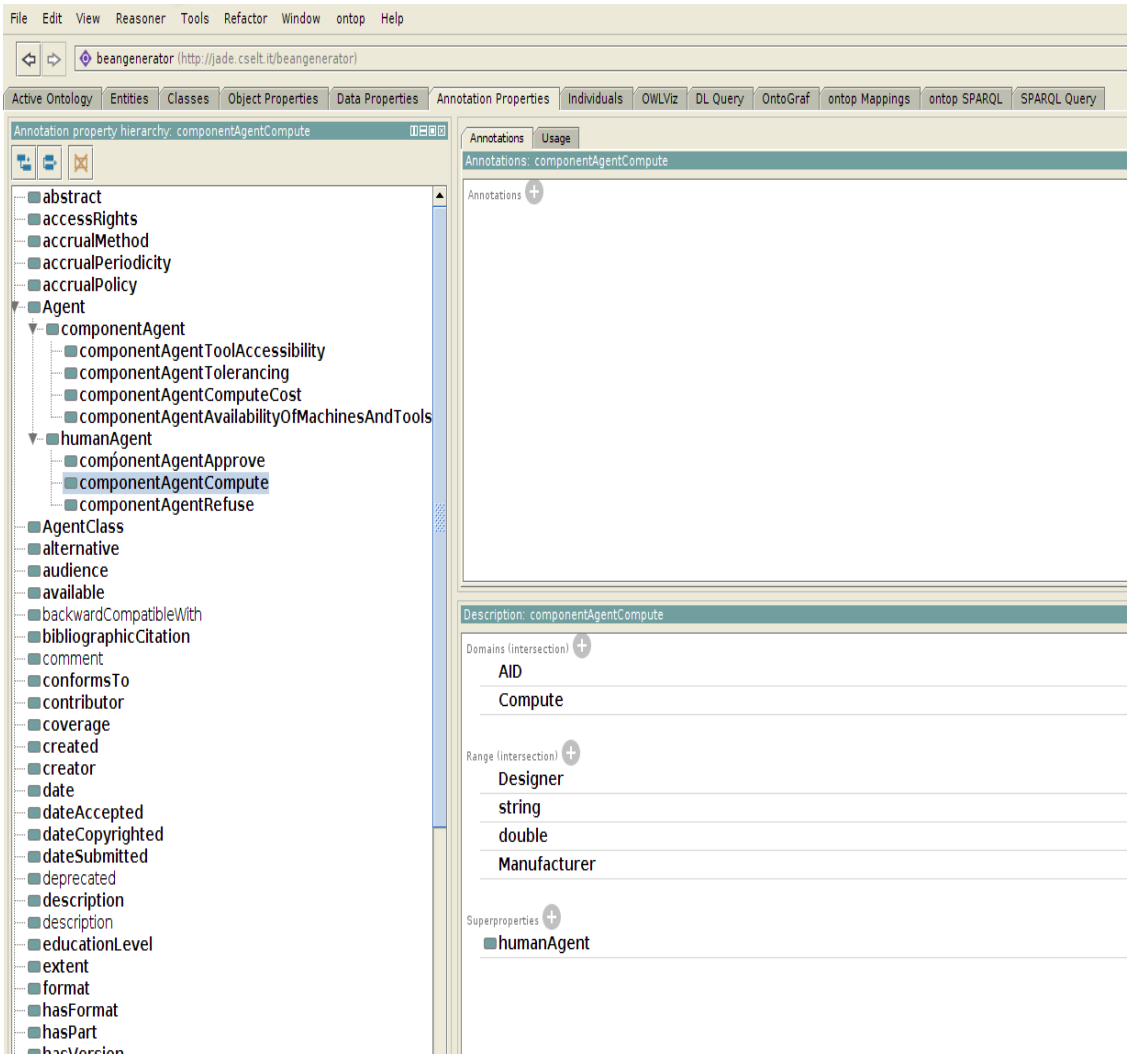


Figura 4.43 – Representação meta da informação e conhecimento no arcabouço utilizando agentes.

Após a estruturação do arcabouço, além de ser taxonomicamente estável, torna-se necessário verificar se as respostas fornecidas pela inferência resultam em uma intrumentalização taxonômica (parte operacional da informação) viável, além de consistente com os objetivos propostos. Os resultados da análise lógica da ontologia são, portanto, discutidos no capítulo 5.

[39] Esta é somente parte do conjunto de vocábulos utilizados como propriedades de anotações, pois vários outros vocábulos foram utilizados.

[40] Favor notar que as associações não são sempre necessariamente diretas entre linhas e colunas.

Tabela 4.39 – Propriedades de Anotações DFM Genéricas.

Propriedades de anotações	Domínio (*domain*) - interseção	Faixa (*range*) - interseção	Super-propriedades
Agent			Agent
componentAgent	AgentAction		Agent
componentAgentToolAccessibility	ToolAccessibility Manufacturer	string	componentAgent componentAgent
componentAgentTolerancing	Designer Tolerancing	double double	componentAgent componentAgent
componentAgentComputeCost	Designer Manufacturer ComputeCost	string string double	componentAgent componentAgent componentAgent
componentAgentAvailabilityOfMachinesAndTools	AvailabilityOfMachinesAndTools Manufacturer	string string	componentAgent componentAgent
humanAgent	AID		Agent
componentAgentCompute	AID Compute	Designer Manufacturer double string	humanAgent humanAgent humanAgent humanAgent
componentAgentRefuse	AID Refuse	Supervisor string	humanAgent humanAgent
componentAgentApprove	AID Approve	Supervisor string	humanAgent humanAgent

5 ANÁLISE DE CONSISTÊNCIA E COMPLETEZA DO ARCABOUÇO DE INFORMAÇÕES E CONHECIMENTO INTEGRADOS DE PROJETO ORIENTADO PARA MANUFATURA (DFM)

A taxonomia descrita no capítulo 4 explicita como o escopo do domínio proposto é definido, termos informacionais. Esta solução, entretanto, é uma das soluções viáveis e, ao mesmo tempo, conforme expressa, ela é somente um conjunto de asserções propostas não provadas, portanto sem validade lógica se não for considerada sua análise por uma máquina de inferência. Desta forma, torna-se um requisito validá-la de forma a garantir a consistência das relações e asserções definidas, ou seja, o modelo lógico de informação proposto. Esta validação é efetuada através de dois aspectos: depuração de erros reportados pelo motor de inferência, ou máquina de raciocínio, e análise da validade do modelo de informação após a inferência. De forma simplista, é necessário verificar se os resultados do raciocínio efetuado por um motor de inferência tornam consistentes entre as hipóteses descritas pelos componentes, operações e indivíduos definidos. As próximas seções, portanto, descrevem a consistência dos resultados obtidos a partir de um motor de inferência, no caso FACT++[1], a partir das definições do capítulo 4.

A validação, reitera-se, embute desde o aspecto hierárquico, seus inter-relacionamentos, ou propriedades, até as descrições dos dados. Salienta-se, novamente, que, segundo sua própria definição, uma ontologia não define o dado pois o objetivo é definir um significado de sua representação. Sob aspecto de informação, somente a partir de uma ontologia válida torna–se possível utilizar efetivamente instâncias, ou o real significado dos dados. Concomitantemente, é relevante ressaltar que esta ontologia, como padrão, obedece OWA, logo sua validação é restrita ao que foi definido, sem perda de generalidade sob contexto lógico.

As próximas seções descrevem os princípios DFM do capítulo anterior embutidos nesta ontologia. Os princípios DFM inferidos descritos demonstram, sem perda da generalidade, como existe amplitude e consistência no resultado da inferência. Portanto, conforme pesquisa em diversas áreas heterogêneas corretamente asserta, o verdadeiro significado da informação é obtido para a ontologia definida em análise.

Inicialmente, para este domínio e escopo, torna-se relevante descrever, genericamente, como o arcabouço de informação está embutido em agentes de *software* de forma a

[1]Existem diversos motores de raciocínio, gratuitos e comerciais, disponíveis; seus resultados são, em geral, similares porém existem detalhes teórico-computacionais que podem ocasionar resultados distintos de inferência. Este aspecto, entretanto, não está dentro do escopo desta pesquisa.

facilitar sua inserção em uma arquitetura computacional. Por princípio, portanto, operacionalmente a arquitetura segue este paradigma e, desta forma, define que seus componentes devem obedecer este *modus operandi*. Os agentes "emulam" seres humanos envolvidos em projeto, fabricação e análise no domínio DFM, enquanto as atividades que eles perfazem, ou atividades decorrentes de suas ações, são, em geral, componentes distintos na arquitetura. Consequentemente, existe integração entre a forma considerada mais flexível na dinamicidade deste domínio, além dela ser interligada à intrínseca contextualização da informação. No caso deste tipo de arquitetura existem duas características básicas: AID e AgentAction. Notar que a figura 5.44 é somente análoga à figura 4.35. Neste caso, ao invés de somente uma descrição arquitetônica hierárquica rígida das classes, a figura retrata um outro aspecto taxonômico: as responsabilidades dos agentes (AID) e suas ações (AgentAction), inclusive no relacionado com informações. Através desta abordagem, a arquitetura permite desconectar as informações requisitadas/utilizadas do algoritmo de solução de um dado problema.

Figura 5.44 – Arquitetura definida de informação fundamentada em agentes (complementação da figura 4.35 e como tal, devido ao ' *plug-in*" utilizado, não impõe equalidade).

Conforme a arquitetura definida de informação, com suas asserções, segundo o expresso visualmente na figura 5.44, é possível entender/visualizar claramente como as relações indiretas existentes conhecidas, porém "intrínsecas", podem ser representadas.

Estes relacionamentos são habilitados de forma viável, eficiente e efetiva somente através da representação fundamentada em ontologias. Se for, por exemplo, utilizada somente a abordagem tradicional centrada no dado ou, inclusive, um arcabouço baseado em STEP, que todavia não inclui este aspecto apesar de prevê-lo, uma solução eventual neste domínio seria incompleta em termos de informação.

Nesta pesquisa é interessante analisar a generalização demonstrada na figura 5.44 e verificar como ela mostra, de forma específica, o quão a arquitetura fundamentada em agentes permite a integração e inclusão de aspectos existentes em DFM. Dentre eles estão as atividades do escopo e as designações de tarefas aos responsáveis e suas responsabilidades, juntamente com a informação. Adicionalmente, é possível, e relevante, concluir que associações e inter-relações entre conceitos DFM formam, semanticamente, sinônimos. Neste sentido, sinônimos, que também são conceitos, demonstram analogamente as diferentes interpretações, ou asserções, sobre um conceito DFM. Enfim, os princípios/regras definidos de DFM no capítulo 4 são "ampliados"; as seções a seguir descrevem os resultados da inferência sobre eles. A inferência é validada sob contexto de problemas relacionados com a implementação da ontologia e sua consistência lógica.

5.1 Princípios/regras DFM do domínio e seu escopo

5.1.1 Flexibilidade Dimensional inferida

Flexibilidade dimensional é um princípio mandatório em DFM, pois qualquer processo de manufatura utilizado no escopo deste domínio, obrigatoriamente, ocasiona alterações dimensionais em uma peça. Estas alterações dimensionais, por sua vez, possuem consequências contextuais relacionadas com outros conceitos, que não são diretamente inferidos se for utilizada a "representação direta" de uma dimensão, ou sua modificação.

Desta forma, sob visualização mais genérica, influências, ou interações contextuais, só podem ser representadas através de ontologias. Neste caso específico, esta regra/princípio pode ser vista na figura 5.45. Esta figura é um representação reduzida do resultado da inferência efetuada pelo motor de raciocínio FACT++ utilizando as relações das tabelas do capítulo anterior relacionadas com este princípio definido.

Conceitual e contextualmente é claro e inequívoco o fato que as interações, e/ou dependências, existem, bem como o importante e relevante aspecto que elas não são diretamente representadas, mas inferidas. Por exemplo, a Flexibilidade Dimensional além de, obviamente, demandar dependência do conceito Dimensão também necessita de conexões com outros conceitos do domínio, como Feature, Part e Product para que seja representada inequivocamente.

É relevante notar que os conceitos relacionados não necessariamente precisam ter associação direta com um conceito em questão. Somente a partir dos resultados inferidos é

que torna-se possível descrever o quão viável é, na verdade, conhecer um determinado conceito. Somente o conhecimento intrínseco sobre um determinado conceito é que permite que ele seja utilizado efetiva e eficazmente[2].

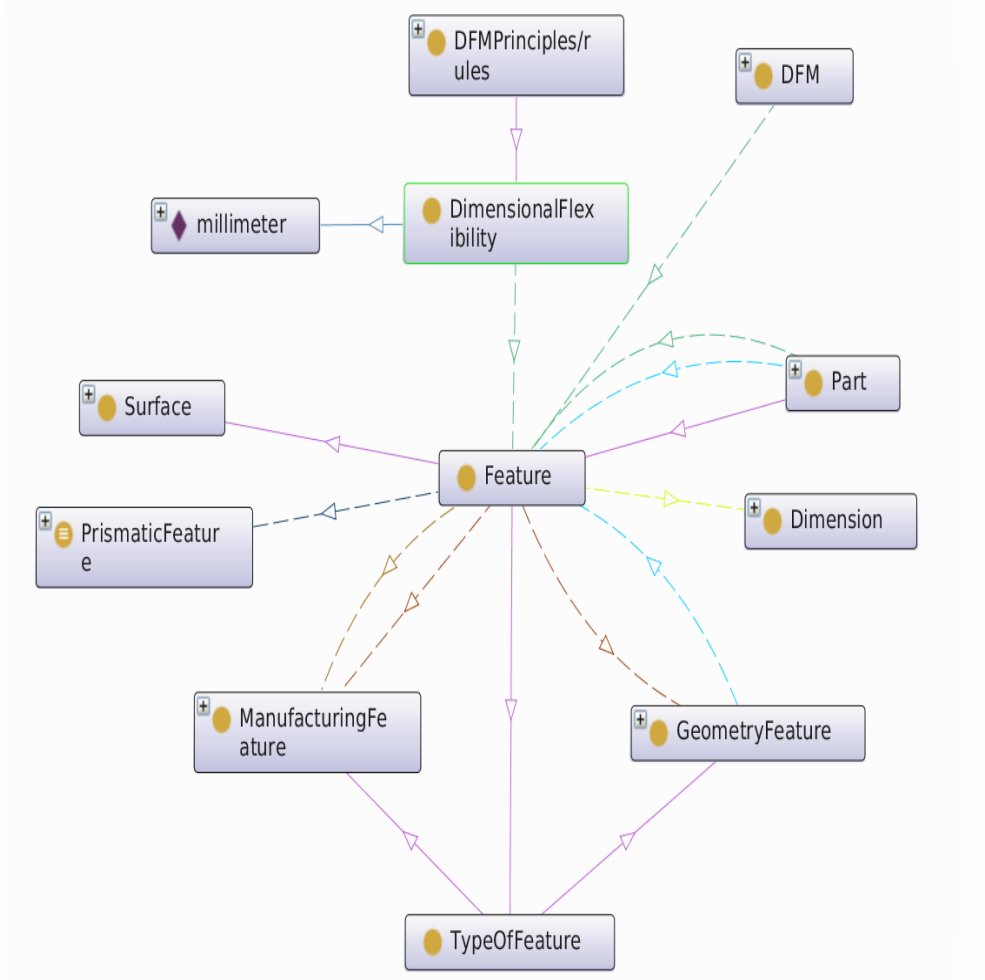


Figura 5.45 – Regra DFM: Flexibilidade Dimensional inferida.

Portanto, a partir das tabelas relacionadas descritas no capítulo 4, que representam as interações definidas entre os contextos associados, as tabelas inferidas 5.40a,b[3] demonstram como existe a inclusão de informação devido à associação com outros conceitos e propriedades. Especificamente, neste caso é inequívoco como as classes Dimension, Feature e Part são associadas com o princípio DimensionalFlexibility.

A falta desta inter-relação explícita claramente limita e complica o desenvolvimento de sistemas, no caso DFM. Em primeiro lugar, sistemas sem conhecimento pleno sobre suas inter-relações não são totalmente eficazes e eficientes. *As inter-relações embutem o real conhecimento, portanto mais completo, sobre um sistema*. Adicionalmente, uma eventual implementação sistêmica eficaz sem levar em consideração aspectos como este, por exemplo, torna-se-ia muito mais complexa, ou inclusive não factível.

[2]Esta utilização pode ser de diferentes formas: interação ou utilização conceitual direta no desenvolvimento de sistemas.

[3]Aspectos inferidos nas tabelas são expressos em *itálico*.

Tabela 5.40a – Classes de princípio, ou regra, DFM–específico Flexibilidade Dimensional.

Atividade	Equivalência	Superclasses
DimensionalFlexibility		Principles/rules ∃ hasDimension (NominalDimension ∧ Tolerance) ∃ hasFeature Feature *Dimension* *Feature* ***Part***

Tabela 5.40b – Membros do princípio/regra DFM Flexibilidade Dimensional.

Princípios/regras	Membros
DimensionalFlexibility	millimiter

5.1.2 Usinabilidade inferida

A manufaturabilidade, obviamente, é o aspecto mais relacionado com o foco deste trabalho, logo o resultado da inferência sobre este conceito demanda atenção especial para que o escopo seja validado corretamente. A figura 4.38 mostra graficamente como é a taxonomia definida associada deste trabalho. No caso, ela demonstra a relevância taxonômica da especificação no projeto ontológico de classes **definidas** e **primitivas**.

Portanto, conforme o expresso, sob contexto produtivo, na figura 5.46, **obrigatoriamente** existe uma relação entre tolerância e manufaturabilidade, aspecto óbvio e largamente conhecido em termos contextuais de manufatura. As tabelas definidas relacionadas no capítulo 4 descrevem em termos lógico-assertivos como esta parte da taxonomia é definida. Em resumo, apenas tolerância é o foco porém, analogamente, o mesmo tipo de inferência é válido para outros princípios DFM, conforme expresso nas seções 5.1.3, 5.1.4.

A figura, que denota o resultado após a inferência, demonstra graficamente como inter-relações indiretas existentes e não definidas explicitamente são habilitadas. Neste caso, todas as interfaces conceituais relacionadas à informação do tipo tolerância, como propriamente expressa, são representadas, além de sua óbvia inter-relação/dependência com dimensões. A parte automatizada da lógica da inferência, por sua vez, é também baseada nas tabelas 4.11,b,c,d. Levando em consideração que as dimensões são relacionadas com *features* → peça → produto, torna-se clara e inequívoca a formação de conceitos mais complexos, os quais abstraem melhor a informação e a tornam mais próxima da compreensão humana, por exemplo.

Tabela 5.41 – Classes de princípio, ou regra, DFM–específico: Manufaturabilidade.

Atividade	Equivalência	Superclasses
Machinability	Tolerance ∧ TypeOfTolerance *TypeOfTolerance* *Tolerance*	Principles/rules ∀ hasMachine Milling ∀ hasMachineAccessory (Fixture ∧ Jig ∧ Tool) ∀ hasTypeOfToleranceDefined TypeOfTolerance

Figura 5.46 – Regra DFM: Manufaturabilidade inferida.

Analogamente ao expresso na seção 5.1.1, a inferência resultante está descrita na tabela 5.41. No caso, apenas a tabela 5.41 é descrita pois as descrições análogas do capítulo 4 não são modificadas. O resultado da inferência demonstra um aspecto conhecido da realidade em manufatura: as interações relacionadas com a classe Tolerância são de sistematização complexaxs.

Isto envolve diversas classes de informação, tais como a intrínseca tolerância existente com seus diversos tipos, bem como a abrangência deste conceito e sua consequente interface com outros conceitos relacionados. A utilização de ontologias é a forma mais adequada para efetuar esta integração.

5.1.3 Padronização inferida

A padronização é atualmente, e cada vez mais se for levado em consideração o contexto de qualidade e mercadológico, uma demanda obrigatória sobre aspectos de manufatura. Isto inclui tanto os aspectos supracitados quanto detalhes relacionados especificamente com a otimização, sob contexto puramente técnico, dos processos de produção[4]. Neste sentido, a definição de uma estrutura conceitual de informação pode facilitar a compreensão e a utilização efetiva e eficiente de componentes em um processo complexo que tem diversas inter relações entre aspectos díspares e com abstração variável, como DFM.

No caso deste conceito definido, a figura e tópicos análogos mostram qual é a compreensão determinada pelo projetista/engenheiro. Este conceito/princípio, inferido na figura 5.47, demonstra como a definição envolve mais informações relacionadas. Especificamente, nela podem ser visualizados e compreendidos os tradicionais, típicos e intrínsecos inter relacionamentos consequentes. Desta forma, quando a informação relacionada ao conceito Padronização é utilizada/requerida automaticamente, suas relações existentes definidas na ontologia, que incluem Principles/rules DFM, Product e Material, também estarão disponíveis.

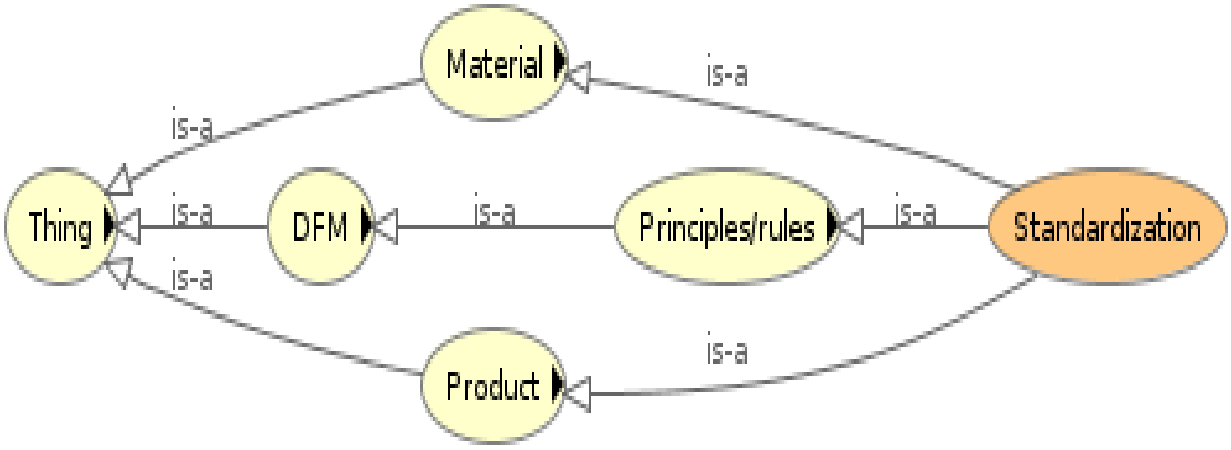


Figura 5.47 – Regra DFM: Padronização inferida.

A tabela 5.42, que nesta etapa considera apenas propriedades do Material, demonstra como o resultado da inferência é semelhante às tabelas definidas análogas. Neste caso, entretanto, a semelhança deve-se ao fato que a ferramenta Protégé automatiza a obtenção de inter-relacionamentos mesmo antes do processo de inferência, como, por exemplo, nos conceitos Product e Material.

[4]Considerações sobre aspectos associados, demandas e requisitos sobre Padronização podem ser vistos à página 114.

Tabela 5.42 – Classes de princípio, ou regra, DFM–específico: Padronização.

Atividade	Equivalência	Superclasses
Standardization	∃ hasMaterialProperty (Appearance ∧ CorrorionResistance ∧ ElectricalConductivity ∧ Formability ∧ Stiffness ∧ Strength ∧ SurfaceRoughness) ∃ hasMachineAccessory (Tool ∧ Fixture ∧ Jig)	Principles/rules ∀ hasMaterial Material *Material* *Product*

5.1.4 Simplicidade inferida

Sob foco de análise contextual, a Simplicidade é o conceito mais facilmente associável com diferentes níveis de abstração dentre os selecionados nesta pesquisa. Isto deriva do fato dele ser um conceito muito abstrato que depende de uma série de outros conceitos, bem como da valoração de cada um deles para uma determinada organização. Analogamente, isto equivale em DFM a diferentes interpretações de o que é simples, ou seja, depende do especialista em questão analisando um projeto: engenheiro, administrador ou responsável do chão de fábrica. Esta variabilidade torna mais complexa a solução de um problema.

Figura 5.48 – Regra DFM: Simplicidade inferida.

Tabela 5.43 – Classes de princípio, ou regra, DFM–específico Simplicidade.

Atividade	Equivalência	Superclasses
Simplicity	∃ hasTypeOfFeature (GeometryFeature ∧ ManufacturingFeature) ∃ hasLimitsMfgOperation (Dimension ∧ Milling ∧ MfgLimit)	Principles/rules

Analisando os três princípios DFM selecionados, é possível perceber que a ontologia possibilita o "enriquecimento" dos conceitos e/ou propriedades ⇒ relações definidas, inclusive associando novas inter relações. Estas novas inter relações *inferidas* são que permitem representar novos aspectos intrínsecos da informação DFM; tais elementos não são passíveis de representação via abordagens tradicionais.

Em geral, pode-se concluir que quanto mais genéricos forem os conceitos, como por exemplo Regras/Princípios e Atividade, maior será a dependência de um elemento intrínseco a ontologias: o membro. O membro é o descritor mais próximo antes do dado, vide, por exemplo, a tabela 5.44. Da mesma forma, esta analogia é válida para as Atividades (componentes), conforme a tabela 5.45 descreve. Neste caso, inclusive, recordando que a arquitetura sistêmica é baseada em agentes, um membro relacionado com esta operacionalidade é expresso pois tem relevância para habilitar este aspecto. Por fim, após a definição da camada lógica mais próxima ao dado, as figuras 5.49 e 5.50

Tabela 5.44 – Princípios/Regras definidas DFM.

Superclasses	Membros
DFM	**DFMindex**
	millimiter

Tabela 5.45 – Atividades (componentes) definidas DFM.

Superclasses	Membros
DFM	AgentAction
	millimiter
	Axis
	Real

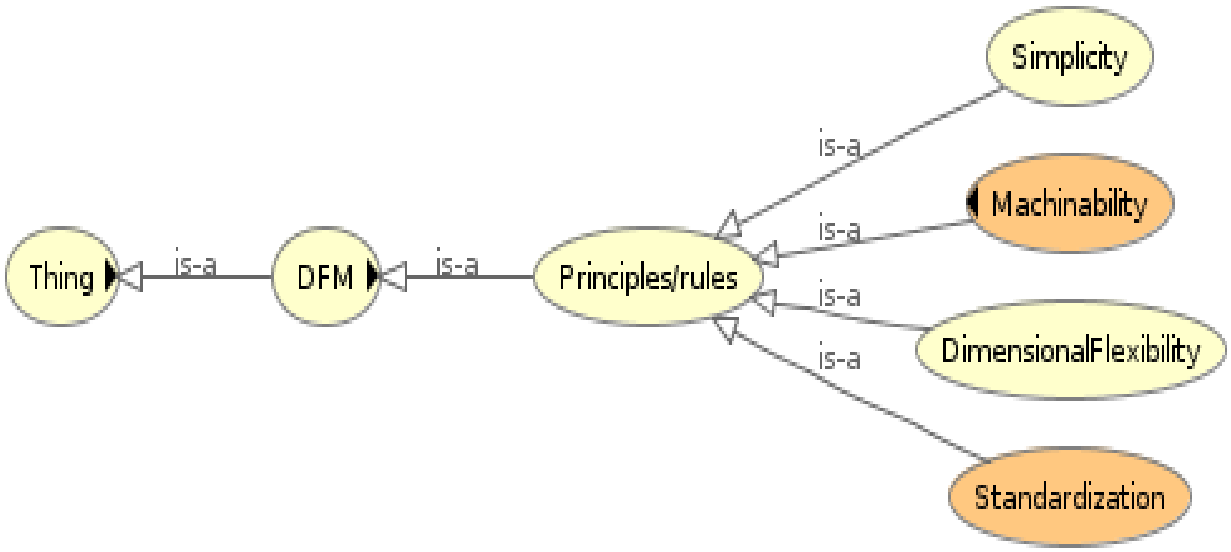


Figura 5.49 – Princípios/Regras definidas DFM.

explicitam, expressas sequencialmente considerando os resultados da inferência pelo motor, ou máquina, de raciocínio, como existe aumento, tanto hierárquico quanto lógico, na informação do escopo no domínio.

5.2 Análise da metainformação

A utilização de ontologias prima pela integração entre fontes de informação diferentes, que são muitas vezes heterogêneas e com focos díspares. Este é um preposto largamente conhecido nesta área. As regras e princípios DFM inferidos, apesar de possibilitarem resultados que clarificam melhor o arcabouço definido, todavia não abstraem a informação ao nível mais próximo da compreensão humana. Desta forma, torna-se necessário utilizar o nível meta da informação, conforme descrito na seção 4.5. A meta representação da informação é feita através da utilização de anotações, que, essencialmente, descrevem o vocabulário de um domínio e seu escopo somente no nível meta. Elas, portanto, servem somente como suporte e não são utilizadas pelo motor de inferência diretamente.

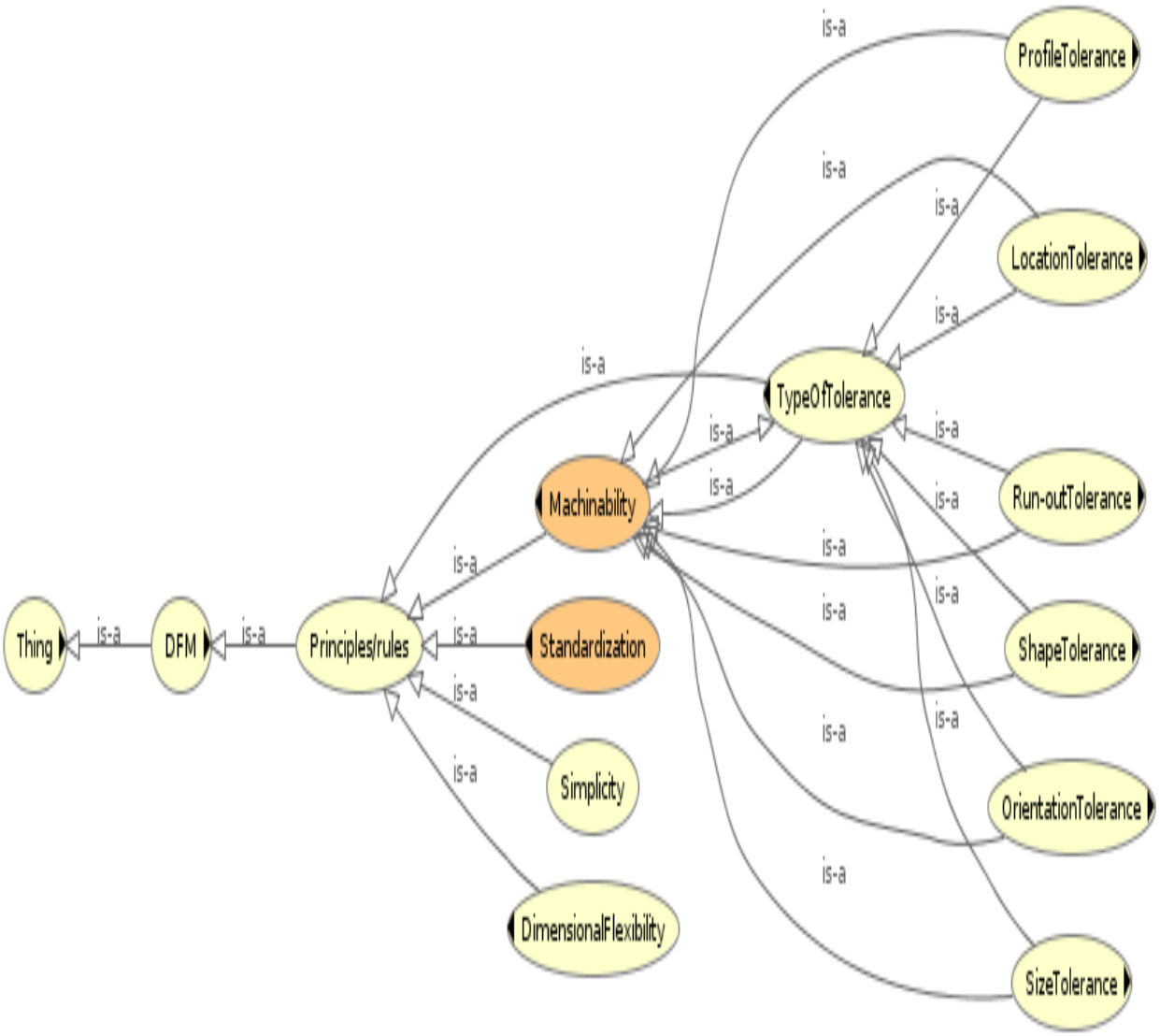


Figura 5.50 – Princípios/Regras inferidas DFM.

Buscando reduzir a duplicação de vocabulários que possuam gramática e sintaxe análogas com o foco desta pesquisa, portanto seguindo o preposto ontológico, o vocabulário DCMI é utilizado. As adaptações são feitas a nível de anotações nas propriedades de anotações da ferramenta Protégé. Devido à sua própria natureza, as propriedades de anotações do vocabulário DCMI, neste momento, não são totalmente integráveis com a ferramenta, conforme descrito acima.

A tabela 5.46[5] descreve como o vocabulário de metainformação[6] é utilizado no arcabouço desta pesquisa. Essencialmente, ela descreve as meta relações informacionais de aspectos considerados relevantes para compreensão dos conceitos envolvidos em DFM sob aspecto de produção. Adicionalmente, torna-se relevante destacar que nesta tabela existe a eventual repetição da nomenclatura de propriedades, p.ex. hasPart como *Annotation*

[5]Descrição=*Annotation Property*. Não fazer associações diretas entre colunas em cada linha da tabela.

[6]Estas descrições refletem apenas parte do conjunto de meta informações passíveis de serem definidas nesta pesquisa.

Property em comparação a hasPart como propriedade de objeto. Esta repetição não é decorrente de uma limitação ou ocasiona problemas na inferência pois são de natureza distinta. Desta forma, torna-se possível (vide a tabela 5.46) definir e descrever meta informações, ou meta conceitos, ou seja, aspectos que não são claramente, ou explicitamente, discerníveis na taxonomia, seja hierárquica ou inter-relacional (propriedades de objetos).

Tabela 5.46 – Meta-vocabulário DFM.

Descrição	Domínios	Faixas	Superpropriedades
abstract	DFM	Analysis Activity Principles/rules	
accessRights	Designer Manufaturer Supervisor		
Agent			
componentAgent	AgentAction		Agent
compAgAvailabilityOfMachinesAndTools	AvailabilityOfMachinesAndTools Manufacturer	string	componentAgent
compAgComputeCost	ComputeCost Designer Manufacturer Supervisor	double	componentAgent
compAgTolerancing	Tolerancing Designer	double	componentAgent
compAgToolAccessibility	ToolAccessibility Manufacturer	string	componentAgent
humanAgent	AID		Agent
humAgApprove	AID Approve	Supervisor string	humanAgent
humAgCompute	AID Compute	Designer Manufacturer double string	humanAgent
humAgRefuse	AID Refuse	Supervisor string	humanAgent
Agenclass	JADE-CLASS		
audience	AvailabilityOfMachinesAndTools Tolerancing ToolAccessibility ComputeCost	Designer Manufacturer Supervisor	
hasPart, ispartOf	DFM	AvailabilityOfMachinesAndTools Tolerancing ToolAccessibility ComputeCost	
isReferencedBy	AvailabilityOfMachinesAndTools Tolerancing ToolAccessibility ComputeCost	Simplicity Machinability DimensionalFlexibility Standardization	

5.3 Completeza da estrutura ontológica no escopo selecionado do domínio DFM

A análise da completeza de uma ontologia é realizada através da execução de questões ("*queries*") ao motor de raciocínio que, em geral, já inferiu a ontologia resultante a partir das asserções definidas. As respostas do processo de inferência e, eventualmente, *dados reais* permitem definir que uma ontologia, como a desta pesquisa, descreve os níveis do arcabouço da informação de forma eficaz. Tipicamente, este processo resulta na *realização dos conceitos* e seu *mapeamento para dados*, denominado *materialização*. A materialização, portanto, perfaz unir a representação taxonômica contexto/conceitual representada pela

ontologia com os dados, que tipicamente estão disponíveis em bancos de dados relacionais. Assim sendo, foi utilizada uma ferramenta, denominada -ontop-, 2015, para relacionar os dados com a ontologia através de mapeamento.

5.3.1 Asseveração taxonômica da ontologia no domínio e escopo da pesquisa

A ontologia é, por definição e conforme já expresso, composta de dois aspectos: sua definição, ou como ela é projetada, e sua versão inferida, ou seja, qual sua estrutura taxonômica após a atuação de um motor de inferência. Além de proporcionar uma versão utilização efetiva, ou "operacional", da taxonomia da informação, ele garante a validade lógica da ontologia. Esta pesquisa, com este objetivo, utilizou três motores de inferência: FACT++, HermiT v1.3.7 e Quest v1.12. A confirmação da ontologia, após a inferência ser validada, é feita através da realização de questões ("*queries*") à base ontológica e verificação sobre a adequacidade das respostas.

Levando em consideração este aspecto e *nesta seção*, assumindo foco DFM apenas sistêmico, as questões seguintes, que possuem foco puramente taxonômico, foram executadas. Os aspectos utilizados para retratar a materialização da Análise DFM são os que possuem relação mais direta com informações armazenadas no banco de dados, bem como permitem explicitar que DFM consiste, em essência, de princípios/regras e análise; são eles: Disponibilidade de Máquinas e Ferramentas (*AvailabilityOfMachinesAndTools*) e Usinabilidade (*Machinability*).

1. Quais são os componentes de análise DFM da ontologia? (Listas 5.18 e 5.19.)
2. Qual é a materialização de conceitos associados com o componente de análise Disponibilidade de Máquinas e Ferramentas ("*AvailabilityOfMachinesAndTools*")? (Listas 5.5, 5.6 e 5.7.)
3. Qual é a materialização de conceitos associados com o princípio DFM Flexibilidade Dimensional ("*DimensionalFlexibility*")? (Listas 5.8, 5.14 e tabela 4.34.)
4. Qual é a materialização de conceitos associados com o princípio DFM Usinabilidade ("*Machinability*")? (Listas 5.10, 5.11 e 5.12.)

5.3.2 Materialização da ontologia no domínio e seu escopo

A materialização da ontologia [Kontchakov *et al.*, 2014] utiliza dados originários de referência clássica em DFM por Bralla, 1999, que estão armazenados em um banco de dados relacional de código aberto denominado PostgreSQL, 2015. A materialização está subdividida em duas partes: aspectos metodológicos relacionados com DFM, por exemplo disponibilidade de infraestrutura de manufatura ("*AvailabilityOfMachinesAndTools*") e seus

princípios/regras ou Flexibilidade Dimensional ("*DimensionalFlexibility*") e Usinabilidade ("*Machinability*"). Tópicos adicionais como detalhamento de peça, maquinário, ferramentas e operações, dentre outros, são partes intrínsecas/embutidas informacionalmente para que os acima citados possam existir e por esta razão não estão descritos. Adicionalmente, torna-se relevante destacar que, nesta etapa, a ontologia deve, obrigatoriamente, já ter sido inferida.

Essencialmente, na integração entre grandes conjuntos de dados existe a necessidade de compreendê-los adequadamente. Sob contexto de informação organizada de forma estrutural, como no paradigma representacional SQL, existe a necessidade adicional de mapear a informação deste paradigma para uma forma taxonômico-contextual mais "amigável" com a interpretação humana dos conceitos utilizados. Desta forma, as seções a seguir correspondem a como é feito este mapeamento a partir dos dados utilizando -ontop-.
.

Em geral, a realização da materialização em ontologias é feita em duas partes sequenciais. Em primeiro lugar, é feito o mapeamento dos dados reais, que no caso são relacionais, com suas condições de inclusão, exclusão, agrupamento e subsunção na taxonomia, conforme necessário. Após o mapeamento é gerada uma série de instâncias que podem ser associadas com indivíduos na ontologia. Em segundo lugar, com o mapeamento gerando indivíduos pode ser feito questionamento ("*querying*") à base ontológica, que já os inclui. As respostas às questões demonstrarão que a ontologia, suas asserções e condições associadas, reflete adequadamente as hipóteses existentes na taxonomia hierárquica e relacional.

Essencialmente, é necessário utilizar a máquina de inferência, seção 5.3.3, diretamente. Vários aspectos nas seções a seguir não possuem um mapeamento direto a dados, ou seja, não existe um questionamento simples com respostas diretas. Os dados são dispersos, além de heterogêneos, adicionalmente. Assim sendo, existe um conjunto de materializações por mapeamentos bastante grande como, por exemplo, toda parte relacionada com materiais e suas propriedades. Desta forma, objetivando um documento o tão objetivo e conciso quanto possível, apenas os dados da relacionados com a materialização da infraestrutura na ontologia são descritos explicitamente (seção 5.3.2.1). Os aspectos adicionais, seja relacionados à análise ou princípios DFM, são melhores descritos na seção 5.3.3.

Integração contextual produto/manufatura

De forma a demonstrar genericamente a integração entre informações DFM, a materialização do domínio e escopo é descrita nas listas 5.1, 5.2 e 5.3.

Lista 5.1 – Código-fonte taxonômico da integração produto/manufatura para adquirir dados da ontologia segundo a hierarquia lógica.

```
SELECT  p.process, g.geometry
FROM    geometry_type g, processes p
WHERE   p.proc_id=5
        AND g.geometry <> 'Round holes'
        AND g.geometry <> 'Nonround holes';
```

Lista 5.2 – Código-objeto taxonômico ("*triplets*") da integração produto/manufatura para mapear/associar dados na ontologia segundo a hierarquia taxonômica.

```
:{p.process} a :MillingSingleCut .
:{g.geometry} a :GeometryFeature .
```

Lista 5.3 – Mapeamento dos dados da integração produto/manufatura segundo a Lista 5.2.

```
:MillingSingleCut              :GeometryFeature
Milling-machine machining      Flat
Milling-machine machining      Two-dimensional contoured
Milling-machine machining      Three-dimensional contoured
Milling-machine machining      Embossed
Milling-machine machining      Hollow shapes
```

5.3.2.1 Materialização da Análise DFM

Disponibilidade de Máquinas e Ferramentas

O descrito acima, contextualmente, permite concluir que analisando a materialização do conceito de disponibilidade de maquinário e ferramental espera-se que estejam associados com ele dados reais destes conceitos que, por sua vez, obedecem às asserções lógico-taxonômicas da ontologia. O maquinário e ferramental utilizados são os presentes no Laboratório de Manufatura Integrada por Computador (LabCIM)[7] da PUCRS. Resumidamente, no contexto desta pesquisa ele está expresso nas tabelas a seguir, que são taxonomicamente indivíduos após a inferência.

Após o inventário da infraestrutura selecionada, foi feita a definição de um modelo entidade-relacional (E-R) implementado no banco de dados supracitado. Desta forma, o mapeamento entre os dados e a taxonomia através de analogia lógico-semântica entre as colunas relevantes desta tabela é associada com classes, subclasses e propriedades de objetos da classe-mãe Machine, conforme a seguir. Como primeiro exemplo está o relacionamento mais genérico semanticamente possível, ou seja, o que faz parte, ou compõe um centro de usinagem de peças prismáticas. Taxonomicamente, portanto, o conceito da classe-mãe Machine engloba diversos tipos de componentes, que dependem de sua especialização/objetivo/finalidade.

[7] <http://www.feng.pucrs.br/~labcim/>

Tabela 5.47a – Componentes físicos do Laboratório CIM da PUCRS.

Tipo	Nome
Fr	Fresa CNC modelo VMC-100 com controlador EMCOtronic TM02 - ESHED Robotec
To	Torno CNC modelo PC TURN 120 - ESHED Robotec
Ro	Robô cartesiano modelo ASRS com controlador com Controller B - ESHED Robotec
Ro	Robô articulado modelo SCORBOT ER IX com Controller B - ESHED Robotec
Ro	Robô articulado modelo Performer-Mk3 com controlador Controller AC - YASKAWA
Ro	Robô SCARA ER14 com controlador Controller B - ESHED Robotec
Di	Parafusadeira pneumática com alimentador autom. de parafusos - VSI Automation FM502H
Di	Paquímetro digital com interface serial óptica
Di	Altímetro digital com interface serial
Di	Alimentador automático de esferas
Di	Leitor de códigos de barras industrial - Symbol
Di	Sensores indutivos - IG-3008-BPKG
Di	Sensores capacitivos - KG-3008-BPKG
Ferr	Vários tipos - SANDVIK

Tabela 5.47b – Significado dos componentes.

Componentes	Abreviação
Torno	Tr
Fresa	Fr
Robô	Ro
Dispositivo	Di
Ferramenta	Ferr

A realização da taxonomia, conforme previamente expresso, é feita por mapeamento dos dados expressos relacionalmente. Os dados, consequentemente, vão ser associados com diferentes classes e suas propriedades, pois ela faz relacionamento direto. Adicionalmente, é necessário que o mapeamento inclua a "operacionalização" das informações expressas nas classes através de propriedades de objetos, pois as propriedades é que permitem inferir como os conceitos são utilizados. Desta forma, torna-se necessário atenção redobrada ao associar colunas diferentes de, muitas vezes, tabelas relacionais distintas. De forma oposta, o paradigma E-R tem seu foco no dado, que é o nível-base da informação numa ontologia. O mapeamento da classe Disponibilidade de Máquinas e Ferramentas aos dados está descrita a seguir: inicialmente, a estrutura dos dados com sua estrutura relacional (Lista 5.4) e posteriormente seu mapeamento taxonômico. Em termos "implementacionais", existe a possibilidade de transformar os dados importados em indivíduos e aprofundar a análise taxonômica da ontologia, porém este não é o objetivo desta pesquisa e os dados mapeados foram salvos em uma nova ontologia.

Lista 5.4 – Estrutura relacional da tabela principal para adquirir dados da ontologia.

```
1 SELECT COLUMN_name, data_type, character_maximum_length
  FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name = '
     equipment';
  column_name data_type character_maximum_length
  equip_id    integer NOT NULL
```

```
equip_type  character  4
equip_brand character varying  50
equip_name  character varying  100
CONSTRAINT equipment_pkey PRIMARY KEY (equip_id)
```

O *plug-in* previamente descrito, -ontop-, realiza o mapeamento para classes e propriedades de objetos da ontologia. As propriedades não estão expressas nas listas pois descrevem o que envolve um conceito, bem como dependem da taxonomia e seu vocabulário. Por exemplo, tais dados na classe mapeada :Milling apenas adicionarão :p{equip_name} ao fim da linha expressa. Este aspecto denota a relevância de sua metafunção na informação. A Lista 5.5 descreve o mapeamento das classes deste componente de análise DFM; cada uma de suas linhas corresponde, respectivamente, às linhas das Listas 5.5 e 5.6 adicionando o primeiro mapeamento genérico da classe Machine como primeiro elemento.

Lista 5.5 – Código-fonte taxonômico dos componentes para adquirir dados da ontologia segundo a hierarquia lógica.

```
SELECT equip_type, equip_name, equip_brand FROM equipment;

1 - SELECT equip_name FROM equipment WHERE equip_type='Fr';

2 - SELECT equip_name FROM equipment WHERE equip_type='Tr';

3 - SELECT equip_brand FROM equipment WHERE equip_type='Ro';

4 - SELECT equip_name FROM equipment WHERE equip_type='Ferr';

5 - SELECT equip_name FROM equipment WHERE equip_type='Di'
    AND equip_brand='Symbol';

6 - SELECT equip_name FROM equipment WHERE equip_type='Di'
    AND equip_brand IS NULL AND equip_name<>'Alimentador␣automático␣de␣
        esferas';

7 - SELECT equip_name FROM equipment WHERE equip_type='Di'
      AND equip_name='Alimentador␣automático␣de␣esferas';
```

Lista 5.6 – Código-objeto taxonômico ("*triplets*") dos componentes para mapear/associar dados na ontologia segundo a hierarquia taxonômica.

```
:{equip_type} a :MachineType . :{equip_name} a :Machine .
1 - :{equip_name} a :MillingMachine .
2 - :{equip_name} a :TurningMachine .
3 - :{equip_name} a :RobotMachine .
4 - :{equip_brand} a :Tool .
5 - :{equip_name} a :BarcodeDevice .
6 - :{equip_name} a :MeteringDevice .
7 - :{equip_name} a :Device .
```

O mapeamento a seguir resulta, em ordem, conforme a Lista 5.6, ao relacionamento de conceitos diretamente ligados ao escopo com os seguintes dados associados: :MillingMachine, :Tool (neste caso devido à quantidade de modelos disponíveis apenas o nome comercial foi utilizado), :TurningMachine, :RobotMachine, :BarcodeDevice, :MeteringDevice e :Device. Os conceitos mais genéricos relacionados com :Machine e :MachineType obtidos como saída (“*output*”) do mapeamento são iguais à segunda e primeira coluna, respectivamente, da tabela 5.47a logo não são repetidos.

Lista 5.7 – Mapeamento dos dados aos conceitos segundo a Lista 5.6.

```
1 - Fresa CNC modelo VMC-100 com controlador EMCOtronic TM02
2 - Torno CNC modelo PC TURN 120
3 - Robô cartesiano modelo ASRS com controlador com Controller B
    Robô articulado modelo SCORBOT ER IX com Controller B
    Robô articulado modelo Performer-Mk3 com controlador Controller AC
    Robô SCARA ER14 com controlador Controller B
4 - SANDVIK
5 - Leitor de códigos de barras industrial
6 - Paquímero digital com interface serial óptica
    Altímetro digital com interface serial
7 - Alimentador automático de esferas
```

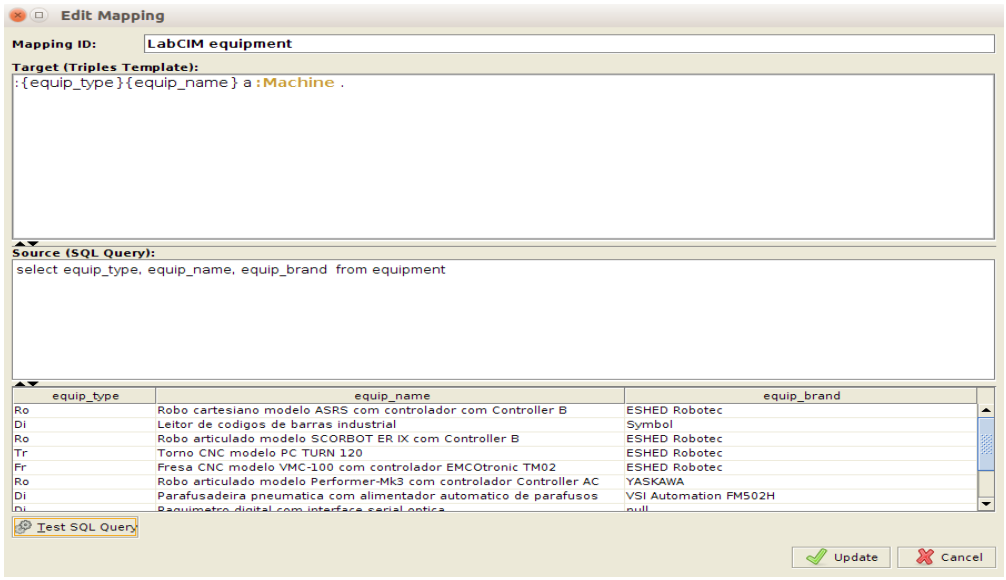


Figura 5.51 – Mapeamento classe/dados do Laboratório CIM da PUCRS.

Quando os dados estão mapeados é possível efetuar questionamento (seção 5.3.3). O resultado da análise ontológica por questionamento deste conceito e dos subsequentes, a seguir, foi exportado para novas ontologias. Alternativamente, pode ser feita a geração de indivíduos para ontologia corrente.

5.3.2.2 Materialização de Princípios/Regras DFM

Flexibilidade Dimensional

A Flexibilidade Dimensional é um princípio/regra que, dentro do contexto DFM, pode ser subdividido, primariamente, com dois aspectos: adequabilidade à consistência dimensional, que é advinda de processos de análise de encadeamento dentre as tolerâncias descrita através sólidos característicos ("features") dos componentes e a possibilidade de fabricar uma determinada peça dentro dos limites do maquinário disponível *per se de acordo o material utilizado*. Portanto, este princípio é, neste exemplo e sem perda da generalidade, restrito à integração destas classes. A ser ressaltado é o fato de o quão intrínseca é a interface entre este princípio com as outras classes pertencentes ao domínio no escopo.

Portanto, sob o aspecto estrito de adequacidade, e somente nesta seção pois é um conceito mais amplo, vide acima, a materialização da Flexibilidade Dimensional descreve as classes definidas associáveis sugeridas por processos associados com materiais ferrosos ou alumínio segundo Bralla, 1999. Expressa em termos de uma questão à base de conhecimento, ela pode ser descrita como: quais são os componentes básicos de um projeto de produto e como posso realizá-los com o maquinário, especificamente a máquina fresadora?

Lista 5.8 – Código-fonte taxonômico do conceito Flexibilidade Dimensional para adquirir dados da ontologia segundo a hierarquia lógica.

```
SELECT DISTINCT ON (c.component,m.mb_description,g.geometry) c.component, g.geometry, m.
    mb_description, m.al_description, p.process, e.equip_name
FROM component c, geometry_type g, processes p, material m, equipment e
WHERE (g.geom_type_id=2 OR g.geom_type_id=3)
      AND proc_id=5
      AND (m.mb_description='Aluminum' OR m.mb_description='Ferrous')
      AND equip_type='Fr'
      AND (g.geom_type_id=1 OR g.geom_type_id=2 OR g.geom_type_id=3)
      AND c.comp_type='C'
      AND (c.comp_id = 1 OR c.comp_id = 11 OR c.comp_id = 19 OR c.comp_id = 22 OR c.
          comp_id = 26 OR c.comp_id = 27)
      AND g.geom_type_id<>1
ORDER BY c.component ASC, m.mb_description ASC, g.geometry DESC LIMIT 100;
```

Lista 5.9 – Código-objeto taxonômico ("*triplets*") do conceito Flexibilidade Dimensional para mapear/associar dados na ontologia segundo a hierarquia taxonômica.

```
:{c.component} a :Part .
:{g.geometry} a :Feature .
:{m.mb_description} a :Metal_base .
:{m.al_description} a :Alloy .
:{p.process} a :Operation .
:{e.equip_name} a :MillingMachine .
```

Devido ao tamanho elevado do resultado deste mapeamento, a tabela 5.48 é pós-processada a partir do *"query"* da lista 5.8 associada com o "*triplet*" da lista 5.9, ou seja,

informa apenas dados considerados relevantes pelo autor no escopo do contexto da pesquisa.

Tabela 5.48 – Mapeamento dos dados ao conceito Flexibilidade Dimensional.

:Part	:Feature	:Metal_base	:Alloy	:Operation	:MillingMachine
Cam lobs (automotive)	Two-dimensional contoured	Aluminum	213.F (CH3-F)	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Cam lobs (automotive)	Three-dimensional contoured	Aluminum	1100-0	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Cam lobs (automotive)	Two-dimensional contoured	Ferrous	Stainless steels	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Cam lobs (automotive)	Three-dimensional contoured	Ferrous	431	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Chased threads	Two-dimensional contoured	Aluminum	213.F (CH3-F,1)	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Chased threads	Three-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Chased threads	Two-dimensional contoured	Ferrous	C1212	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Chased threads	Three-dimensional contoured	Ferrous	C1020	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Gear teeth (heavy loads)	Two-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Gear teeth (heavy loads)	Three-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Gear teeth (heavy loads)	Two-dimensional contoured	Ferrous	416	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Gear teeth (heavy loads)	Three-dimensional contoured	Ferrous	Carbon and allory steels 12L14	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Gear teeth (ord. service, diam. pitch under 10)	Two-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Gear teeth (ord. service, diam. pitch under 10)	Three-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Gear teeth (ord. service, diam. pitch under 10)	Two-dimensional contoured	Ferrous	431	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Gear teeth (ord. service, diam. pitch under 10)	Three-dimensional contoured	Ferrous	416	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Milled threads	Two-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Milled threads	Three-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Milled threads	Two-dimensional contoured	Ferrous	Stainless steels	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Milled threads	Three-dimensional contoured	Ferrous	431	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Worm gears, general	Two-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Worm gears, general	Three-dimensional contoured	Aluminum	Cast and wrought alloys	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Worm gears, general	Two-dimensional contoured	Ferrous	C1119	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)
Worm gears, general	Three-dimensional contoured	Ferrous	C1020	Milling-machine machining	Fresa CNC VMC-100 (EMCOtronic TM02)

Usinabilidade

A usinabilidade materializada, idealmente como requisito mínimo, deve responder ao questionamento sobre o inventário de uma instalação de fabricação (*"facility"*) no mapeamento da informação; fato provado acima. Adicionalmente, ela deve prover informações associadas com os materiais disponíveis e o relacionamento de suas propriedades para o processo e tipo de operação em questão. Analogamente ao algoritmo de solução/mapeamento descrito acima, as listas a seguir descrevem esta resolução. Os aspectos associados ao mapeamento dos dados do processo de usinagem são relacionados com a taxa de usinagem, fator que tem o material B1112 (latão) como base com valor 100, natureza do corte e acabamento esperado.

Os materiais[8] (Lista 5.12) são expressos em termos de seu tipo e das eventuais ligas com a ferramenta utilizada para manufatura. Neste caso, conforme expresso acima, a descrição do ferramental restringe-se a uma marca comercial e assume que todos materiais possuam uma ferramenta capaz de processá-la, sem perda de generalidade.

Lista 5.10 – Código-fonte taxonômico do conceito Material para adquirir dados da ontologia segundo a hierarquia lógica.

```
1 SELECT mb_description, al_description, equip_type FROM
    material,equipment WHERE equip_brand='SANDVIK';
```

Lista 5.11 – Código-objeto taxonômico ("*triplets*") do conceito Material para mapear/associar dados na ontologia segundo a hierarquia taxonômica.

```
1 :{mb_description} a :Metal_base .
  :{al_description} a :Alloy .
  :{equip_name} a :Tool .
```

[8]Nomenclatura em Inglês.

Lista 5.12 – Mapeamento dos dados ao conceito Material segundo a Lista 5.11.

```
1  :Metal_base   :Alloy                          :Tool
   Aluminum      Cast and wrought alloys         Ferr
   Aluminum      213.F (CH3-F,1)                 Ferr
   Aluminum      520.0-T4 (220-T4,1)             Ferr
5  Aluminum      217-T4                          Ferr
   Aluminum      1100-H12                        Ferr
   Aluminum      1100-0                          Ferr
   Aluminum      520.0-F (220-F,1)               Ferr
   Beryllium     Unalloyed Beryllium             Ferr
10 Columbium     Unalloyed, wrought Columbium    Ferr
   Copper        Wrought alloys                  Ferr
   Copper        Brass, high leaded (342,4)      Ferr
   Copper        Brass, medium leaded (340,4)    Ferr
   Copper        Naval brass (464,4)             Ferr
15 Copper        Aluminum bronze (614,4)         Ferr
   Ferrous       Carbon and allory steels 12L14  Ferr
   Ferrous       C1212                           Ferr
   Ferrous       C1119                           Ferr
   Ferrous       C1114                           Ferr
20 Ferrous       C1020                           Ferr
   Ferrous       C1040                           Ferr
   Ferrous       4140 resulfurized               Ferr
   Ferrous       4140                            Ferr
   Ferrous       Stainless steels                Ferr
25 Ferrous       410                             Ferr
   Ferrous       416                             Ferr
   Ferrous       430 F                           Ferr
   Ferrous       203                             Ferr
   Ferrous       303                             Ferr
30 Ferrous       430                             Ferr
   Magnesium     ASTM-AZ-61                      Ferr
   Magnesium     ASTM-AZ-91                      Ferr
   Molybdenum    Unalloyed, wrought Molybdenum   Ferr
   Nickel        Nickel 200 to 233               Ferr
35 Nickel        Monel alloy 400 to 404          Ferr
   Nickel        Superalloys                     Ferr
   Tantalum      Unalloyed, wrought Tantalum     Ferr
   Titanium      Commercially pure               Ferr
   Titanium      T1-6A1-4V                       Ferr
40 Titanium      T1-13V-HCr-3A1                  Ferr
   Tungsten      Unalloyed, wrought              Ferr
   Tungsten      Unalloyed, sintered             Ferr
   Zirconium     Commercially pure               Ferr
   Zinc          ASTM-AG40A (XX111)              Ferr
```

5.3.3 Interface com a máquina de inferência OWL

A partir da materialização dos dados na ontologia, no caso ontologia DFM no escopo relatado, é possível finalizar a interconexão entre a logicidade de conceitos "pura" com sua *representação através de dados*. A verificação da lógica da ontologia, ou da materialização, é feita através de "*queries*" em linguagem SPARQL. Esta etapa, entretanto, para sua realização, possui aspectos que interagem intrinsecamente com o conceito/princípio ontológico de informação distribuída e sua integração.

Neste contexto, bem como nesta seção, torna-se relevante analisar se a ferramenta utilizada, Protégé, possibilita este aspecto. Adicionalmente, é relevante ressaltar que este aspecto amplia o conceito ontológico *per se*. Utilizando o descrito acima e a capacidade do "*plugin*" -ontop-, os resultados da materialização podem ser armazenados em novas ontologias. Portanto, as "*queries*" não tem perda de logicidade devido ao -ontop-. As "*queries*" -ontop- SPARQL a seguir respondem a algumas questões típicas em DFM, provando a validade e viabilidade da abordagem desta pesquisa.

As "*queries*" são, sob certo aspecto, análogas à linguagem SQL. Seus resultados demonstram a "compreensão" ontológica. Conforme descrito, os resultados podem ser associados com outras ontologias, porém, mais tipicamente, são armazenados em novos indivíduos gerados pelo processo de inferência.

As respostas às perguntas, portanto, derivam deste interfaceamento dado/classe proporcionado pelo -ontop-. A figura 5.52 descreve como esta estratégia é utilizada nesta pesquisa. As duas perguntas associadas aos dados adquiridos na ontologia, que utiliza, neste caso, a máquina de inferência Quest, estão descritas a seguir.

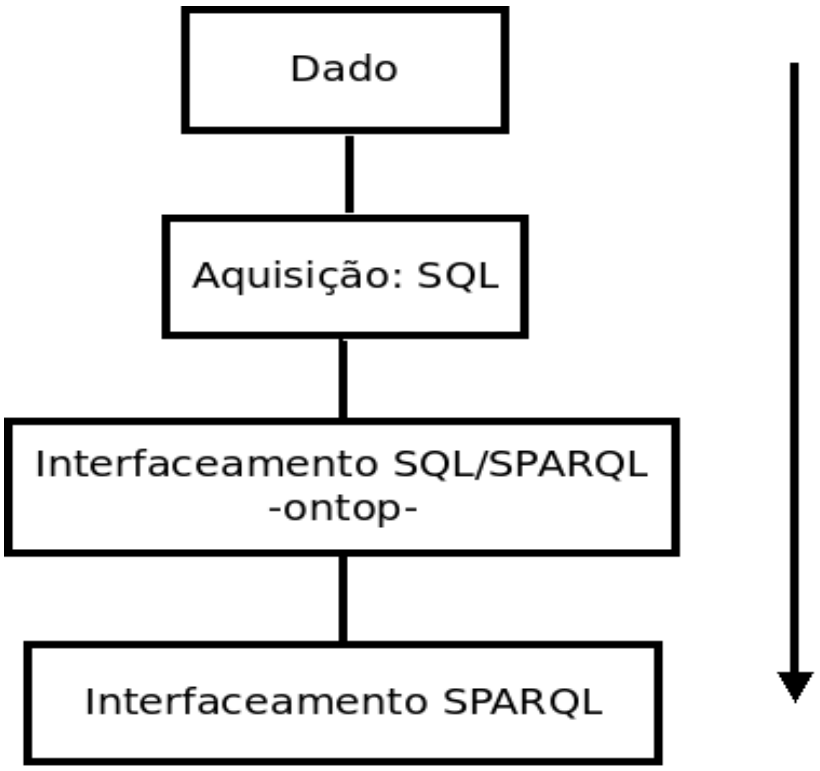


Figura 5.52 – Materialização e inferência SPARQL.

1. Qual é a materialização[9] dos componentes básicos de um projeto de produto e como pode-se realizá-los com o maquinário, especificamente a fresa, e ferramental disponível[10]?

Lista 5.13 – Código-fonte da ontologia segundo a hierarquia lógica.

```
SELECT c.component, g.geometry, p.process, e.equip_type
    FROM processes p, geometry_type g, component c,
    equipment e
WHERE c.comp_type='C'
  AND p.process = 'Milling-machine machining'
  AND g.geom_type_id <> 5
  AND c.geom_type_id = g.geom_type_id
  AND (p.proc_id = 4 OR p.proc_id = 5 OR p.proc_id = 14 OR
       p.proc_id = 15 OR p.proc_id = 24 OR p.proc_id = 38
      OR p.proc_id = 39 OR p.proc_id = 40)
  AND e.equip_type = 'Ferr'
ORDER BY geometry DESC, component, process ASC;
```

Lista 5.14 – Código-objeto taxonômico ("*triplets*").

```
:{component} a :Feature
:{geometry} a :GeometryFeature
:{process} a :Machining
:{equip_type} a :Tool .
```

Que resulta na tabela 5.49 na próxima página.

[9] Esta primeira pergunta lida com materialização, mais especificamente.
[10] A ferramenta é descrita genericamente.

Tabela 5.49 – Componentes básicos de um projeto de produto e sua manufatura com o maquinário, especificamente a fresa considerando somente contexto genérico de operações de usinagem, e ferramental disponível no LabCIM.

Componente	Geometria	Processo	Ferramenta
Milled threads	Two-dimensional contoured	Milling-machine machining	Ferr
Rolled threads	Two-dimensional contoured	Milling-machine machining	Ferr
Seats for antifriction-bearing races	Two-dimensional contoured	Milling-machine machining	Ferr
Cam lobs (automotive)	Three-dimensional contoured	Milling-machine machining	Ferr
Crankpins	Three-dimensional contoured	Milling-machine machining	Ferr
Cylinder bores (automotive)	Three-dimensional contoured	Milling-machine machining	Ferr
Cylinder bores: O-rings or leather packings	Three-dimensional contoured	Milling-machine machining	Ferr
Gear teeth (ordinary service, diametral pitch over 10)	Three-dimensional contoured	Milling-machine machining	Ferr
Housing fits (no gasket or seal)	Three-dimensional contoured	Milling-machine machining	Ferr
Journal bearings, general	Three-dimensional contoured	Milling-machine machining	Ferr
Journal bearings (precision)	Three-dimensional contoured	Milling-machine machining	Ferr
Piston pins	Three-dimensional contoured	Milling-machine machining	Ferr
Piston rods: O-rings or leather packings	Three-dimensional contoured	Milling-machine machining	Ferr
Pistons	Three-dimensional contoured	Milling-machine machining	Ferr
Press fits, general keys and keyways	Three-dimensional contoured	Milling-machine machining	Ferr
Pressure-lubricated bearings	Three-dimensional contoured	Milling-machine machining	Ferr
Push fits	Three-dimensional contoured	Milling-machine machining	Ferr
Slideways and gibs	Three-dimensional contoured	Milling-machine machining	Ferr
Teeth of ratches and pawls	Three-dimensional contoured	Milling-machine machining	Ferr
Valve seats	Three-dimensional contoured	Milling-machine machining	Ferr
Valve stems (automotive)	Three-dimensional contoured	Milling-machine machining	Ferr
Worm gears, general	Three-dimensional contoured	Milling-machine machining	Ferr
Worm gears (heavy loadings)	Three-dimensional contoured	Milling-machine machining	Ferr

Finalmente, apenas para confirmar o contexto da ontologia, três perguntas sobre as inter relações sujeito/objeto e sobre as regras e componentes DFM são feitas em SPARQL utilizando a máquina de inferência FACT++.

2 Quais são as inter relações taxonômico-hierárquicas sujeito/objeto da ontologia[11,12]?

Lista 5.15 – Quais são as inter relações sujeito/objeto da ontologia?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX owl: <http://www.w3.org/2002/07/owl>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?object ?subject
  WHERE {?subject rdfs:subClassOf ?object}
```

Que resulta na tabela 5.50.

[11] Devido à grande extensão de dados/conceitos, apenas duplas selecionadas são descritas.

[12] No caso, em termos coloquiais em Português, a pergunta a seguir equivale a um questionamento sobre o que está relacionado com um determinado objeto.

Tabela 5.50 – O que está relacionado com um determinado objeto intrínseco à DFM?

object	*subject*
Product	Part
Part	Feature
Feature	TypeOfFeature
TypeOfFeature	GeometryFeature
TypeOfFeature	ManufacturingFeature
hasFeature **some** Feature	Part
hasFeature **exactly** 1 Feature	GeometryFeature
Dimension	NominalDimension
Dimension	ToleranceDimension
Tolerance	ToleranceDimension
Tolerance	DesignedTolerance
DesignedTolerance	DesignedLowerLimit
hasNominalDimensionDefined **some** Dimension	Feature
hasDimension **only** NominalDimension	NominalDimension
isManufacturingAccessoryOf **only** Machine	Accessories
hasRole **some** (Designer **or** Manufacturer **or** Supervisor)	Compute
hasRole **only** Supervisor	Control
hasRole **only** Supervisor	Approve
hasActivity **some** (AvailabilityOfMachinesAndTools **or** ToolAccessibility)	Manufacturer
Accessories	Tool
Accessories	Fixture
Accessories	Jig
Accessories	Device
hasFeature **some** Feature	DimensionsalFlexibility
hasLimitsMfgOperation **some** (DesignedUpperLimit **and** MfgUpperLimit)	ManufaturingFeature
hasLimitsMfgOperation **some** (DesignedLowerLimit **and** MfgLowerLimit)	ManufaturingFeature
hasOperation **some** (End_(slot_widths) **or** Face **or** Slotting_(width) **or** Straddle)	MillingMachine
hasProcess **some** MillingSingleCut	DFM
hasMaterialCharacteristic **some** MaterialIdentifier	Machining
hasMaterialCharacteristic **some** Cutting_nature	Machining
Process	Operation
Operation	OperationType

3 Como são relacionadas informações entre equipamentos de fabricação e tolerâncias de projeto?

A clara e inequívoca vantagem da utilização de ontologias é direta no que tange a esta pergunta, vide a seguir. Neste caso, existe a necessidade da integração entre *duas hierarquias taxonômicas diferentes*: uma relacionada com conceitos de tolerâncias de projeto e outra relacionada com os limites possíveis de manufatura de um conjunto de maquinário. Analogamente, esta é uma situação comum, típica e tradicional no processo DFM *per se*. A seguir, sete "*queries*" à ontologia denotam como existe o enriquecimento da informação através do aumento de sua abrangência neste aspecto em DFM

A forma ideal de representar esta diferença taxonômica é unificá-las através de propriedades que as interligam. Portanto, a forma de integração entre ?designed_tolerance e ?tolerance_link utiliza um propriedade que demanda que as tolerâncias de projeto sejam relacionadas com as tolerâncias "operacionais". Esta estratégia é válida em

todos componentes análogos da ontologia, conforme necessário[13].

Lista 5.16 – Contextualmente, o que relaciona as tolerâncias dos equipamentos de fabricação disponíveis e tolerâncias projetadas em DFM?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX beang: <http://jade.cselt.it/beangenerator#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

1:
SELECT DISTINCT ?designed_tolerance
WHERE {
?designed_tolerance rdfs:domain beang:DesignedTolerance .}

2:
SELECT DISTINCT ?tolerance_link
WHERE {
beang:hasToleranceLinkDefined rdfs:domain ?tolerance_link .}

3:
SELECT DISTINCT ?type_tolerance_link
WHERE {
?type_tolerance_link rdfs:subPropertyOf beang:hasToleranceLinkDefined .}

4:
SELECT DISTINCT ?tolerance_lower_link
WHERE {
beang:hasLowerToleranceLinkDefined rdfs:range ?tolerance_lower_link .}

5:
SELECT DISTINCT ?tolerance_upper_link
WHERE {
beang:hasUpperToleranceLinkDefined rdfs:range ?tolerance_upper_link .}
```

A primeira questão acima denota o que está envolvido com uma tolerância feita durante o projeto de um produto/peça/montagem. Considerando que a informação em ontologias é descrita por "*triplets*" (SPO), o resultado da inferência relacionada resulta numa informação contextual que descreve o que é uma tolerância (hasTolerance), possui uma ligação com a capacidade de fabricação (hasToleranceLinkDefined) e é representada por um intervalo numérico no nível de dado (is_tol_mfg_interval).

designed_tolerance
hasTolerancce
hasToleranceLinkDefined
is_tol_mfg_interval

Conforme expresso, se for considerado o aspecto de eventuais discrepâncias informacionais, que são passíveis de existir em DFM, a propriedade de objeto hasToleranceLinkDefined descreve a ligação que existe entre uma tolerância projetada e a

[13] As "*queries*" a seguir tem suas respostas descritas sequencialmente nas tabelas subsequentes.

capacidade de uma determinada máquina. Portanto, ela necessita possuir em seu domínio e faixa ("range") ambas representações de tolerâncias, descrita na segunda questão da lista 5.16 com sua resposta na tabela a seguir.

tolerance_link
DesignedTolerance
ToleranceDimension

A terceira questão demonstra quais são os tipos de ligações dentre tolerâncias projetadas: limites superior e inferior entre as tolerâncias:

tolerance_interval
hasLowerToleranceLinkDefined
hasUpperToleranceLinkDefined

A quarta e quinta questões descrevem que tipos de informações estão relacionadas com cada extremo de uma tolerância; através disto torna-se possível compreensão contextual mais completa do significado real da tolerância[14]. No caso específico, o limite é descrito por um número representado por um indivíduo.

tolerance_lower_link
DesignedLowerLimit and MfgLowerLimit

tolerance_upper_link
DesignedUpperLimit and MfgUpperLimit

4 Quais são as regras e componentes DFM, sob contexto hierárquico, presentes na ontologia, ou o que está relacionado com DFM?

Analogamente, utilizando a *"query"* genérica da lista 5.15 e processando-a da mesma forma, torna-se possível obter as listas a seguir para as seguintes perguntas:

4.1 Conceitualmente, o que existe relacionado com DFM?

[14]Estas duas propriedades de objeto possuem como range inferido as classes Tolerancing, MfgLowerLimit ou MfgUpperLimit, respectivamente, e DesignedLowerLimit ou DesignedUpperLimit, respectivamente.

Lista 5.17 – DFM envolve que aspectos?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX owl: <http://www.w3.org/2002/07/owl>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX beang: <http://jade.cselt.it/beangenerator#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?subject
WHERE {?subject rdfs:subClassOf beang:DFM .}
```

Resultando em,

subject
DFMPrinciples/rules
DFMActivity

4.2 Quais são os princípios e regras relacionados com DFM?

Lista 5.18 – Quais são os princípios e regras desta ontologia DFM?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX owl: <http://www.w3.org/2002/07/owl>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX beang: <http://jade.cselt.it/beangenerator#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?subject
WHERE {?subject rdfs:subClassOf beang:DFMPrinciples/rules .}
```

Resultando em,

subject
DimensionalFlexibility
Simplicity
Machinability
Standardization

4.3 Quais são as atividades DFM desta ontologia?

Lista 5.19 – Quais são algumas atividades relacionadas com DFM?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX owl: <http://www.w3.org/2002/07/owl>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX beang: <http://jade.cselt.it/beangenerator#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?subject
WHERE {?subject rdfs:subClassOf beang:DFMActivity .}
```

Resultando em,

subject
Compute
ComputeCost
AvailabilityOfMachinesAndTools
ToolAccessibility
Tolerancing

4.4 Sobre o que lidam os princípios e regras descritros acima?

Lista 5.20 – Os princípios e regras são sobre que assunto?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX owl: <http://www.w3.org/2002/07/owl>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX beang: <http://jade.cselt.it/beangenerator#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?subject
WHERE {?subject rdfs:range beang:DFMPrinciples/rules .}
```

Resultando em,

subject
isDFM

5.3.3.1 Exemplo de análise DFM em uma peça

As perguntas acima responderam como, contextualmente, a taxonomia definida na ontologia habilita a análise DFM, inclusive com exemplos associados a indivíduos de um ambiente acadêmico real. Entretanto, sob visão mais genérica, inclusive denotada pelo título deste trabalho, idealmente torna-se necessário conhecer como um exemplo de uma peça-protótipo projetada seria analisado por este arcabouço de pesquisa.

A peça-protótipo, somente geometria, figura 5.53 abaixo, é um bloco prismático simples com uma "*feature*" do tipo "*pocket*" com "*sharp corners*" (coloquialmente, "cantos vivos"). A peça foi projetada no Pro-Engineer Wiildfire v5.0 e exportada para o formato STEP num arquivo .stp. Neste aspecto, ela foi importada para o arcabouço pós-processada através de *scripts* bash para geração de tabelas que serão utilizadas pelo -ontop-, pois o *plug-in* OntoSTEP não está plenamente integrado todavia.

Analogamente à aquisição de *dados reais* para validar a ontologia, como na seção 5.3 e subseções, ou, em termos ontológicos, *indivíduos*, adquiridos e descritos abaixo; neste caso eles são adquiridos pelo mesmo método lá descrito. Especificamente, a peça-exemplo é analisada em relação à viabilidade de processos de manufatura[15].

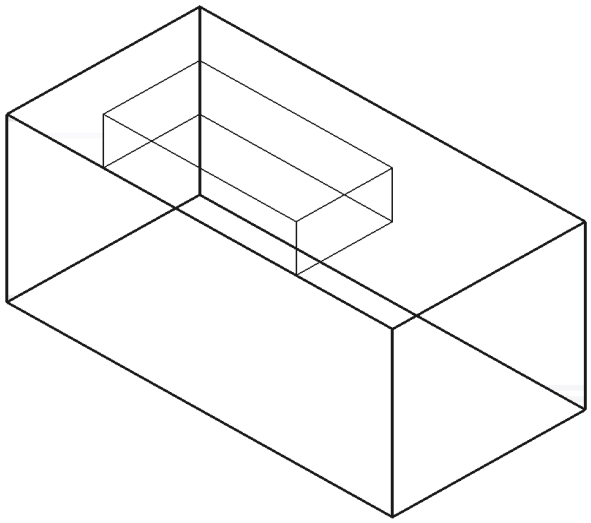


Figura 5.53 – Peça-exemplo: bloco com "*pocket*".

Tabela 5.51 – Dados da peça-protótipo (as nomenclaturas de "*features*" e faces são customizadas)

Elemento	Face
MainFeatPartData	f1, f2, f3, f4, f5, f6
PPPartData	f7, f8, f9, f10, f11

Os *indivíduos* são adquiridos, segundo descrito acima, em tabelas, que são utilizadas por -ontop- para sua criação na ontologia. Analogamente, mais abaixo, os dados das faces associadas com cada "*feature*", que contem vértices, são adquiridos. Por exemplo, os dados das faces de "*feature*" PPPartData são adquiridos do banco de dados pela Lista 5.22, resultando na tabela 5.53. Os *indivíduos* são "*triplets*" formados a partir dessa tabela e suas "*queries*" relacionadas, como na seção anterior.

[15]Todas "*features*" da tabela 5.51 são adquiridas. Apenas sua nomenclatura adotada é utilizada a título de simplificação (ao invés de descrições STEP e/ou valores numéricos).

Lista 5.21 – Dados de *"features"* adquiridos da peça-protótipo.

```
1 SELECT DISTINCT m.elem_mill_mfg, nm.elem_no_mill_mfg
       FROM milled_features m, no_milled_features nm;
```

Tabela 5.52 – Dados de *"features"* da peça-protótipo no banco de dados.

elem_mill_mfg	elem_no_mill_mfg
MainFeatPartData	PPPartData

Lista 5.22 – Dados das faces de *"feature"* PPPartData adquiridos da peça-protótipo.

```
1 SELECT feature, face1, face2, face3, face4, face5, face6
       FROM feature WHERE feature='PPPartData';
```

Tabela 5.53 – Dados das faces de *"feature"* PPPartData de peça-protótipo.

"feature"	face1	face2	face3	face4	face5
PPPartData	f7	f8	f9	f10	f11

Levando em consideração o escopo desta pesquisa: usinagem através de operações de fresagem, a forma mais simplificada e direta para definição da viabilidade é questionar a ontologia se esta peça é possível ser manufaturada nele. Portanto, no caso de análise de viabilidade, a resposta a uma *"query"* deve ser possível ou não possível.

Neste exemplo, a *"query"* específica é um teste se através de operação de fresagem de face a peça-exemplo é fabricável. De forma puramente informacional, abaixo são descritas a validade da ontologia para fresagem de determinados tipos de *"features"*. Adicionalmente, a segunda e a terceira coluna descrevem, respectivamente, que tipo de *"feature"* não é compatível com a operação e qual dado da peça-exemplo, que foi convertido em um indivíduo, corresponde com esta *"feature"*.

Os dados da peça são adquiridos em instâncias de classes do tipo :2Dcontoured, :3Dcontoured e :PocketPrismatic e, neste caso da peça-exemplo, armazenados em indivíduos destas classes. Portanto, a comprovação de impossibilidade de fabricação por esta operação deste processo fica demonstrada verificando que o indivíduo correspondente a *"feature"* PocketPrismatic da peça-exemplo pertence à classe :PocketPrismatic bem como, adicionalmente, que esta geometria não faz parte da propriedade de operações de fresagem: hasMillingOperation.

Lista 5.23 – Como a peça-exemplo da figura 5.53 poderia ser fabricada no escopo, ou seja, quais "features" podem ser realizadas (?subject)?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX owl: <http://www.w3.org/2002/07/owl>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX quest: <http://obda.org/quest#>
PREFIX beang: <http://jade.cselt.it/beangenerator#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT DISTINCT ?subject
WHERE {beang:MillingSingleCut rdfs:subClassOf ?subject .}
```

Resultando em,

Tabela 5.54 – Tipos de "features" que podem ser manufaturadas pela operação do escopo da pesquisa.

subject
hasMillingOperation only (2Dcontoured or 3Dcontoured or Embossed or HollowShape or SlotPrismatic or SlotPrismaticHole)

Lista 5.24 – Quais os dados da "*feature*" com limitação para sua fabricação e onde existe (?feature_no_mfg_part_data)?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>
PREFIX owl: <http://www.w3.org/2002/07/owl>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema>
PREFIX quest: <http://obda.org/quest#>
PREFIX beang: <http://jade.cselt.it/beangenerator#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>

SELECT ?feature_type_non_mfg WHERE {?feature_type_non_mfg a beang:
   PocketPrismatic .}
```

Que resulta em,

Tabela 5.55 – URI do ***dado*** da "feature" que não pode ser manufaturada pela operação do escopo da pesquisa.

feature_type_non_mfg
<http://jade.cselt.it/beangenerator#dfmonto**PPPartData**>

Portanto, fica comprovada a integração consistente, eficiente e efetiva de aspectos DFM no escopo do domínio da pesquisa através desta ontologia, vide a partir da seção 5.3 na página 149, tanto para fabricação *per se* quanto relativamente a uma peça-protótipo a ser fabricada na célula de manufatura. Obviamente, existe um conjunto muito mais amplo

de perguntas-teste que podem ser efetuadas, porém as demonstradas acima corroboram as hipóteses e premissa da desta pesquisa.

6 CONCLUSÕES

Neste capítulo são feitas algumas considerações sobre a pesquisa proposta e sua contribuição nesta área do conhecimento. Após uma breve introdução à questão sobre a pesquisa, este documento revisou tanto seus aspectos mais genéricos associados, como, por exemplo, arquiteturas de projeto e sistemas integrados, quanto especificamente relacionados, como DFM com foco em integração e intercâmbio de informação nesta área.

Ao mesmo tempo, é relevante contextualizar na conclusão deste trabalho a pesquisa DFM proposta dentro de um ambiente mais moderno, real e atual que existe devido à crescente competitividade global, denominado Engenharia Concorrente, ou Simultânea (ES). ES, intrinsecamente, *envolve e demanda* a utilização da informação e do conhecimento como aspecto obrigatório para que seja sistematicamente eficiente e eficaz. Inclusive, historicamente, a literatura ressalta a relevância da taxonomia, ainda que indiretamente expressa, no processo de realização do produto. Logo, a constituição da informação no domínio DFM pode auxiliar/suportar, de forma otimizada e efetiva, a implementação de Engenharia Simultânea eficientemente.

As formas tradicionais de representação de informação foram descritas sob contexto histórico. Adicionalmente, foram detalhadas abordagens relacionadas com um aspecto relevante desta pesquisa: utilização de informações contextuais. Levando em consideração a relevância da presença e realidade computacional há um período importante, foram revisadas estruturas e formas de representação de informação (ou sua "implementação").

A falta de um formalismo capaz de gerenciar a heterogeneidade da informação intrínseca em DFM ocasionará, provavelmente, uma, ou ambas, destas limitações: arquiteturas sistêmicas pouco flexíveis ou excesso de padronização obrigatória nos requisitos de informação de seus componentes. Esta pesquisa propõe a arquitetura de informação que suplante as limitações das abordagens anteriores, as quais minimizam, ou desconsideram, um relevante e intrínseco aspecto relacionado à informação: sua *taxonomia*.

A taxonomia perfaz a base para que a informação e seu contexto sejam corretamente expressos e compreendidos. Isto é um aspecto inviável se, por exemplo, o foco for somente em dados, como foi tradicionalmente utilizado até o surgimento da arquitetura STEP, todavia incompleta. A solução ótima considerando estes aspectos, portanto, é baseada em ontologias. Genericamente, é possível expressar que a minimização desta problemática situação deve-se à falta de uma metodologia específica e eficiente para definir, hierarquizar e permitir a operacionalização da informação intrínseca em sistemas, particularmente se for considerada a heterogeneidade do domínio DFM.

Adicionalmente ao projeto e definição de um arcabouço de informação para o

domínio DFM, é levada em consideração sua utilização prática. Neste caso, a solução utilizada para integração com a ontologia é baseada em agentes de *software*. Essencialmente, esta alternativa é utilizada por permitir a desmembração entre um algoritmo de solução de um problema comparado a como ela é obtida.

A solução descrita substitui este restritivo vínculo por um processo de definição taxonômica da informação DFM, organizando-a através relações hierárquicas e operacionais baseadas nos requisitos informacionais de cada componente do sistema. A ontologia projetada utiliza estas relações hierárquico-operacionais tanto para conceitos DFM-específicos quanto para agentes. Portanto, a arquitetura de informação, incluindo estes dois conceitos: domínio/escopo DFM e de utilização prática (agentes) foi desenvolvida segundo a sequência expressa na figura 4.34.

Provavelmente, a mais relevante qualidade da utilização de ontologias é que elas permitem explicitar informações sob aspecto contextual, ou seja, a informação pode ter diferentes interpretações a partir de uma dada asserção. Conforme expresso, em sistemas tradicionais toda interpretação contextual é implícita consequentemente limitando, em geral, conclusões sobre sua validade.

Em sistemas heterogêneos, como DFM, este aspecto é majorado para obtenção de conclusões lógicas baseadas em um dado conjunto de informações, tanto *input* quanto *output*, pois existe *dependência semântica* entre seus componentes. Desta forma, a utilização de ontologias permite fazer inferências sobre *queries* RDF que ampliam a representação direta pura com, por exemplo, extração e conversão de dados de volume elevado e heterogêneo, bem como habilita diferentes visualizações sobre eles (aspecto típico no domínio DFM).

A ontologia desta pesquisa é válida para o domínio e escopo para o qual foi definida; as conclusões lógicas e sua validade, portanto, são válidas apenas sob o contexto de logicidade. As inferências sobre os resultados deste processo são utilizadas para, adicionalmente, validar os resultados sob contexto humano.

Portanto, conforme expresso nos capítulos 4 e 5, a ontologia define, ou "enriquece", o significado real de um conceito no domínio e escopo, no caso desta pesquisa DFM. A taxonomia hierárquica definida permite a classificação dos conceitos associados, enquanto suas propriedades definidas geralmente estruturam de forma customizada os inter relacionamentos entre os conceitos representados pelas classes.

Enfatiza-se, ademais, que uma taxonomia não impõe nenhuma definição rígida pré-definida. Seus componentes informacionais, classes e propriedades, que são conceitos, são de livre definição e escolha do projetista da ontologia. Logo, a customização da interpretação da informação é possível. Desta forma, a utilização de ontologias, especificamente em DFM, tem uma clara vantagem sobre padronização pura, enquanto, todavia, mantendo-a integrável com o padrão existente STEP [Barbau *et al*., 2012].

O capítulo 5 descreve como o real significado dos conceitos relacionados com a informação do escopo deste domínio é obtido para alguns princípios DFM, previamente escolhidos, através de uma máquina de raciocínio, que realiza a inferência sobre a ontologia definida. Os resultados inferidos correntes mostram que a ontologia inferida resultante representa de forma adequada o escopo deste domínio.

Esta característica, habilitada somente através de ontologias, permite a definição customizada do real significado de um dado. A pesquisa demonstrou uma abordagem efetiva e logica/taxonomicamente mais completa para melhor utilização da informação num escopo do domínio DFM. Desta forma, sistemas DFM podem utilizar o mesmo dado porém sob contextos distintos, flexibilizando sensivelmente a necessária demanda atual por integração. Conforme descrito inicialmente, a abordagem sistêmica *per se* é contínua. Assim sendo, ela poderá ser ampliada para completar o detalhamento no escopo, além de incrementá-lo.

Direcionamento futuro

Levando em consideração que a pesquisa neste expressa permite várias ampliações, imediatas e a médio/longo prazo, as próximas linhas relatam o que, na opinião do autor, perfazem adicionar amplitude realizável a ela. Em primeiro lugar, o aperfeiçoamento lógico-contextual ontológico via IDEF5 para que todas as restrições e interfaces entre classes tenham sido descritas. Desta forma, a ontologia tornar-se-ia fundamentalmente mais consistente num contexto mais amplo do que o realizado.

O segundo aspecto interessante para adicionar à ontologia seria o seu "enriquecimento" através da ampliação de classes, propriedades e indivíduos referentes a aspectos DFM igualmente importantes porém não considerados, tais como desmontagem e confiabilidade, por exemplo. Além da "ampliação" taxonômica sugerida, estudo mais aprofundado de todas consequências resultantes, ou que faltam, inferidas nesta ou na(s) desenvolvida(s) subsequentemente pode providenciar novas direções a serem exploradas.

O terceiro aspecto, provável foco adicional desta pesquisa, é a sua integração com OntoSTEP. Um arcabouço integrado entre esta ontologia, aqui denominada OntoDFM, e OntoSTEP poderá facilitar muito o desenvolvimento de novos sistemas DFM em escopos distintos, pois todos poderão utilizar o mesmo fundamento de informação. Associado com este aspecto, a integração efetiva, eficaz e eficiente com informação STEP, via AP-224, realizará o arcabouço de forma completa.

REFERÊNCIAS BIBLIOGRÁFICAS

-ontop-. *-ontop-*. <http://ontop.inf.unibz.it>, 2015. Knowledge Representation meets Databases group - KRDB.

AART, C. van; CAIRE, G.; PELS, R.; BERGENTI, F. Creating and Using Ontologies in Agent Communication. In: **Proc. Workshop on Ontologies and Agent Systems at AAMAS**. [S.l.: s.n.], 2002.

AHMED, F.; HAN, S. Semantic mismatches for interoperability of product and manufacturing information. **International Journal of CAD/CAM**, v. 13, n. 2, p. 97–108, 2013.

AHMED, S.; KIM, S.; WALLACE, K. A methodology for creating ontologies for engineering design. **Journal of Computing and Information Science in Engineering**, v. 7, n. 2, p. 132–40, 2007.

ALBERTS, M. **YMIR: an Ontology for Engineering Design**. Tese (Doutorado) — Universiteit Twente, 1993.

ALVARES, A.; FERREIRA, J. Development of a system internet-based collaborative cad/-capp/cam in a context of e-manufacturing. In: ABCM. **Proceedings of COBEM 18th International Congress of Mechanical Engineering**. [S.l.], 2005.

ANDERSON, D. **Design for Manufacturability and Concurrent Engineering: How to Design for Low Cost, Design in High Quality, Design for Lean Manufacture, and Design Quickly for Fast Production**. [S.l.]: CIM Press, 2004.

ASM INTERNATIONAL. **ASM Handbook - Volume 1 - Properties and Selection: Irons Steels and High Performance Alloys**. 10th edition. ed. [S.l.]: ASM International Handbook Committee, 1990.

ASM INTERNATIONAL. **ASM Handbook - Volume 2 - Properties and Nonferrous Alloys and Special-Purpose Materials**. 10th edition. ed. [S.l.]: ASM International Handbook Committee, 1990.

AUTODESK Inc. **AutoCAD User Manual**. [S.l.]: Autodesk Inc., 1996.

AUTODESK Inc. **AutoCAD Product Brochure**. 1997.

BAJAJ, M.; PEAK, R.; WILSON, M.; KIM, I.; THURMAN, T.; BENDA, M.; JOTHISHANKAR, M.; FERREIRA, P.; STORI, J. Towards next–generation design–for–manufacturability (dfm) frameworks for electronics product realization. **Session 210, IEMT, Semicon West 2003**, 2003.

BARBAU, R.; KRIMA, S.; RACHURI, S.; NARAYANAN, A.; FIORENTINI, X. OntoSTEP: Enriching product model data using ontologies. **Computer-Aided Design**, v. 44, p. 575–590, 2012.

BAUM, L.; JAGANNATHAN, V.; DODHIAWALA, R. **Blackboard Architectures and Applications, The Erasmus System**. [S.l.]: Academic Press, 1989.

BAYSAL, M.; ROY, U.; SUDARSAN, R.; SRIRAM, R.; LYONS, K. The open assembly model for the exchange of assembly and tolerance information: overview and examples. In: **Proceedings of the ASME international design engineering technical conferences & computers and information in engineering conference**. [S.l.: s.n.], 2004.

BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. **Developing Multi-Agent Systems with JADE**. [S.l.]: Wiley, 2007.

BITTNER, T.; DONNELLY, M.; WINTER, S. Ontology and semantic interoperability. In: PROSPERI, D.; ZLATANOVA, S. (Ed.). **Large-scale 3D data integration: Problems and challenges**. [S.l.]: CRCpress (Tailor & Francis), 2005. p. 139–160.

BLAHA, M.; RUMBAUGH, J. **Modelagem e Projetos Baseados em Objetos com UML 2**. Segunda edição revisada e atual. [S.l.]: Editora Campus, 2006.

BOCK, C.; ZHA, X.; SUH, H. won; LEE, J.-H. Ontological product modeling for collaborative design. **Advanced Engineering Informatics**, v. 24, p. 510–524, 2010.

BOOCH, G. **Object Oriented Analysis and Design**. [S.l.]: Benjamin/Cummins, 1994.

BOOTHROYD, G.; DEWHURST, P. **Design for Manufacture, Product Design for Manufacture and Assembly**. [S.l.]: Addison-Wesley, 1991.

BOOTHROYD, G.; DEWHURST, P. **Design for manufacture and assembly**. 2006. <http://www.dfma.com/>.

BORGO, S. An ontological approach for reliable data integration in the industrial domain. **Computers in Industry**, n. 65, p. 1242–1252, 2014.

BORGO, S.; MASOLO, C. Handbook on Ontologies. In: ____. 2nd. ed. [S.l.]: Springer Verlag, 2009. (International Handbooks on Information Systems), cap. Foundational choices in DOLCE, p. 361–381.

BORST, P.; AKKERMANS, H.; TOP, J. Engineering ontologies. **International Journal Human–Computer Studies**, v. 46, p. 365–406, 1997.

BORST, W. **Construction of Engineering Ontologies**. Tese (Doutorado) — University of Twente, Enschede, 1997.

BRADSHAW, J. M. **Software Agents**. [S.l.]: MIT Press and AAAI Press, 1997.

BRALLA, J. G. **Design for Manufacturability Handbook**. Second edition. [S.l.]: McGraw-Hill, 1999.

BRANDT, S. C.; MORBACH, J.; MIATIDIS, M.; THEISSEN, M. J. M.; MARQUADT, W. Ontology–based information management in design processes. In: MARQUARDT, W. .; PANTELIDES, C. (Ed.). **16th European symposium on computer aided process engineering and 9th international symposium on process systems engineering**. [S.l.]: Elsevier, 2006. p. 2021–2026.

BRANDT, S. C.; MORBACH, J.; MIATIDIS, M.; THEISSEN, M. J. M.; MARQUADT, W. An ontology–based approach to knowledge management in design processes. **Computers and Chemical Engineering**, v. 32, p. 320–342, 2008.

BROCKMAN, J.; DIRECTOR, S. The Hercules CAD task management system. In: **Proc. of the IEEE ICCAD-91 Conference**. [S.l.: s.n.], 1992. p. 254–257.

BROWN, B.; CUTKOSKI, M.; TENENBAUM, J. Next-Cut: next generation framework for concurrent engineering. In: **Proceedings of the MI/IJSME Workshop on Computer Aided Product Development**. [S.l.: s.n.], 1989. p. 269–297.

BUSHNELL, M.; DIRECTOR, S. Automated design tool execution in the Ulysses design environment. **IEEE Trans. on CAD of Integrated Circuits and Systems**, v. 3, p. 279–287, 1989.

BYLANDER, T.; CHANDRASEKARAN, B. Knowledge Acquisition for Knowledge Based Systems. In: ____. [S.l.]: Academic Press, 1988. cap. Generic tasks in knowledge–based reasoning.

CADENCE. **Framework Technology for the 1990's**. 1990.

CAMBAA, J.; CONTERO, M.; JOHNSON, M.; COMPANY, P. Extended 3d annotations as a new mechanism to explicitly communicate geometric design intent and increase CAD model reusability. **Computer-Aided Design**, v. 57, p. 61 73, 2014.

CARBONERA, J.; Rama Fiorini, S.; PRESTES, E.; JORGE, V.; ABEL, M.; MADHAVAN, R.; LOCORO, A.; GONCALVES, P.; HAIDEGGER, T.; BARRETO, M.; SCHLENOFF, C. Defining positioning in a core ontology for robotics. In: IEEE. **Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on**. [S.l.], 2013. p. 1867 – 1872.

CARVER, N. **A Revisionist View of Blackboard Systems**. 1997.

CATALANO, C. E.; CAMOSSI, E.; FERRANDES, R.; CHEUTET, V.; SEVILMIS, N. A product design ontology for enhancing shape processing in design workflows. **Journal of Intelligent Manufacturing**, v. 20, n. 5, p. 553–567, 2009.

CHALUPSKY, H.; FININ, T.; FRITZSON, R.; MCKAY, D.; SHAPIRO, S.; WIEDERHOLD, G. **An Overview of KQML: A Knowledge Query and Manipulation Language**. 1992.

CHANDRASEGARAN, S.; RAMANI, K.; SRIRAM, R.; HORVÁTH, I.; BERNARD, A.; HARIK, R.; GAO, W. The evolution, challenges, and future of knowledge rerepresentation in product design. **Computer-Aided Design**, v. 45, p. 204–228, 2013.

CHANDRASEKARAN, B.; JOSEPHSON, J.; BENJAMINS, V. What are ontologies, and why do we need them? **IEEE Inteligent Systems**, v. 14, n. 1, 20-26 1999.

CHANG, X. **Ontology Development and Utilization in Product Design**. Tese (Doutorado) — Virginia Polytechnic Institute and State University, April 2008.

CHANG, X.; RAI, R.; TERPENNY, J. Development and Utilization of Ontologies in Design for Manufacturing. **Journal of Mechanical Design**, v. 132, 2010.

CHANG, X.; SAHIN, A.; TERPENNY, J. An ontology–based support for product conceptual design. **Robotics and Computer–Integrated Manufacturing**, v. 24, p. 755–762, 2008.

CHANGCHIEN, S.; LIN, L. Concurrent design of machined products: a multivariate decision approach. **IEEE Transactions on Systems, Man and Cybernetics, Part C**, v. 30, n. 2, p. 252–264, 2000.

CHO, J.; HAN, S.; KIM, H. Meta-ontology for automated information of parts libraries. **Computer-Aided Design**, v. 38, p. 713–725, 2006.

COCCHIARELLA, N. B.; FREUND, M. A. **Modal Logic: An introduction to its syntax and semantics**. [S.l.]: Oxford University Press, 2008.

COLACE, F.; SANTO, M. D.; NAPOLETANO, P. Product Configurator: an Ontological Approach. In: **Ninth International Conference on Intelligent Systems Design and Applications**. [S.l.: s.n.], 2009. p. 908–911.

CONCENTRA. **ICAD Product Brochure**. [S.l.], 1997.

CORBETT, J.; DOONER, M.; MELEKA, J.; PYM, C. **Design for Manufacture: Strategies, Principles and Techniques**. [S.l.]: Addison Wesley, 1991.

CRUZ, I.; STROE, C.; PALMONARI, M. Interactive user feedback in ontology matching using signature vectors. In: SOCIETY, I. C. (Ed.). **IEEE 28th International Conference on Data Engineering**. [S.l.: s.n.], 2012. p. 1321–1324.

CUTKOSKI, M.; ENGELMORE, R.; FIKES, R.; GENESERETH, M.; GRUBER, T.; MARK, W.; TENENBAUM, J.; WEBER, J. PACT: An experiment in Integrating Concurrent Engineering Systems. **IEEE Computer**, v. 26, n. 1, January 1993.

DAI, K.; WANG, Y.; ZHANG, S.; ZHANG, J. Research on DFx evaluation system for distributed design based on multi-agent. In: **The 2nd International Workshop on Autonomous Decentralized System**. [S.l.: s.n.], 2002. p. 200–204.

DARPA. **The DARPA Knowledge Sharing Initiative – External Interfaces Group Draft Specification of the KQML Agent Communication Language**. [S.l.], 1993.

DASGUPTA, S. **Creativity in design and invention: computation and cognitive explorations of technological originality**. [S.l.]: Cambridge University Press, 1994.

DASSAULT. **CATIA**. [S.l.]: Dassault Systems, 2006. <http://www.3ds.com/products-solutions/plm-solutions/catia/overview/>.

DASSAULT. **ENOVIA VPLM**. [S.l.]: Dassault Systems, 2006. <http://www.3ds.com/products-solutions/plm-solutions/enovia-vplm/overview/>.

DAUES, J.; MEEKER, J. Keeping ahead of the CAD/CAM curve. **Aerospace America**, v. 31, p. 20–27, 1993.

DEC. Data Management Yelds a High Return on Investment. **Electronic Design**, p. 88–96, June 1992.

DEKKER, A. H. **Possible Worlds, Belief, and Modal Logic: a Tutorial**. 2004.

DOMAZET, D. S.; LU, S.; KALAJDZIC, M. Concurrent design and process planning of rotational parts. **CIRP Annals**, v. 41, p. 181–184, 1992.

DOWLATSHAHI, S. An integrated manufacturing system design: an applied approach. **International Journal Production Economics**, v. 42, p. 187–199, 1995.

Dublin Core Metadata Initiative. **DCMI Metadata Terms**. 2014. <http://dublincore.org/documents/2012/06/14/dcmi-terms/?v=elements>.

EDGECAM. **EdgeCAM Manual**. 2005. <http://www.edgecam.com>.

EDMONDS, E. A.; CANDY, L.; JONES, R.; SOUFI, B. Support for Collaborative Design: Agents and Emergence. **Communications of the ACM**, v. 37, p. 41–47, 1994.

ENGINEOUS. 2012.

ERMAN, L. D.; LESSER, V. R. A Multi-Level Organization for Problem Solving Using Many, Diverse, Co-operating Sources of Knowledge. **CMUSD**, 1975.

FADDOUL, J. **Reasoning Algebraically With Description Logics**. Tese (Doutorado) — Concordia University, September 2011.

FENNEL, R. **Multiprocess Software Architecture for AI Problem Solving**. [S.l.]: Carnegie-Mellon University, 1975.

FENVES, S. J.; FOUFOU, S.; BOCK, C.; SRIRAM, R. D. CPM2: A Core Model for Product Data. **Journal of Computing and Information Science in Engineering**, March 2008.

FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A.; SIERRA, J.; SIERRA, A. Building a chemical ontology using Methondology and the Ontology Design Environment. **IEEE Intelligent Systems**, v. 14, n. 1, p. 37–46, 1999.

FORTINEAU, V.; PAVIOT, T.; LAMOURI, S. Improving the interoperability of industrial information systems with description logic-based models - The state of the art. **Computers in Industry**, v. 64, p. 363–375, 2013.

FOX, W.; FRIEDRICH, J.; KOPP, R.; AL., T. K. et. The architecture of the object management system within the CADLAB framework 1990. In: **IFIP WG 10.2 Workshop on Electronic Design Frameworks**. [S.l.: s.n.], 1990. p. 141–154.

FRIEDMAN-HILL, E. **Jess in Action: Java Rule-Based Systems (In Action series)**. [S.l.]: Manning Publications, 2003.

FROST, H.; CUTKOSKY, M. **Design for Manufacturability via Agent Interaction**. 1996.

GALLAGHER, K. Q.; CORKILL, D. D.; JOHNSON, P. M. **GBB Reference Manual.** [S.l.], 1988.

GARCIA-CRESPO, A.; RUIZ-MEZCUA, B.; LOPEZ-CUADRADO, J.; GOMEZ-BERBIZ, J. Conceptual model for semantic representation of industrial manufacturing processes. **Computers in Industry**, v. 61, p. 595–612, 2010.

GAŠEVIĆ, D.; DJURIĆ, D.; DEVEDŽIĆ, V. **Model Driven Architecture and Ontology Development**. [S.l.]: Springer, 1998.

GENESERETH, M. E.; FILKES, R. E. **Knowledge Interchange Format Version 3.0 - Reference Manual**. [S.l.], 1992. (Report Logic 92-1).

GILMORE, J. F.; ROTH, S. P.; TYNOR, S. D. **Blackboard Architectures and Applications, A Blackboard System for Distributed Problem Solving**. [S.l.]: Academic Press, 1989.

GIOVANNINI, A.; AUBRY, A.; PANETTO, H.; DASSISTI, M.; El Haouzi, H. Ontology-based system for supporting manufacturing sustainability. **Annual Reviews in Control**, v. 36, n. 2, p. 309–317, 2012.

GRANTA. **Granta - Material Intelligence**. 2006. <http://www.grantadesign.com/>.

GRUBER, T. A Translation Approach to Portable Ontology Specifications. **Knowledge Acquisition**, v. 2, n. 5, p. 199–220, 1993.

GRUBER, T. Formal Ontology in Conceptual Analysis and Knowledge Representation. In: ____. [S.l.]: Kluwer Academic Press, 1993. cap. Toward Principles for the Design of Ontologies Used for Knowledge Sharing.

GRUBER, T. **Ontologies defined**. 1994. Shared Reusable Knowledge Bases (SRKB) Mailing List.

GRUBER, T. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. **IJHCS**, v. 43(5/6), p. 907–928, 1994.

GRUBER, T.; TEINEMBAUM, J.; WEBER, J. Toward a Knowledge Medium for Collaborative Product Development. In: **Proc. of the Second Int. Conf. on Artificial Intelligence in Design**. [S.l.: s.n.], 1992. p. 413–432.

GRUBER, T. R. **Toward Principle for the Design of Ontologies Used for Knowledge Sharing**. 1993. Technical Report KSL 93-04.

GRüNINGER, M. **Using the PSL Ontology**. Toronto, Ontario, Canada, 2007.

GRüNINGER, M.; MENZEL, C. The process specification language (psl) theory and applications. **AI Magazine**, v. 24, n. 3, p. 63–74, 2003.

GUARINO, N. Understanding, building and using ontologies. **International Journal of Human and. Computer Studies**, v. 46, p. 293–310, 1997. A commentary on "Using Explicit Ontologies in KBS Development" by van Heijst, Scheiber, and Wielinga.

GUERRA-ZUBIAGA, D.; YOUNG, R. Information and knowledge interrelationships within a manufacturing knowledge model. **Int. Journal of Advanced Manufacturing Technology**, v. 39, p. 182–198, 2008.

GUO, W.; CHA, J.; XUE, Q.; SI, L. A Case Study in Concurrent Design. In: **ASME Advances in Design Automation**. [S.l.: s.n.], 1993. p. 57–64.

GUPTA, S. K.; NAU, D. S. Systematic approach to analysing the manufacturability of machined parts. **Computer Aided Design**, v. 27, n. 5, p. 232–342, 1995.

HAMMER, P. van den; TREFFERS, M. A data flow control based architecture for CAD frameworks. In: **Proc. ICCAD**. [S.l.: s.n.], 1990. p. 482–485.

HARRISON, D.; MOORE, P.; SPICKELMIER, R.; NEWTON, A. Data management and graphics editing in the Berkeley Design Environment. In: **Proc. of the IEEE ICCAD-86 Conference**. [S.l.: s.n.], 1986. p. 24–27.

HAYES-ROTH, B. A Blackboard Architecture for Control. **AIMag**, v. 26, n. 3, p. 251–321, 1985.

HEIJENGA, W.; JASNOCH, U.; RADEKE, E. Dadamo: a conceptual data model for electronic design applications. In: **Proc. of the European Design Automation Conference**. [S.l.: s.n.], 1992. p. 394–398.

HEIJST, G. V.; SCHREIBER A. WIELINGA, B. Using explicit ontologies in KBS development. **International Journal of Human-Computer Studies**, v. 46, n. 2-3, p. 183–292, 1997.

HENDLER, J. Agents and the Semantic Web. **IEEE Intelligent Systems**, 2001.

HIEKATA, K.; YAMATO, H. A Study on Process Description Method for DFM Using Ontology. In: **Proceedings of the 19th CIRP Design Conference - Competitive Design**. Cranfield University: [s.n.], 2009. p. 210.

HOFSTADTER, D. **Fluid Concepts and Creative Analysis - Computer Models of the Fundamental Mechanisms of Thought**. [S.l.]: Harper Collins, 1994.

HOQUE, A.; HALDER, P.; PARVEZ, M.; SZECSI, T. Integrated manufacturing features and Design-for-manufacture guidelines for reducing product cost under CAD/CAM environment. **Computers & Industrial Engineering**, v. 66, p. 988–1003, 2013.

HORRIDGE, M. **The Manchester OWL Syntax Guide**. [S.l.], 2005.

HORRIDGE, M.; BRANDT, S. **A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.3**. [S.l.], 2011. Sebastian Brandt is a contributor.

HORRIDGE, M.; DRUMMOND, N.; GOODWIN, J.; RECTOR, A.; STEVENS, R.; WANG, H. H. The Manchester OWL Syntax. In: **OWL: Experiences and Directions 2006**. [S.l.: s.n.], 2006.

HOSAKA, M.; KIMURA, F. Model-based approach to CAD/CAM integration. **Computers in Industry**, v. 14, p. 35–42, 1990.

HOWARD, L.; LEWIS, H. The development of a database system to optimise manufacturing processes during design. **Journal of Materials Processing Technology**, v. 134, n. 3, p. 374–382, 2003.

HUANG, G.; HUANG, J.; MAK, K. Agent-based workflow management in collaborative product development on the Internet. **Computer-Aided Design**, v. 32, p. 133–144, 2000.

IBM Inc. **CATIA/CADAM Product Brochure**. 1997.

ISO. **ISO TC184/SC4/WG7 Document N394 - Product Data Representation and Exchange Part 24: Standard Data Access Interface - C Language Late Binding**. 1995. ISO - International Standards Organization Committee Draft.

ISO. **ISO TC184/SC4/WG7 Document N403 - Product Data Representation and Exchange Part 25: C++ Programming Language Binding to the Standard Data Access Interface**. 1995. ISO - International Standards Organization Committee Draft.

ISO. **ISO–10303-21: Industrial automation systems and integration —- Product data representation and exchange —- Part 21: Implementation methods: Clear text encoding of the exchange structure**. 2002. ISO - International Standards Organization.

ISO. **ISO–10303-22: 1998. Industrial automation systems and integration —- Product data representation and exchange —- Part 22: Implementation methods: Standard data access interface**. 2002. ISO - International Standards Organization.

ISO. **ISO–10303-11: Industrial automation systems and integration —- Product data representation and exchange —- Part 11: Description methods: The EXPRESS language reference manual**. 2004. ISO - International Standards Organization.

ISO. **ISO–10303-239: Industrial automation systems and integration —- Product data representation and exchange —- Part 239: Application protocol: Product life cycle support**. 2005. ISO - International Standards Organization.

ISO, I. **ISO-15926:2003 Integration of Lifecycle Data for Process Plant Including Oil and Gas Production Facilities: Part 2 – Data Model**. [S.l.], 2003.

JI, P.; LAU, H. Design for manufacturing: a dimensioning aspect. **Journal of Materials Processing Technology**, v. 91, n. 1-3, p. 121–127, 1999.

JIN, B.; H.F., T.; WANG, Y.; QU, F. Product design reuse with parts libraries and an engineering semantic web for small– and medium–sized manufacturing enterprises. **Int. Journal of Advanced Manufacturing Technology**, v. 38, p. 1075–1084, 2008.

JORGE, V.; REY, V.; MAFFEI, R.; FIORINI, S.; CARBONERA, J.; BRANCHI, F.; MEIRELES, J. ao; FRANCO, G.; FARINA, F.; SILVA, T.; KOLBERG, M.; ABEL, M.; PRESTES, E. Exploring the IEEE ontology for robotics and automations for heterogeneous agent integration. **Robotics anf Computer-Integrated Manufacturing**, 2014.

KATZ, R. A database approach for managing VLSI design data. In: **Proceedings of the 19th ACM/IEEE Design Automation Conference**. [S.l.: s.n.], 1982. p. 274–282.

KATZ, R. Managing the chip design database. **IEEE Computer Magazine**, v. 16, p. 26–35, 1983.

KATZ, R. **Information Management for Engineering Design**. [S.l.]: Springer-Verlag, 1985.

KIM, S.; HOM, S.; PARTHASARATHY, S. Design and manufacturing advisor for turbine disks. In: ____. [S.l.]: Addison-Wesley, 1991. cap. Design for Manufacture, p. 251–230.

KONTCHAKOV, R.; REZK, M.; RODRíGUEZ-MURO, M.; XIAO, G.; ZAKHARYASCHEV, M. The semsantic web - iswc 2014: 13th international semantic web conference. In: ____. [S.l.]: Elsevier, 2014. (Lectures Notes in Computer Science, v. 8796), cap. Answering SPARQL Queries over Databases under OWL 2 QL Entailment Regime, p. 552–567.

KQML Advisory Group. **An Overview of KQML: A Knowledge Query and Manipulation Language**. [S.l.], 1992.

KRIMA, S.; BARBAU, R.; FIORENTINI, X.; SUDARSAN, R.; SRIRAM, R. D. OntoSTEP: OWL-DL ontology for STEP. In: NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **International Conference on Product Lifecycle Management**. [S.l.], 2009.

KRIMA, S.; BARBAU, R.; FIORENTINI, X.; SUDARSAN, R.; SRIRAM, R. D. **OntoSTEP: OWL-DL ontology for STEP**. [S.l.], 2009.

KRIMA, S.; BARBAU, R.; FIORENTINI, X.; SUDARSAN, R.; SRIRAM, R. D. OntoSTEP: OWL-DL ontology for STEP. In: **International Conference on Product Lifecycle Management**. [S.l.: s.n.], 2009.

KRISHNAN, K. **DFM Methodology and Data Representation for Machined Components**. Tese (Doutorado) — Virginia Polytechnic Institute and State University, 1995.

KULON, J.; BROOMHEAD, P.; MYNORS, D. Applying knowledge–based engineering to traditional manufacturing design. **Int. Journal of Advanced Manufacturing Technology**, v. 30, p. 945–951, 2006.

KUO, T.-C.; HUANG, S. H.; ZHANG, H.-C. Design for manufacture and design for 'X': concepts, applications, and perspectives. **Computers and Industrial Engineering**, v. 41, p. 241–260, 2001.

KUOKKA, D.; LIVEZEY, B. **A Collaborative Parametric Design Agent**. [S.l.], 1995.

LÂASRI, H.; MATRÊ, B.; MONDOT, T.; CHARPILLET, F.; HALTON., J. ATOME: A Blackbord Architecture with Temporal and Hypothetical Reasoning. In: **Proceeding of the 8th European Conference on Artificial Intelligence (ECAI-88)**. [S.l.: s.n.], 1988. p. 5–10.

LAI, G.; GUPTA, V.; REDDY, N. V. **Fundamentals of Design and Manufacturing**. [S.l.]: Alpha Science International Ltd., 2005.

LEMAIGNAN, S.; SIADAT, A.; DANTAN, J.-Y.; SEMNENKO, A. MASON: A Proposal For An Ontology Of Manufacuting Domain. In: IEEE. LGIPM - École Nationale Supérieure d'Arts et Métiers, RPK - Universität Karlsruhe (TH): IEEE, 2006.

LIANG, J.; SHAH, J.; SOUZA, R. D.; URBAN, S.; AYYASWAMY, K.; HARTER, E.; BLUM, T. Synthesis of consolidated data schema for engineering analysis from multiple STEP application protocols. **Computer Aided Design**, v. 31, p. 429–447, 1999.

LOFFREDO, D. **Fundamentals of STEP Implementation**. [S.l.], 1999.

MADHUSUDANAN, N.; CHAKRABARTI, A. A questioning based method to automatically acquire expert assembly diagnostic knowledge. **Computer-Aided Design**, v. 57, p. 1–14, 2014.

MAIER, F.; STUMPTNER, M. Enhancements and Ontological Use of ISO-10303 (STEP) to Support the Exchange of Parameterised Product Data Models. In: ADVANCED COMPUTING RESEARCH CENTER, UNIVERSITY OF SOUTH AUSTRALIA. **Seventh International Conference on Intelligent Systems Design and Applications**. [S.l.]: IEEE Computer Society, 2007.

MALONE, B. **ModelCenter: An Integrated Environment for Simulation Based Design**. 2012. <http://www.phoenix-int.com>.

MCGUIRE, J.; KUOKKA, D.; WEBER, J.; TENENBAUM, J.; GRUBER, T.; OLSEN, G. SHADE Technology for Knowledge-based Collaborative Engineering. **Concurrent Engineering: Research and Applications**, v. 1, 1993.

MEGALE, A.; MARTINS, F.; SAKAMOTO, F.; BUENO, A. L.; RODRIGUES, V. Using a blackboard architecture in CAD-CAM systems integration. In: **Proc 3 Int Conf Ind Eng Appl Artif Intell Expert Syst IEA AIE 90**. New York, NY: ACM, 1991.

MEI, J.; BONTAS, E. P.; LIN, Z. OWL2Jess: A Transformational Implementation of the OWL Semantics. In: AL., G. C. et (Ed.). **ISP Workshops 2005, LNCS 3759**. Berlin Heidelberg: Springer-Verlag, 2005.

MENZEL, C.; MAYER, R. **IDEF5 Ontology Description Capture Method Concepts And Formal Foundations**. [S.l.], 1992.

MENZEL, C.; MAYER, R. **The IDEF Family of Languages**. [S.l.], 1998.

MERRIAM-WEBSTER. **The Merriam–Webster Dictionary**. [S.l.]: Merriam–Webster Inc., 2004.

MICHAELIS Dicionário da Língua Portuguesa. [S.l.]: Melhoramentos, 2011.

MIKOS, W. L. **Modelo baseado em agentes em apoio à solução de problemas de não-conformidades em ambientes de manufatura com recursos distribuídos**. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2008.

MO, J.; FERNANDEZ, G.; LLANG, G. An Object Oriented Framework for Intelligent Manufacturing from Computer Aided Features. In: **ASME Computer Applications and Design Features**. [S.l.: s.n.], 1993. p. 123–129.

MONTICOLO, D.; BADIN, J.; GOMES, S.; BONJOUR, E.; CHAMORET, D. A meta-model for knowledge configuration management to support collaborative engineering. **Computers in Industry**, v. 66, p. 11–20, 2015.

MORBACH, J.; YANG, A.; MARQUADT, W. . OntoCAPE – A large-scale ontology for chemical process engineering. **Engineering Applications of Artificial Intelligence**, v. 20, n. 2, p. 147–161, 2007.

MOWBRAY, T. J.; ZAHAVI, R. **The Essential CORBA**. [S.l.]: John Wiley and Sons, 1995.

NEWELL, A. **The knowledge level**. 1981. AI Magazine.

NII, H. Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. **AIMag**, v. 7, n. 2, p. 38–53, 1986.

NII, H.; AIELLO, N. **AGE Attemp to Generalize: Profile of the AGE-0 System**. [S.l.], 1978.

NORTEL. FIPA OS Tutorial Step 1 Generic Agent. Documento eletrônico. 2000.

NORTEL. FIPA OS Tutorial Step 2 Search Agent. Documento eletrônico. 2000.

NORTEL. FIPA OS Tutorial Step 3 Ping Agent. Documento eletrônico. 2000.

NORTEL. FIPA OS Tutorial Step 5 eMarkets. Documento eletrônico. 2000.

NORVIG, P. **Paradigms of Artificial Intelligence Programming - Case Studies in Common Lisp**. [S.l.]: Morgan Kaufmann Publishers, 1992.

NOY, N. F.; MCGUINNESS, D. L. Ontology Development 101: A Guide to Creating Your First Ontology. **Stanford University**, 2004.

NYULAS, C.; O'CONNOR, M.; TU, S.; BUCKERIDGE, D.; OKHMATOVSKAIA, A.; MUSEN, M. An ontology-driven framework for deploying JADE agent systems. In: IEEE (Ed.). **IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology WI-IAT '08**. Sydney, NSW: [s.n.], 2008. Volume 2, p. 573–577.

OBERLE, D.; VOLKS, R.; MOTIK, B.; STAAB, S. Handbook on ontologies. In: ____. [S.l.]: Springer, 2004.

ONECNC. **OneCNC Manual**. 2005. <http://www.onecnc.com>.

ONG, S.; SUN, M.; NEE, A. Y. C. A fuzzy set AHP-based DFM tool for rotational parts. **Journal of Materials Processing Technology**, v. 138, p. 223–230, 2003.

PAHL, G.; BEITZ, W.; FELDHUSEN, J.; GROTE, K.-H. **Engineering Design A Systematic Approach**. Third edition. [S.l.]: Springer, 2007.

PANETTO, H.; ASSISTI, M.; TURSI, A. ONTO–PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. **Advanced Engineering Informatics**, v. 26, p. 334–348, 2012.

PHOENIX INTEGRATION. **ModelCenter Specification**. 2012. <http://www.phoenix-int.com>.

PINTO, H.; MARTINEZ, J. Ontologies: How can they be built? **Knowledge Information Systems**, v. 6, n. 4, p. 441–464, 2004.

PITTET, P.; CRUZ, C.; NICOLLE, C. An ontology change management for facility management. **Computers in Industry**, n. 65, p. 1301–1315, 2014.

PLATZER, A. Lecture Notes on Classical Modal Logic. Lecture notes. 2010.

POLI, C. **Design for Manufacturing: A Structured Approach**. [S.l.]: Butterworth-Heinemann, 2001.

POLI, R. Ontology, step by step. In: PRESS, I. (Ed.). **New Trends in Software Methodologies, Tools and Techniques**. [S.l.: s.n.], 2003. Proceedings of Lyee, p. 66–70.

POLLOCK, J. T. **Web Semântica para Leigos**. [S.l.]: Alta Books, 2010.

PONZETTO, S. P.; STUBE, M. Taxonomy induction based on a collaborative built knowledge repository. **Artificial Intelligence**, v. 175, p. 1737–1756, 2011.

POSADA, J.; TORO, C.; WUNDRAK, S.; STORK, A. Using ontologies and STEP standards for the semantic simplification of CAD models in different engineering domains. **Applied Ontology**, v. 1, p. 263–279, 2005–2006.

POSADA, J.; WUNDRAK, S.; STORK, A.; TORO, C. Semantically controlled LMV techniques for plant design review. In: **Proceedings of DETC/CIE 2004 ASME 2004 Design Engineering Technical Conferences 24th Computers and Information in Engineering (CIE) Conference**. Salt Lake City, Utah: [s.n.], 2004.

POSTGRESQL. **PostgreSQL Manuals**. 2015. <http://www.postgresql.org/docs/manuals/>.

PRATT, M. J. **Introduction to ISO 10303 - the STEP Standard for Product Data Exchange**. [S.l.], 2001.

PRESTES, E.; CARBONERA, J.; FIORINI, S.; JORGE, V.; ABEL, M.; MADHAVAN, R.; LOCORO, A.; GONçALVES, P.; BARRETO, M.; HABIB, M.; CHIBANI, A.; GRARD, S.; AMIRAT, Y.; SCHLENOFF, C. Towards a core ontology for robotics and automation. **Robotics and Autonomous Systems**, <http://dx.doi.org/10.1016/j.robot.2013.04.005>, 2013.

PROTÉGÉ TEAM. **Protégé is a free, open source ontology editor and knowledge-base framework**. 2014. <http://protege.stanford.edu/>.

PTC. T072-20-02. **Introduction to Pro/ENGINEER Release 20.0**. [S.l.]: Parametric Technology Corporation, 1999.

PTC. **Getting Started with Pro/ENGINEER 2001 - A Tutorial-Based Guide to the Pro/ENGINEER Workflow**. [S.l.]: Parametric Technology Corporation, 2001.

PTC. **Getting Started with Pro/ENGINEER Wildfire 2.0 - A Tutorial-based Guide to Workflow**. [S.l.]: Parametric Technology Corporation, 2004.

PTC. **J/Link User Guide**. [S.l.]: Parametric Technology Corporation, 2004.

PTC. **Pro-Engineer Product Brochure**. [S.l.]: Parametric Technology Corporation, 2012. Parametric Technology Corporation.

RAHMANI, K.; THOMSON, V. Ontology-based interface design and control methodology for collaborative product development. **Computer-Aided Design**, v. 44, p. 432–444, 2012.

RAMOS, A. L. T.; DEISENROTH, M. P. STEP as the data base of a Design for Manufacturing (DFM) framework. **Third Internation Conference on Production Research – Americas Region 2006 (ICPR–AM06)**, 2006.

RESNIK, P. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In: MELLISH, C. (Ed.). **14th International Joint Conference on Artificial Inteligence, IJCAI**. [S.l.]: Morgan Kaufmann Publishers Inc, 1995. v. 1, p. 448–453.

RICO, M.; CALIUSCO, M. L.; CHIOTTI, O.; GALLI, M. R. OntoQualitas: A framework for ontology quality assessment in information interchanges between heterogeneous systems. **Computers in Industry**, n. 65, p. 1291–1300, 2014.

RICOA, M.; CALIUSCOA, M.; CHIOTTIAB, O.; GALLI, M. An approach to define semantics for BPM systems interoperability. **Enterprise Information Systems**, p. 1–24, March 2013. Disponível em: <http://dx.doi.org/10.1080/17517575.2013.767381>.

ROCCA, G. L. Knowledge based engineering: Between AI and CAD. review of a language based rechnology to support engineering design. **Advanced Engineering Informatics**, v. 26, p. 159–179, 2012.

ROSEN, D.; DIXON, R. J.; POLI, C. Features and algorithms for tooling cost evaluation in injection molding and die casting. **Computers in Engineering**, v. 1, p. 45–52, 1992.

SAMBU, S. P. **A Design for Manufacture Method for Rapid Prototyping and Rapid Tooling**. Tese (Doutorado) — Georgia Tech, 2001.

SÁNCHEZ, D.; BATET, M.; ISERN, D.; VALLS, A. Ontology–based semantic similarity: A feature–based approach. **Expert Systems with Applications**, v. 39, p. 7718–7728, 2012.

SANCHEZ, J. M.; PRIEST, J. W.; SOUTO, R. Intelligent Reasoning Assistant for Incorporating Manufacturability Issues into the Design Process. **Expert Systems with Applications**, v. 12, n. 1, p. 81–88, 1997.

SAPOSSNEK, M.; TALUKDAR, S.; ELFES, A.; SEDAS, S.; EISENBERGER, M.; HOU, L. Design Critics in the Computer-Aided Simultaneous Engineering (CASE) project. In: **Proc. ASME Winter Annual Meeting on Concurrent Product and Process Design**. [S.l.: s.n.], 1989. p. 137–141.

SARDER, M. B. **The development of a design ontology for products and processes**. Tese (Doutorado) — THE UNIVERSITY OF TEXAS AT ARLINGTON, 2006.

SCHLENOFF, C.; IVESTER, R.; KNUTILLA, A. A robust process ontology for manufacturing systems integration. In: SOCIETY, I. C. (Ed.). **DEXA '02 Proceedings of the 13th International Workshop on Database and Expert Systems Aplications**. Washington, DC, USA, 2002. p. 597–602.

SCHLENOFF, C.; PRESTES, E.; MADHAVAN, R.; GONçALVES, P.; LI, H.; BALAKIRSKY, S.; KRAMER, T.; MIGUELANEZ, F. An IEEE standard ontology for robotics and automation. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems**. Vilamoura, Algarve, Portugal: Springer-Verlag, 2012. p. 1337–1342.

SDRC. **I-DEAS User Manual**. [S.l.]: Structural Dynamics Research Corporation, 1996.

SDRC. **I-DEAS Master Series 5 Product Brochure**. 1997.

SDRC. **Metaphase 2 Product Brochure**. 1997.

SEIDWITZ, E. What models mean. **IEEE Software**, v. 20, n. 5, p. 26–32, 2003.

SELIC, B. The pragmatics of model-driven development. **IEEE Software**, v. 20, n. 5, p. 19–25, 2003.

SHAW, M.; GARLAN, D. **Software Architecture: Perspectives on an Emerging Discipline**. [S.l.]: Prentice Hall, 1996.

SHEN, W.; BARTHES, J. DIDE: A Multi Agent Environment for Engineering Design. In: **Proc. First International Conference on Multi Agent Systems**. [S.l.: s.n.], 1994. p. 344–351.

SIM, S. K.; DUFFY, A. H. Towards an ontology of generic engineering design activities. **Research in Engineering Design**, v. 14, p. 200–223, 2003.

Software Productivity Solutions. **Catalyst Training Manual**. [S.l.], 1997.

STEP Tools. **STEP Tutorials - AP-203 Basic CC1 Data, Geometry Report, Assembly Structure Data, Bill-of-Materials Assembly Report, Design Activity Data**. 2004. <http://www.steptools.com>.

STOLL, W. **Design for Manufacture: Strategies, Principles and Techniques**. [S.l.]: Addison-Wesley, 1991. Design for Manufacture: An Overview.

SUDARSAN, R.; FENVES, S.; SRIRAM, R.; WANG, F. A product information modeling framework for product lifecycle management. **Computer–Aided Design**, v. 37, p. 1399–1411, 2005.

SUN, H.; FAN, W.; SHEN, W.; XIAO, T. Ontology-based interoperation model of collaborative product development. **Journal of Network and Computer Applications**, v. 35, p. 132–144, 2012.

TAILOR, A. **Blackboard Systems - MXA - A Blackboard Expert System Shell**. [S.l.]: Addison-Wesley, 1988.

TIAN, Y.; ZOU, H.; GUO, W. An integrated knowledge representation model for the computer–aided conceptual design of mechanisms. **Int. Journal Advanced Manufacturing Technology**, v. 28, p. 435–444, 2006.

TSAI, J.-C.; CHUANG, T.-C.; GUO, D.-N. Development of a STEP-based Dimensioning and Tolerancing Data Model. **Proceedings of National Science Council ROC(A)**, v. 22, n. 6, p. 831–840, 1998.

TVERSKY, A. Features of similarity. **Psycological Review**, n. 1984, p. 327–352, 1977.

UGS. **Gerenciamento do Ciclo de Vida do Produto**. 2006. <http://www.ugs.com/brasil/plm/plm_gerenciar.shtml>.

UGS. **UGS product overview, descriptions, and details**. [S.l.]: NX, 2006. <http://www.ugs.com/products/nx/>.

W3C Member Submission. **<http://www.w3.org/Submission/SWRL>**. 2004.

WAND, Y.; WEBER, R. An Ontological Model of an Information System. **IEEE Transactions of an Information System**, v. 16, n. 11, November 1990.

WARMAN, E. A. Integration revisited. An appraisal of the state of the integration of CAD. **Computers in Industry**, v. 14, p. 59–65, 1990.

WEI, S.; QIN-YI, M.; TIAN-YI, G. An Ontology-based Manufacturing Design System. **Information Technology Journal**, v. 8, n. 5, 2009.

WILENSKI, R. **Common LISPcraft**. [S.l.]: Norton, 1986.

Proceeding Selected and Expanded Papers from the IFIP TC5/WG5.2 Working Conference on Geometric Modeling for Product Realization. The Netherlands: North-Holland Publishing Co., 1992. 267-296 p.

WINSTON, P.; HORN, B. **LISP**. 3rd. ed. [S.l.]: Addison Wesley, 1989.

WITHEREL, P.; KRISHNAMURTY, S.; GROSSE, I. Ontologies for supporting engineering design optimization. **Journal of Computing and Information Science in Engineering**, v. 7, n. 2, p. 141–50, 2007.

WOLF, P. van der. **CAD Frameworks - Principles and Architecture**. [S.l.]: Kluwer Academic Publishers, 1994.

World Wide Web Consortium. **XSL Transformations - XLST**. 1999. <http://www.w3.org/TR/xslt>.

World Wide Web Consortium. **Extensible Markup Language (XML) 1.0**. 2000. <http://www.w3.org/TR/2000/REC-xml-20001006>.

WOYAK, S. **An Object-Oriented Methodology and Support Framework for Creating Engineering Software Using Dynamic Integration**. Tese (Doutorado) — Virginia Polytechnic Institute and State University, 1995.

WOYAK, S. A.; MYKLEBUST, A. Functionality and data integration of software modules through dynamic integration. **Journal of Engineering Design**, v. 9, n. 2, p. 137–158, June 1998.

WU, S.; GHENNIVA, H.; ZHANG, Y.; SHEN, W. Personal assistant agents for collaborative design environments. **Computers in Industry**, v. 57, p. 732–739, 2006.

XIAO, W.; ZHENG, L.; HUAN, J.; LEI, P. A complete 'cad/cam/cnc' solution for 'step'-compliant manufacturing. **Robotics and Computer-Integrated Manufacturing**, v. 31, p. 1–10, 2015.

YANG, D.; DONG, M.; MIAO, R. Development of a product configuration system with an ontology-based approach. **Computer Aided Design**, v. 40, p. 863–878, 2008.

YIM, S. **An Information Infrastructure (DFM Framework) for Distributed Design for Addictive Manufacturing**. Dissertação (Mestrado) — Georgia Tech, 2005.

YOUNG, R. Informing decision–makers in product design and manufacture. **Int. Journal of Computer Integrated Manufacturing**, v. 16, n. 6, p. 428–438, 2003.

ZANCONATTO, R. 335-345. **Blackboard Systems - BLOBS: An Object Oriented Blackboard Framework for Reasoning in Time**. [S.l.]: Addison-Wesley, 1988.

ZEID, I. **CAD/CAM Theory and Practice**. [S.l.]: McGraw-Hill, 1991.

ZHA, X.; DU, H. A PDES/STEP based model and system for concurrent integrated design and assembly planning. **Computer Aided Design**, v. 34 (14), p. 1087–1110, 2002.

ZHAO, W.; LIU, J. OWL/SWRL representation methodology for EXPRESS–driven product information model. part I: Implementation methodology. **Computers in Industry**, v. 59, p. 580–589, 2008.

ZHAO, W.; LIU, J. OWL/SWRL representation methodology for EXPRESS–driven product information model. part II: practice. **Computers in Industry**, v. 59, p. 590–600, 2008.

ZHAO, Z.; SHAH, J. Domain independent shell for DFM and its application to sheet metal forming and injection molding. **Computer-Aided Design**, CAD' 04 Special Issue: Product Design, Integration and Manufacturing, n. 37, p. 881–898, 2005.

APÊNDICE A – INTRODUÇÃO À LINGUAGEM OWL

A linguagem mais tipicamente utilizada para definição e edição de ontologias é OWL (*Ontology Web Language*), que possui sintaxe RDF/XML. Atualmente, esta linguagem é a que possui as características mais adequadas em termos de sintaxe porém, ao mesmo tempo, tem muita verbosidade e pode ser deveras complexa dependendo da informação representada [Horridge *et al.*, 2006]. Adicionalmente, esta linguagem é sintaticamente prefixada, fato que pode ocasionar interpretação ambígua da compreensão adequada do significado da informação; senão vejamos um exemplo,

$$\exists \text{ hasPart Product} \tag{A.1}$$

A interpretação adequada desta asserção não é "direta" devido tanto a aspectos sintáticos quanto relacionados com a verbosidade, logo sua compreensão pode ser complexa. No domínio DFM, por exemplo, a equação A.1 pode ser interpretada como "alguns produtos possuem peças" quando sua asserção correta é "todos produtos possuem peças". Esta limitação existe devido ao fato da linguagem OWL ser derivada de linguagem de Descrição Lógica (*Description Logics – DL*), onde as asserções são mais genéricas. De forma a tornar este tipo de representação ótima, foi desenvolvida a sintaxe Manchester OWL [Horridge, 2005]. A tabela A.1 descreve, sucintamente, as diferenças sintáticas entre as variantes desta linguagem.

Tabela A.1 – Primitivas construtoras de classes OWL e suas diferentes sintaxes.

Primitiva construtora OWL	Sintaxe DL	Manchester OWL	Exemplo
intersectionOf	C ∧ D	C **AND** D	Projetista (agente): AID and Compute and Personnel
unionOf	C ∨ D	C **OR** D	Dimensão: MfgDimension or NomDimension
complementOf	¬ C	**NOT** C	not Hole
oneOf	{a} ∨ {b}	{a b. . . }	Peça: ((hasFeature some Hole) or (hasFeature some Prismatic) or (hasFeature some Rotational))
someValuesFrom	∃ R C	R **SOME** C	*Feature*: Dimension and NomDimension and Part and ((hasFeature some Hole) or (hasFeature some Prismatic) or (hasFeature some Rotational))
allValuesFrom	∀ R C	R **ONLY** C	Peça: hasFeature only (Part and (Hole or Prismatic or Rotational))
minCardinality	≥ N R	R **MIN** N	Produto: Concept and (hasPart min 1 Part)
maxCardinality	≤ N R	R **MAX** N	Propriedade: Property and (hasLimit max 2 Limit)
cardinality	= N R	R **EXACTLY** N	Limites: Limit exactly (1 UpperLimit and 1 LowerLimit)
hasValue	∃ R {a}	R **VALUE** A	Constante: isPhysicalConstant and Real and 9,80665

APÊNDICE B – DESCRIÇÃO TEXTUAL DETALHADA DAS CLASSES, PROPRIEDADES E INDIVÍDUOS

Em uma arquitetura de informação é importantíssimo que o significado de cada variável representacional seja descrito corretamente sob pena de sua compreensão adequada não ocorrer. Assim sendo, este apêndice descreve sob termos análogos aos utilizados por humanos a correta representação das classes, propriedades e indivíduos do capítulo 4 e capítulo 5 desta pesquisa.

Adicionalmente, é relevante destacar que existem diferentes formatos de texto para descrição de uma ontologia, os quais variam em termos de sua verbosidade e detalhamento. As listas abaixo estão descritas segundo o formato RDF sem os detalhes sintáticos iniciais do arquivo, exceto a parte inicial da taxonomia, que está descrita na seção B.1 abaixo para evitar repetição. Neste contexto, o prefixo URI, ou IRI, <http://jade.cselt.it/beangenerator> também foi retirado das definições de classes, propriedades, indivíduos e dados para simplificar a legibilidade. A parte inicial do arquivo descreve tanto aspectos genéricos da ontologia quanto detalhes sobre o(s) espaço(s) de nome(s) utilizados, que descrevem, por exemplo, o(s) vocabulário(s). A taxonomia descrita, que é uma subparte da taxonomia completa, foi subdividida de forma a facilitar sua compreensão.

B.1 Taxonomia anotada - introdução

Lista B.1 – Parte inicial da taxonomia anotada dos componentes DFM básicos.

```
Prefix: owl: <http://www.w3.org/2002/07/owl#>
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>

Ontology: <http://www.semanticweb.org/andreltr/ontologies/2015/2/5/dfmonto>

Annotations:
    rdfs:seeAlso _:http://jade.cselt.it/beangenerator#genid10364635455155,
    owl:versionInfo "0.5"^^xsd:string,
    rdfs:seeAlso <http://gaper.swi.psy.uva.nl/beangenerator>,
    rdfs:comment "The Design For Manufacturing (DFM) Ontology tries to
        structure
         information used in DFM using intrinsic knowledge associated with it
              plus
         actual information from the STEP architecture model."^^xsd:string,
    rdfs:seeAlso <http://jade.cselt.it/beangenerator#DFM>,
    rdfs:comment "A DFM Ontology for prismatic parts subject to machining
         processes."^^xsd:string,
    rdfs:isDefinedBy "Andre Luiz Tietbohl Ramos"^^xsd:string,
    rdfs:comment "DFM ontology
         The DFM ontology aims to provide a foundation for the devolpment of
             a common
         information structure in this filed.  Ideally, it will be integrated
              with
         the STEP plugin developed by NIST in order to use the current
             standard
         exchange in CAD/CAM/CAE.  Finally, in order to achieve an actual
             system
         integration the bean generator is used for the JADE agent
         platform."^^xsd:string, rdfs:isDefinedBy "Chris van Aart"^^xsd:
             string,
    owl:versionInfo "1.0"^^xsd:string
AnnotationProperty: owl:versionInfo
AnnotationProperty: rdfs:seeAlso
AnnotationProperty: rdfs:isDefinedBy
AnnotationProperty: rdfs:label
AnnotationProperty: <http://purl.org/dc/elements/1.1/description>
Annotations:
    rdfs:isDefinedBy <http://purl.org/dc/elements/1.1/>,
    rdfs:label "Description"@en-us,
    rdfs:comment "An account of the content of the resource."@en-us,
         <http://purl.org/dc/elements/1.1/description> "Description may
             include
         but is not limited to: an abstract, table of contents, reference to
             a
         graphical representation of content or a free-text account of the
         content."@en-us
Annotations:
        <http://purl.org/dc/elements/1.1/description> "Description may
            include
        but is not limited to: an abstract, table of contents, reference to a
```

```
        graphical representation of content or a free-text account of the
45      content."@en-us,
        rdfs:comment "An account of the content of the resource."@en-us,
        rdfs:isDefinedBy <http://purl.org/dc/elements/1.1/>,
        rdfs:label "Description"@en-us
AnnotationProperty: rdfs:comment
```

B.2 Taxonomia hierárquica anotada

Lista B.2 – Taxonomia hierárquica anotada dos componentes DFM básicos.numbers

```
Class: <#ManufacturingFeature>
    Annotations:
        rdfs:comment "Ensures the limits of manufacturing operation and
            design limits exist:
hasLimitsMfgOperation some
    (DesignLowerLimit
     and MfgLowerLimit)",
        rdfs:comment "Manufacturing feature must include material aspects!"^^
            xsd:string,
        rdfs:comment "Ensures the limits of manufacturing operation and
            design limits exist:
hasLimitsMfgOperation some
    (DesignUpperLimit
     and MfgUpperLimit)"
    SubClassOf:
        <#Approximate_ratio>,
        <#Precautions>,
        <#TypeOfFeature>,
        <#Tool>,
        <#hasLimitsMfgOperation> some
            (<#DesignedUpperLimit>
             and <#MfgUpperLimit>),
        <#hasToleranceDefined> some <#ToleranceDimension>,
        <#Product>,
        <#hasFeature> only
            (<#HoleFeature>
             or <#PrismaticFeature>
             or <#RotationalFeature>),
        <#Tolerance>,
        <#Cost>,
        <#Cutting_nature>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#hasLimitsMfgOperation> some
            (<#DesignedLowerLimit>
             and <#MfgLowerLimit>)
Class: <#Standardization>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#Surface>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
```

```
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
    DisjointWith:
        <#Simplicity>, <#DimensionalFlexibility>, <#Machinability>
Class: <#Face>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#hasOperation> min 1 <#Face>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#HollowShape>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#TypeOfFeature>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
```

```
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Precautions>
    SubClassOf:
        <#is_numeric_array> only (xsd:double or xsd:float),
        <#hasMaterialCharacteristic> only <#Precautions>,
        <#MaterialCharacteristics>
Class: <#ProcessPrecautions>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        owl:Thing,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>,
        <#hasProcess> min 1 <#ProcessPrecautions>
Class: <#Hobbing>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
```

```
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Shaping_(gear)>
    SubClassOf:
        <#Planning_and_shaping>
Class: <#ToolCost>
    SubClassOf:
        <#hasToolCost> only <#ToolCost>,
        <#AccessoriesCost>
Class: <#InitialCost>
    EquivalentTo:
        <#Cost>
         and (<#hasMachineFixedCostDefined> some <#MachineFixedCost>)
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Plastic_Processing>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
```

```
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#2Dcontoured>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#Surface>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#AgentAction>
    SubClassOf:
        <#Concept>
Class: <#LocationTolerance>
    SubClassOf:
        <#TypeOfTolerance>
Class: <#ExternalCylindrical>
    SubClassOf:
        <#Cylindrical>
Class: <#Cost>
    SubClassOf:
        owl:Thing
Class: <#Semi-finish>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
```

```
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#PrismaticFeature>
    EquivalentTo:
        <#GeometryFeature>
         and (<#hasFeature> only <#PrismaticFeature>)
         and (<#hasFeature> exactly 1 <#Feature>)
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#Tool>
Class: <#Machining>
    Annotations:
        rdfs:comment "DFM aspects related with the Material for machining:
             - Cutting nature
             - Expected finish
             - Approximate ratio"^^xsd:string
    SubClassOf:
        <#Precautions>,
        <#Tool>,
        <#hasMaterialCharacteristic> some <#Appearance>,
        <#hasMaterialCharacteristic> some <#ElectricalConductivity>,
        <#Operation>,
        <#Tolerance>,
        <#Cost>,
        <#hasMaterialCharacteristic> some <#Expected_finish>,
        <#Machining>,
        <#hasMaterialCharacteristic> some <#Precautions>,
        <#Expected_finish>,
        <#hasMaterialCharacteristic> some <#Stiffness>,
        <#hasMaterialCharacteristic> some <#MaterialIdentifier>,
        <#hasMaterialCharacteristic> some <#UNS>,
        <#hasMaterialCharacteristic> some <#Strenght>,
        <#hasMaterialCharacteristic> some <#Base_metal>,
        <#Approximate_ratio>,
        <#hasMaterialCharacteristic> some <#Approximate_ratio>,
        <#hasMaterialCharacteristic> some <#MaterialCharacteristics>,
        <#hasMaterialCharacteristic> some <#Alloy>,
        <#hasMaterialCharacteristic> some <#CorrosionResistance>,
        <#Product>,
        <#hasMaterialCharacteristic> some <#SurfaceRoughness>,
```

```
        <#Cutting_nature>,
        <#hasMaterialCharacteristic> some <#Cutting_nature>,
        <#NominalDimension>,
        <#hasMaterialCharacteristic> some <#Formability>,
        <#hasMaterialCharacteristic> some <#MaterialProperty>
Class: <#MaterialIdentifier>
    SubClassOf:
        <#Material>
Class: <#Linear>
    SubClassOf:
        <#SizeTolerance>
Class: <#Line_profile>
    SubClassOf:
        <#ProfileTolerance>
Class: <#JigCost>
    SubClassOf:
        <#hasJigCost> only <#JigCost>,
        <#AccessoriesCost>
Class: <#Dimension>
    SubClassOf:
        owl:Thing
Class: <#Chemical>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#TypeOfFeature>,
        <#Surface>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#HandReaming>
    SubClassOf:
        <#Reaming>
Class: <#Diameter>
    SubClassOf:
        <#Boring>
Class: <#Straddle>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
```

```
        <#Process>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#hasOperation> min 1 <#Straddle>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#UNS>
    SubClassOf:
        <#MaterialIdentifier>,
        <#is_string_array> only xsd:string,
        <#hasMaterialIdentifier> only <#UNS>
Class: <#Designer>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#Personnel>,
        <#AID>
         and <#Personnel>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#hasActivity> some
            (<#Tolerancing>
             or <#ToolAccessibility>),
        <#AID>,
        <#DFMPrinciples/rules>,
```

```
        <#DesignedTolerance>
Class: <#Reaming>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Process>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#AccessoriesCost>
    SubClassOf:
        <#Cost>,
        <#hasCost> only
            (<#FixtureCost>
             and <#JigCost>
             and <#ToolCost>)
Class: <#Internal_Broaching>
    SubClassOf:
        <#Broaching>
Class: <#NominalDimension>
    SubClassOf:
        <#Dimension>,
        <#hasDimension> only <#NominalDimension>
Class: <#End>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#Surface>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
```

```
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Symmetricity>
    SubClassOf:
        <#LocationTolerance>
Class: <#ToolAccessibility>
    Annotations:
        rdfs:comment "DFM aspects related to Material:"^^xsd:string,
        rdfs:comment "Informs if the tool is accessible to manufacture a part
            /assembly."^^xsd:string
    EquivalentTo:
        <#AgentAction>
         and <#DFMActivity>
         and (<#hasLimitsMfgOperation> some
            (<#DesignedLowerLimit>
             and <#MfgLowerLimit>))
         and (<#hasRole> some
            (<#Designer>
             or <#Manufacturer>))
         and (<#hasToolAccessibility> some
            (<#GeometryFeature>
             and <#ManufacturingFeature>))
         and (<#hasMfgProcessDefined> only
            (<#End_(slot_widths)>
             or <#Face>
             or <#Slotting_(width)>
             or <#Straddle>))
         and (<#hasOperation> only <#MillingSingleCut>)
         and (<#hasRole> only
            (<#Designer>
             or <#Manufacturer>))
         and (<#hasToleranceDefined> only
            ((<#DesignedLowerLimit>
             and <#DesignedUpperLimit>)
             or (<#NominalDimension>
             and <#ToleranceDimension>)))
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#Personnel>,
        <#ToleranceDimension>,
        <#Surface>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
```

```
        <#GeometryFeature>,
        <#AgentAction>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Flatness>
    SubClassOf:
        <#ShapeTolerance>
Class: <#Surface_(thickness)_broaching>
    SubClassOf:
        <#Broaching>
Class: <#ElectricalConductivity>
    SubClassOf:
        (<#is_float> only xsd:float)
         and (<#is_string> only xsd:string),
        <#MaterialProperty>,
        <#hasMaterialProperty> only <#ElectricalConductivity>
Class: <#OrientationTolerance>
    SubClassOf:
        <#TypeOfTolerance>
Class: <#SlotPrismatic>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
        <#Process>,
        <#Surface>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: owl:Thing
Class: <#MfgLowerLimit>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
```

```
        <#TypeOfFeature>,
        <#Surface>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Forging>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#ToleranceDimension>
    Annotations:
        rdfs:comment "Tolerance types"^^xsd:string
    SubClassOf:
        <#Product>,
        <#Dimension>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#NominalDimension>,
```

```
        <#Expected_finish>,
        <#Tool>,
        <#hasToleranceDefined> some
            ((<#DesignedLowerLimit>
             and <#DesignedUpperLimit>)
             or (<#NominalDimension>
             and <#ToleranceDimension>)),
        <#is_float> some xsd:float
Class: <#Shoulder_location_(forming)>
    SubClassOf:
        <#Screwing>
Class: <#DesignerCost>
    SubClassOf:
        <#PersonnelCost>
Class: <#InternalCylindrical>
    SubClassOf:
        <#Cylindrical>
Class: <#Surface>
    SubClassOf:
        <#Product>,
        <#Feature>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Precautions>,
        <#Cutting_nature>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#Tool>
Class: <#MillingSingleCut>
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Rough>,
        <#Cutting_nature>,
        <#Precautions>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#Tool>,
        <#hasOperation> min 1 <#MillingSingleCut>
Class: <#Fixture>
    SubClassOf:
        <#hasFixtureCost> only <#Fixture>,
        <#Accessories>
Class: <#Material>
    Annotations:
        rdfs:comment "Manufacturing material."^^xsd:string
    SubClassOf:
        <#hasMaterial> only <#Material>
Class: <#SizeTolerance>
    SubClassOf:
        <#TypeOfTolerance>
Class: rdf:Property
Class: <#Center-type_and_center-less>
    SubClassOf:
```

```
        <#Grinding>
Class: <#DesignedUpperLimit>
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Precautions>,
        <#Cutting_nature>,
        <#hasLimitsMfgOperation> only
            (<#DesignedUpperLimit>
             and <#MfgUpperLimit>),
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>,
        <#DesignedTolerance>
Class: <#Rough>
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Precautions>,
        <#Cutting_nature>,
        <#Machining>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>
Class: <#Assembly>
    SubClassOf:
        <#Product>,
        <#hasAssembly> min 2 <#Part>
Class: <#External_forming>
    SubClassOf:
        <#Screwing>
Class: <#Deep_Drawing>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
```

```
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Manufacturer>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#AID>
         and <#Personnel>,
        <#ToleranceDimension>,
        <#hasActivity> some
            (<#AvailabilityOfMachinesAndTools>
             or <#ToolAccessibility>),
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#MillingSingleCut>,
        <#DFMActivity>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>,
        <#Personnel>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#GeometryFeature>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#AID>
Class: <#Jig>
    SubClassOf:
        <#hasJigCost> only <#Jig>,
        <#Accessories>
Class: <#JADE-SLOT>
    SubClassOf:
        rdf:Property
Class: <#DesignedTolerance>
    Annotations:
        rdfs:comment "Designed tolerance for a part."^^xsd:string
    SubClassOf:
        <#Product>,
        <#is_tol_mfg_interval> some xsd:float,
        <#Dimension>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Precautions>,
        <#Cutting_nature>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#Tool>
```

```
Class: <#Cylindricity>
    SubClassOf:
        <#ShapeTolerance>
Class: <#MfgUpperLimit>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Formability>
    SubClassOf:
        <#MaterialProperty>,
        <#hasMaterialProperty> only <#Formability>,
        (<#is_float> only xsd:float)
         and (<#is_numeric_array> only xsd:float)
         and (<#is_string_array> only xsd:string)
Class: <#Casting>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
```

```
        <#DesignedLowerLimit>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Slotting_(width)>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Process>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#hasOperation> min 1 <#Slotting_(width)>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#TypeOfFeature>
    SubClassOf:
        <#Product>,
        <#Feature>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>
Class: <#Finish>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
        <#Surface>,
        <#Process>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
```

```
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Parallelism>
    SubClassOf:
        <#OrientationTolerance>
Class: <#Strenght>
    SubClassOf:
        <#is_float> only xsd:float,
        <#MaterialProperty>,
        <#hasMaterialProperty> only <#Strenght>
Class: <#Total>
    SubClassOf:
        <#Run-outTolerance>
Class: <#RoundHole>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
        <#Process>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Shoulder_location_(turning)>
    SubClassOf:
        <#Screwing>
Class: <#OperationType>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#TypeOfFeature>,
```

```
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Cylindrical>
    SubClassOf:
        <#Grinding>
Class: <#Alloy>
    SubClassOf:
        <#MaterialIdentifier>,
        <#is_string_array> only xsd:string,
        <#hasMaterialIdentifier> only <#Alloy>
Class: <#Jet_Cutter>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Blind>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#TypeOfFeature>,
```

```
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Simplicity>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#Process>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
    DisjointWith:
        <#Standardization>, <#DimensionalFlexibility>, <#Machinability>
Class: <#Cutting_nature>
    SubClassOf:
        <#is_string_array> only xsd:string,
        <#hasMaterialCharacteristic> only <#Cutting_nature>,
        <#MaterialCharacteristics>
Class: <#Through>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#TypeOfFeature>,
```

```
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Secondary>
    SubClassOf:
        <#hasOperation> only <#Secondary>,
        <#OperationType>
    DisjointWith:
        <#Primary>
Class: <#DFM>
    SubClassOf:
        owl:Thing,
        <#hasProcess> some <#Reaming>,
        <#Precautions>,
        <#hasProcess> some <#Screwing>,
        <#hasProcess> some <#Burnishing>,
        <#Tool>,
        <#hasProcess> some <#Hobbing>,
        <#hasProcess> some <#HandReaming>,
        <#Tolerance>,
        <#hasCostDefined> some <#ComputeCost>,
        <#hasProcess> some <#Rough>,
        <#Cost>,
        <#hasProcess> some <#MillingSingleCut>,
        <#hasAvailabityOfMachinesTools> some <#AvailabilityOfMachinesAndTools
            >,
        <#Expected_finish>,
        <#hasProcess> some <#Honing>,
        <#hasProcess> some <#Boring>,
        <#hasProcess> some <#Broaching>,
        <#hasProcess> some <#Shoulder_depth>,
        <#hasProcess> some <#Machining>,
        <#Approximate_ratio>,
        <#hasActivity> some <#Tolerancing>,
        <#hasProcess> some <#MachineReaming>,
        <#hasProcess> some <#Semi-finish>,
        <#hasProcess> some <#Drilling>,
        <#hasProcess> some <#Turning>,
        <#hasPart> some <#Part>,
        <#Product>,
        <#hasTolerance> some <#Tolerancing>,
        <#hasProcess> some <#Finish>,
        <#Cutting_nature>,
```

```
        <#hasToolAccessibility> some <#ToolAccessibility>,
        <#hasFeature> some <#Feature>,
        <#hasProcess> some <#Planning_and_shaping>,
        <#NominalDimension>,
        <#hasProcess> some <#Grinding>,
        <#hasProcess> some <#Chemical>
Class: <#Supervisor>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#Personnel>,
        <#AID>
         and <#Personnel>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#TypeOfFeature>,
        <#Process>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#AID>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>,
        <#hasActivity> some
            (<#AvailabilityOfMachinesAndTools>
             or <#ComputeCost>)
Class: <#TypeOfTolerance>
    SubClassOf:
        <#Tolerance>
Class: <#Hollow>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#hasOperation> min 1 <#Hollow>,
        <#Surface>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
```

```
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Broaching>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
        <#Process>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#External_shaving>
    SubClassOf:
        <#Screwing>
Class: <#Machinability>
    Annotations:
        rdfs:comment "DFM aspects related with the Material for machining:
            - Cutting nature
            - Expected finish
            - Approximate ratio"^^xsd:string
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#hasToleranceDefined> some
            (<#DesignedLowerLimit>
             and <#DesignedUpperLimit>),
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#MillingSingleCut>,
        <#DFMActivity>,
        <#Operation>,
        <#hasMachineAccessory> only
```

```
            (<#Tool>
             and (<#Fixture>
             or <#Jig>)),
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#hasProcess> only <#MillingSingleCut>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#hasTypeOfToleranceDefined> only
            (<#Angular>
             or <#Angularity>
             or <#Circular>
             or <#Circularity>
             or <#Concentricity-coaxiality>
             or <#Cylindricity>
             or <#Diameter-Radius>
             or <#Flatness>
             or <#Line_profile>
             or <#Linear>
             or <#Parallelism>
             or <#Perpendicularity>
             or <#Position>
             or <#Straightness>
             or <#Surface_profile>
             or <#Symmetricity>
             or <#Total>),
        <#GeometryFeature>,
        <#hasMaterial> only
            (<#MaterialCharacteristics>
             and <#MaterialCost>
             and <#MaterialProperty>),
        <#DesignedLowerLimit>,
        <#DFM>,
        <#hasMaterial> only <#Material>
    DisjointWith:
        <#Standardization>, <#Simplicity>, <#DimensionalFlexibility>
Class: <#AID>
    SubClassOf:
        <#Concept>
Class: <#Surface_profile>
    SubClassOf:
        <#ProfileTolerance>
Class: <#Circular>
    SubClassOf:
        <#Run-outTolerance>
Class: <#Machine>
    SubClassOf:
        owl:Thing
Class: <#AvailabilityOfMachinesAndTools>
```

```
    Annotations:
        rdfs:comment "Informs the shop availability of machines and tools."^^
            xsd:string
    SubClassOf:
        <#Product>,
        <#DFMActivity>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Precautions>,
        <#Cutting_nature>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#Tool>
Class: <#Embossed>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Tool>
    SubClassOf:
        <#hasToolCost> only <#Tool>,
        <#Accessories>
Class: <#RobotMachine>
    SubClassOf:
        <#MachineType>
Class: <#JADE-CLASS>
    SubClassOf:
        owl:Class
Class: <#PocketPrismatic>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
```

```
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#DimensionalFlexibility>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#hasDimension> some
            (<#NominalDimension>
             and <#ToleranceDimension>),
        <#MillingSingleCut>,
        <#DFMActivity>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#GeometryFeature>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#hasFeature> some <#Feature>
    DisjointWith:
        <#Standardization>, <#Simplicity>, <#Machinability>
Class: <#MachineType>
    SubClassOf:
        <#Machine>
Class: <#Shoulder_depth>
    SubClassOf:
        <#Boring>
Class: <#Screwing>
    SubClassOf:
        <#Feature>,
```

```
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#Process>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#DFMPrinciples/rules>
    Annotations:
        rdfs:comment "DFM principles/rules are related to quality/possibility
            mainly."^^xsd:string
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#DFM>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>
Class: <#EDM>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
        <#Surface>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
```

```
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#TurningMachine>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#hasOperation> some <#Turning>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#Surface>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#MachineType>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Personnel>
Class: <#RotationalFeature>
    EquivalentTo:
        <#GeometryFeature>
         and (<#hasFeature> only <#RotationalFeature>)
         and (<#hasFeature> exactly 1 <#Feature>)
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Precautions>,
        <#Cutting_nature>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>
Class: <#MaterialProperty>
    SubClassOf:
        <#Material>,
        <#hasMaterialProperty> only <#MaterialProperty>
Class: <#Internal_screwing>
    SubClassOf:
        <#Screwing>
Class: <#HoleFeature>
    EquivalentTo:
```

```
        <#GeometryFeature>
         and (<#hasFeature> only <#HoleFeature>)
         and (<#hasFeature> exactly 1 <#Feature>)
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>
Class: <#Appearance>
    SubClassOf:
        <#hasMaterialProperty> only <#Appearance>,
        <#MaterialProperty>
Class: <#End_(slot_widths)>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#hasOperation> min 1 <#End_(slot_widths)>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#MachineMfgLimit>
    Annotations:
        rdfs:comment "DFM aspects related with the Material for machining:
            - Cutting nature
            - Expected finish
            - Approximate ratio"^^xsd:string,
        rdfs:comment "Machine manufacturing limits"^^xsd:string
    SubClassOf:
        <#Product>,
        <#Operation>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Machine>,
        <#Cost>,
```

```
        <#Cutting_nature>,
        <#Precautions>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>
Class: <#Boring>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Process>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Honing>
    SubClassOf:
        <#Finish>
Class: <#GeometryFeature>
    SubClassOf:
        <#Approximate_ratio>,
        <#Precautions>,
        <#isSurface> some
            (<#2Dcontoured>
             or <#3Dcontoured>
             or <#Embossed>
             or <#HollowShape>),
        <#Surface>,
        <#TypeOfFeature>,
        <#hasGeometryFeature> only <#NominalDimension>,
        <#Tool>,
        <#Product>,
        <#hasFeature> only
            (<#HoleFeature>
             or <#PrismaticFeature>
             or <#RotationalFeature>),
        <#Tolerance>,
        <#Cost>,
        <#Cutting_nature>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#hasFeature> exactly 1 <#Feature>
```

```
Class: <#ManufacturerCost>
    SubClassOf:
        <#PersonnelCost>
Class: <#Metal_extrusion>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#TypeOfFeature>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#BarcodeDevice>
    SubClassOf:
        <#Device>
Class: <#Angularity>
    SubClassOf:
        <#OrientationTolerance>
Class: <#MachineFixedCost>
    SubClassOf:
        <#MachineCost>
Class: <#Turning>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
```

```
        <#hasOperation> min 1 <#Turning>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Planning_and_shaping>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Drilling>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Powder_Metal>
```

```
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Process>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#SensorDevice>
    SubClassOf:
        <#Device>
Class: <#Feature>
    SubClassOf:
        <#Product>,
        <#hasNominalDimensionDefined> some <#Dimension>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#Tool>,
        <#Part>
Class: <#Diameter-Radius>
    SubClassOf:
        <#SizeTolerance>
Class: <#Approve>
    SubClassOf:
        <#hasRole> only <#Supervisor>,
        <#AgentAction>
Class: <#Run-outTolerance>
    SubClassOf:
        <#TypeOfTolerance>
Class: <#3Dcontoured>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#Process>,
```

```
        <#AvailabilityOfMachinesAndTools>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Circularity>
    SubClassOf:
        <#ShapeTolerance>
Class: <#ComputeCost>
    Annotations:
        rdfs:comment "Computes the cost of a part or assembly."^^xsd:string
    EquivalentTo:
        <#AgentAction>
         and <#DFMActivity>
         and (<#hasCostDefined> some
            (<#InitialCost>
             and <#ProcessingCost>))
         and (<#hasDesignerCostDefined> some <#DesignerCost>)
         and (<#hasInitialCostDefined> some <#InitialCost>)
         and (<#hasManufacturerCostDefined> some <#ManufacturerCost>)
         and (<#hasPersonnelCostDefined> some <#PersonnelCost>)
         and (<#hasProcessingCostDefined> some <#ProcessingCost>)
         and (<#hasSupervisorCostDefined> some <#SupervisorCost>)
    SubClassOf:
        <#Personnel>,
        <#AgentAction>,
        <#ProcessingCost>,
        <#InitialCost>
Class: <#Burnishing>
    SubClassOf:
        <#Finish>
Class: <#Angular>
    SubClassOf:
        <#SizeTolerance>
Class: <#PersonnelCost>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#Personnel>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#Surface>,
```

```
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#hasPersonnelCostDefined> some
            (<#DesignerCost>
             and <#ManufacturerCost>
             and <#SupervisorCost>),
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#Cost>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#MachineReaming>
    SubClassOf:
        <#Reaming>
Class: <#DFMActivity>
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Precautions>,
        <#Cutting_nature>,
        <#DFM>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#Tool>
Class: <#Device>
    SubClassOf:
        <#Accessories>
Class: <#Operation>
    SubClassOf:
        <#Product>,
        owl:Thing,
        <#Tolerance>,
        <#hasOperationType> only
            (<#Primary>
             or <#Secondary>),
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#NominalDimension>,
        <#Expected_finish>,
        <#Process>,
        <#Tool>
Class: <#MeteringDevice>
    SubClassOf:
```

```
        <#Device>
Class: <#Control>
    SubClassOf:
        <#AgentAction>,
        <#hasRole> only <#Supervisor>
Class: <#Tolerance>
    SubClassOf:
        <#hasTolerance> only <#Tolerance>
Class: <#ShapeTolerance>
    SubClassOf:
        <#TypeOfTolerance>
Class: <#CorrosionResistance>
    SubClassOf:
        <#MaterialProperty>,
        <#hasMaterialProperty> only <#CorrosionResistance>
Class: <#Expected_finish>
    SubClassOf:
        <#hasMaterialCharacteristic> only <#Expected_finish>,
        <#is_string_array> only xsd:string,
        <#MaterialCharacteristics>
Class: owl:Class
Class: <#Primary>
    SubClassOf:
        <#hasOperation> some <#Primary>,
        <#OperationType>
    DisjointWith:
        <#Secondary>
Class: <#Perpendicularity>
    SubClassOf:
        <#OrientationTolerance>
Class: <#SurfaceRoughness>
    SubClassOf:
        <#hasMaterialProperty> only <#SurfaceRoughness>,
        <#is_numeric_array> only (xsd:double or xsd:float),
        <#MaterialProperty>
Class: <#MachineCost>
    SubClassOf:
        <#Cost>
Class: <#Tolerancing>
    Annotations:
        rdfs:comment "Informs the tolerancing aspects involved in a part/
            assembly design."^^xsd:string
    EquivalentTo:
        <#AgentAction>
         and <#DFMActivity>
         and (<#hasLimitsMfgOperation> some
            ((<#DesignedLowerLimit>
             and <#MfgLowerLimit>)
             and (<#DesignedUpperLimit>
             and <#MfgUpperLimit>)))
         and (<#hasOperation> only <#MillingSingleCut>)
         and (<#hasOperation> only
            (<#End_(slot_widths)>
             or <#Face>
             or <#Slotting_(width)>
             or <#Straddle>))
         and (<#hasRole> only <#Designer>)
```

```
        and (<#hasToleranceDefined> only
           ((<#DesignedLowerLimit>
            and <#DesignedUpperLimit>)
            or (<#NominalDimension>
            and <#ToleranceDimension>)))
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#TypeOfFeature>,
        <#Process>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#AgentAction>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#FixtureCost>
    SubClassOf:
        <#AccessoriesCost>,
        <#hasFixtureCost> only <#FixtureCost>
Class: <#Approximate_ratio>
    SubClassOf:
        <#is_numeric_array> only (xsd:double or xsd:float),
        <#hasMaterialCharacteristic> only <#Approximate_ratio>,
        <#MaterialCharacteristics>
Class: <#SupervisorCost>
    SubClassOf:
        <#PersonnelCost>
Class: <#Concept>
Class: <#Compute>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#MillingSingleCut>,
        <#hasRole> some
            (<#Designer>
             or <#Manufacturer>
             or <#Supervisor>),
        <#DFMActivity>,
        <#Operation>,
```

```
        <#AgentAction>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#hasActivity> some
            (<#AvailabilityOfMachinesAndTools>
             or <#ComputeCost>
             or <#Tolerancing>
             or <#ToolAccessibility>),
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>,
        <#Personnel>,
        <#TypeOfFeature>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#GeometryFeature>,
        <#DesignedLowerLimit>,
        <#DFM>
Class: <#MaterialCost>
    SubClassOf:
        <#Cost>,
        <#hasMaterialCost> only <#Material>
Class: <#Concentricity-coaxiality>
    SubClassOf:
        <#LocationTolerance>
Class: <#MillingMachine>
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#TypeOfFeature>,
        <#Surface>,
        <#RotationalFeature>,
        <#hasOperation> some
            (<#End_(slot_widths)>
             or <#Face>
             or <#Slotting_(width)>
             or <#Straddle>),
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFM>,
        <#DesignedLowerLimit>,
```

```
        <#MachineType>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#Process>
    SubClassOf:
        <#Product>,
        owl:Thing,
        <#hasMaterial> some <#Material>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>,
        <#hasOperation> min 1 <#Operation>,
        <#hasProcess> min 1 <#Process>
Class: <#ProcessingCost>
    EquivalentTo:
        <#hasProcessingCostDefined> some
            (<#FixtureCost>
             and <#JigCost>
             and <#MachineOperationalCost>
             and <#ToolCost>)
    SubClassOf:
        <#ManufacturingFeature>,
        <#Feature>,
        <#ToleranceDimension>,
        <#hasMachineOperationalCostDefined> some <#MachineOperationalCost>,
        <#Process>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#MillingSingleCut>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#Cost>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>
Class: <#NonRoundHole>
    SubClassOf:
        <#Feature>,
        <#ManufacturingFeature>,
        <#ToleranceDimension>,
        <#Surface>,
```

```
        <#TypeOfFeature>,
        <#AvailabilityOfMachinesAndTools>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#MillingSingleCut>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#DFMActivity>,
        <#GeometryFeature>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>
Class: <#Position>
    SubClassOf:
        <#LocationTolerance>
Class: <#Part>
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Precautions>,
        <#Cutting_nature>,
        <#hasFeature> some <#Feature>,
        <#hasFeature> min 1 <#Feature>,
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>
Class: <#Stiffness>
    SubClassOf:
        <#is_float> only xsd:float,
        <#MaterialProperty>,
        <#hasMaterialProperty> only <#Stiffness>
Class: <#Grinding>
    SubClassOf:
        <#Finish>
Class: <#Product>
    SubClassOf:
        owl:Thing
Class: <#Refuse>
    SubClassOf:
        <#AgentAction>,
        <#hasRole> only <#Supervisor>
Class: <#Straightness>
    SubClassOf:
        <#ShapeTolerance>
Class: <#Base_metal>
    SubClassOf:
        <#hasMaterialIdentifier> only <#Base_metal>,
        <#MaterialIdentifier>,
```

```
        <#is_string_array> only xsd:string
Class: <#DesignedLowerLimit>
    SubClassOf:
        <#Product>,
        <#Tolerance>,
        <#Approximate_ratio>,
        <#Cost>,
        <#Cutting_nature>,
        <#Precautions>,
        <#hasLimitsMfgOperation> only
            (<#DesignedLowerLimit>
             and <#MfgLowerLimit>),
        <#Expected_finish>,
        <#NominalDimension>,
        <#Tool>,
        <#DesignedTolerance>
Class: <#ProfileTolerance>
    SubClassOf:
        <#TypeOfTolerance>
Class: <#Accessories>
    SubClassOf:
        <#Machine>,
        <#isManufacturingAccessoryOf> only <#Machine>
Class: <#MaterialCharacteristics>
    Annotations:
        rdfs:comment "Machiuning material characteristucs."^^xsd:string
    SubClassOf:
        <#Material>
Class: <#MachineOperationalCost>
    SubClassOf:
        <#MachineCost>
```

B.3 Taxonomia operacional anotada

Lista B.3 – Taxonomia operacional anotada dos componentes DFM básicos.

```
ObjectProperty: <#isDFMOf>
    InverseOf:
        <#isDFM>
ObjectProperty: <#hasTool>
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#isDFM>
    Domain:
        <#Tool>
    Range:
        <#Jig>,
        <#Tool>,
        <#Fixture>
ObjectProperty: <#hasPersonnelCostDefined>
    SubPropertyOf:
        <#hasCostDefined>
    Characteristics:
    Domain:
        <#Personnel>,
        <#Cost>
    Range:
        <#ManufacturerCost>,
        <#SupervisorCost>,
        <#DesignerCost>
ObjectProperty: <#hasMachineType>
    SubPropertyOf:
        <#hasMachine>
    Characteristics:
    Domain:
        <#MfgUpperLimit>
    Range:
        <#MillingMachine>
         or <#RobotMachine>
         or <#TurningMachine>
ObjectProperty: <#hasOperation>
    DisjointWith:
        <#hasGeometry>
    Characteristics:
    Domain:
        <#Operation>,
        <#Machining>
    Range:
        <#End_(slot_widths)>
         or <#Face>
         or <#Slotting_(width)>
         or <#Straddle>
         or <#Turning>
ObjectProperty: <#hasActivity>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFMOf>),
        <#isDFM>,
         inverse (<#isDFM>)
```

```
    Characteristics:
    Range:
        <#AvailabilityOfMachinesAndTools>
         or <#ComputeCost>
         or <#Tolerancing>
         or <#ToolAccessibility>
ObjectProperty: <#hasTolerance>
    SubPropertyOf:
        <#hasDimension>,
         inverse (<#isDFMOf>),
        <#isDFM>
    Characteristics:
    Domain:
        <#Dimension>,
        <#Tolerance>,
        <#DesignedUpperLimit>,
        <#DesignedLowerLimit>,
        <#DesignedTolerance>
ObjectProperty: <#isDimensionDefinedBy>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#hasGeometry>),
         inverse (<#hasDimension>),
         inverse (<#isDFM>)
    Characteristics:
    Domain:
        <#Dimension>,
        <#NominalDimension>
    InverseOf:
        <#hasNominalDimensionDefined>
ObjectProperty: <#isFeature>
    SubPropertyOf:
         inverse (<#hasGeometry>)
    Characteristics:
    Domain:
        <#Feature>
    Range:
        <#2Dcontoured>
         or <#3Dcontoured>
         or <#Embossed>
         or <#HollowShape>,
        <#PrismaticFeature>,
        <#HoleFeature>
         or <#PrismaticFeature>
         or <#RotationalFeature>
    InverseOf:
        <#hasFeature>
ObjectProperty: <#hasMfgProcessDefined>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFMOf>),
        <#isDFM>,
         inverse (<#isDFM>)
    Characteristics:
    Domain:
        <#Operation>
    Range:
```

```
        <#Face>,
        <#Slotting_(width)>,
        <#Straddle>,
        <#MillingSingleCut>,
        <#End_(slot_widths)>
    InverseOf:
        <#isManufacturingAccessoryOf>
ObjectProperty: <#hasSupervisorCostDefined>
    SubPropertyOf:
        <#hasPersonnelCostDefined>
    Characteristics:
    Domain:
        <#Personnel>
    Range:
        <#Supervisor>
ObjectProperty: <#isCostDefinedBy>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFM>)
ObjectProperty: <#hasMachineCost>
    SubPropertyOf:
        <#hasCost>
    DisjointWith:
        <#hasFixtureCost>
    Characteristics:
    Domain:
        <#hasMachine> only <#Machine>
ObjectProperty: <#hasOperationType>
    SubPropertyOf:
        <#hasOperation>
    Characteristics:
    Domain:
        <#OperationType>
    Range:
        <#Primary>
         or <#Secondary>
ObjectProperty: <#isLimit>
ObjectProperty: <#hasPersonnelCost>
    SubPropertyOf:
        <#hasCost>
    Characteristics:
ObjectProperty: <#isSurface>
    SubPropertyOf:
        <#isFeature>,
         inverse (<#hasFeature>)
    Characteristics:
    Domain:
        <#Feature>
    Range:
        <#2Dcontoured>
         or <#3Dcontoured>
         or <#Embossed>
         or <#HollowShape>
ObjectProperty: <#isNeededComponentOf>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFMOf>),
```

```
        <#isDFM>,
         inverse (<#isDFM>)
    Characteristics:
    Domain:
        <#DFM>
    InverseOf:
        <#needsComponent>
ObjectProperty: <#hasGeometryFeature>
    SubPropertyOf:
         inverse (<#isFeature>),
        <#hasFeature>
    Characteristics:
    Domain:
        <#GeometryFeature>
    Range:
        <#HoleFeature>
         or <#PrismaticFeature>
         or <#RotationalFeature>
ObjectProperty: <#hasMachine>
    DisjointWith:
        <#hasGeometry>,
        <#hasMaterial>
    Characteristics:
    Domain:
        <#Operation>,
        <#Machine>,
        <#AvailabilityOfMachinesAndTools>
    Range:
        <#Face>,
        <#Machine>,
        <#Slotting_(width)>,
        <#Straddle>,
        <#MillingSingleCut>,
        <#End_(slot_widths)>
ObjectProperty: <#needsComponent>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFMOf>),
        <#isDFM>,
         inverse (<#isDFM>)
    Characteristics:
    Domain:
        <#DFM>
    Range:
        <#Tolerancing>,
        <#Machine>,
        <#Approve>,
        <#Compute>,
        <#AvailabilityOfMachinesAndTools>,
        <#ProcessingCost>,
        <#ComputeCost>,
        <#Tool>,
        <#InitialCost>,
        <#Fixture>,
        <#Refuse>,
        <#Designer>,
        <#Cost>,
```

```
        <#Supervisor>,
        <#Manufacturer>,
        <#Jig>,
        <#ToolAccessibility>
    InverseOf:
        <#isNeededComponentOf>
ObjectProperty: <#hasSurface>
    SubPropertyOf:
        <#hasGeometryFeature>
    Characteristics:
    Domain:
        <#hasSurface> only <#Surface>
    Range:
        <#hasSurface> only
            (<#2Dcontoured>
             or <#3Dcontoured>
             or <#Embossed>
             or <#HollowShape>)
ObjectProperty: <#hasToolCost>
    SubPropertyOf:
        <#hasCost>
    DisjointWith:
        <#hasFixtureCost>,
        <#hasJigCost>
    Characteristics:
    Range:
        <#ToolCost>
ObjectProperty: <#hasToolCostDefined>
    SubPropertyOf:
        <#hasProcessingCostDefined>
    Characteristics:
    Domain:
        <#Tool>
    Range:
        <#ToolCost>
ObjectProperty: <#hasJigCost>
    SubPropertyOf:
        <#hasCost>
    DisjointWith:
        <#hasToolCost>,
        <#hasFixtureCost>
    Characteristics:
    Range:
        <#JigCost>
ObjectProperty: <#hasInitialCostDefined>
    SubPropertyOf:
        <#hasCostDefined>
    Characteristics:
    Domain:
        <#Cost>,
        <#InitialCost>
    Range:
        <#Machine>
ObjectProperty: <#hasProcessingCostDefined>
    SubPropertyOf:
        <#hasCostDefined>
    Characteristics:
```

```
    Domain:
        <#Cost>,
        <#ProcessingCost>
    Range:
        <#Jig>,
        <#Tool>,
        <#Fixture>
ObjectProperty: <#hasMaterial>
    DisjointWith:
        <#hasMachine>,
        <#hasGeometry>
    Characteristics:
    Domain:
        <#Material>,
        <#Process>
    Range:
        <#Material>
ObjectProperty: <#hasRole>
    SubPropertyOf:
        <#hasPersonnel>
    Characteristics:
    Domain:
        <#Personnel>
    Range:
        <#Designer>
         or <#Manufacturer>
         or <#Supervisor>
ObjectProperty: <#hasProductivity>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFMOf>),
        <#isDFM>,
         inverse (<#isDFM>)
    Characteristics:
    InverseOf:
        <#isProductivityDefinedBy>
ObjectProperty: <#hasDimension>
    Characteristics:
    Domain:
        <#Dimension>
    Range:
        <#GeometryFeature>,
        <#Tolerance>,
        <#NominalDimension>
ObjectProperty: <#hasNominalDimensionDefined>
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#hasDimension>,
        <#hasGeometry>,
        <#isDFM>
    Characteristics:
    Domain:
        <#Dimension>
    Range:
        <#hasNominalDimensionDefined> only <#NominalDimension>
    InverseOf:
        <#isDimensionDefinedBy>
```

```
ObjectProperty: <#hasTypeOfToleranceDefined>
    SubPropertyOf:
        <#hasToleranceDefined>
    Characteristics:
    Domain:
        <#ToleranceDimension>
    Range:
        <#TypeOfTolerance>,
        <#Angular>
         or <#Angularity>
         or <#Circular>
         or <#Circularity>
         or <#Concentricity-coaxiality>
         or <#Cylindricity>
         or <#Diameter-Radius>
         or <#Flatness>
         or <#Line_profile>
         or <#Linear>
         or <#LocationTolerance>
         or <#OrientationTolerance>
         or <#Parallelism>
         or <#Perpendicularity>
         or <#Position>
         or <#ProfileTolerance>
         or <#Run-outTolerance>
         or <#ShapeTolerance>
         or <#SizeTolerance>
         or <#Straightness>
         or <#Surface_profile>
         or <#Symmetricity>
         or <#Total>
ObjectProperty: <#hasManufacturingFeature>
    SubPropertyOf:
         inverse (<#isFeature>),
        <#hasFeature>
    Characteristics:
    Domain:
        <#ManufacturingFeature>
    Range:
        (<#HoleFeature>
         or <#PrismaticFeature>
         or <#RotationalFeature>)
         and (<#hasGeometryFeature> some <#Surface>)
ObjectProperty: <#hasLimitsOfMfgSize>
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#isDFM>
    Domain:
        <#Operation>
ObjectProperty: <#hasFixtureCost>
    SubPropertyOf:
        <#hasCost>
    DisjointWith:
        <#hasMachineCost>,
        <#hasToolCost>,
        <#hasMaterialCost>,
        <#hasJigCost>
```

```
    Characteristics:
    Range:
        <#FixtureCost>
ObjectProperty: <#hasProperty>
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#isDFM>
    Domain:
        <#Material>
    Range:
        <#Formability>,
        <#CorrosionResistance>,
        <#Strenght>,
        <#ElectricalConductivity>,
        <#Stiffness>,
        <#SurfaceRoughness>,
        <#Appearance>
ObjectProperty: <#hasMaterialProperty>
    SubPropertyOf:
        <#hasMaterial>
    Characteristics:
    Range:
        <#hasMaterialProperty> some <#Strenght>,
        <#hasMaterialProperty> some <#Appearance>,
        <#hasMaterialProperty> some <#Formability>,
        <#hasMaterialProperty> some <#CorrosionResistance>,
        <#hasMaterialProperty> some <#Stiffness>
ObjectProperty: <#isToolAccessibilityFrom>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#hasTool>)
    Characteristics:
    Domain:
        <#Feature>
    Range:
        <#ManufacturingFeature>
    InverseOf:
        <#hasToolAccessibility>
ObjectProperty: <#hasProcessPrecautions>
    SubPropertyOf:
        <#hasProcess>
    Characteristics:
    Domain:
        <#MillingSingleCut>
         and <#Operation>
         and <#Process>
    Range:
        <#End_(slot_widths)>
         or <#Face>
         or <#Slotting_(width)>
         or <#Straddle>
ObjectProperty: <#hasAvailabityOfMachinesTools>
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#isDFM>
    Characteristics:
    Domain:
```

```
        <#Machine>,
        <#AvailabilityOfMachinesAndTools>,
        <#Accessories>
    Range:
        <#Jig>,
        <#Tool>,
        <#Machine>
         and <#Tool>,
        <#Fixture>
ObjectProperty: <#hasLimitsMfgOperation>
    SubPropertyOf:
        <#hasLimitsOfMfgSize>
    Characteristics:
    Domain:
        <#Dimension>,
        <#MillingSingleCut>,
        <#MachineMfgLimit>
    Range:
        <#MfgLowerLimit>,
        <#Face>,
        <#DesignedUpperLimit>,
        <#MfgUpperLimit>,
        <#DesignedLowerLimit>,
        <#Slotting_(width)>,
        <#Straddle>,
        <#End_(slot_widths)>
ObjectProperty: <#isProductivityDefinedBy>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFMOf>),
        <#isDFM>,
         inverse (<#isDFM>)
    Characteristics:
    InverseOf:
        <#hasProductivity>
ObjectProperty: <#hasCostDefined>
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#isDFM>,
        <#hasCost>
    Characteristics:
    Domain:
        <#Cost>
    Range:
        <#InitialCost>
         or <#ProcessingCost>
ObjectProperty: <#hasMachineFixedCostDefined>
    SubPropertyOf:
        <#hasInitialCostDefined>
    Characteristics:
    Domain:
        <#Machine>
    Range:
        <#MachineFixedCost>
ObjectProperty: <#hasPrinciplesRules>
    Annotations:
```

```
        rdfs:comment "DimensionalFlexibility or Machinability sounds
            strange but it is what happens in practice."^^xsd:string
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#isDFM>
    Characteristics:
        Asymmetric,
        Irreflexive,
        Symmetric,
        Transitive
    Domain:
        <#DFMPrinciples/rules>
    Range:
        <#Standardization>,
        <#Simplicity>,
        <#DimensionalFlexibility>
         or <#Machinability>
ObjectProperty: <#hasFixtureCostDefined>
    SubPropertyOf:
        <#hasProcessingCostDefined>
    Characteristics:
    Domain:
        <#Fixture>
    Range:
        <#FixtureCost>
ObjectProperty: <#hasJigCostDefined>
    SubPropertyOf:
        <#hasProcessingCostDefined>
    Characteristics:
    Domain:
        <#Jig>
    Range:
        <#JigCost>
ObjectProperty: <#hasMaterialIdentifier>
    SubPropertyOf:
        <#hasMaterial>
    Characteristics:
    Range:
        <#Alloy>,
        <#Base_metal>,
        <#UNS>
ObjectProperty: <#hasToolAccessibility>
    SubPropertyOf:
        <#hasTool>,
        <#isDFM>
    Characteristics:
    Domain:
        <#Feature>
    Range:
        <#ManufacturingFeature>,
        <#GeometryFeature>
    InverseOf:
        <#isToolAccessibilityFrom>
ObjectProperty: <#isManufacturingAccessoryOf>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFMOf>),
```

```
        <#isDFM>,
         inverse (<#isDFM>)
    Characteristics:
    InverseOf:
        <#hasMfgProcessDefined>
ObjectProperty: <#hasPersonnel>
    DisjointWith:
        <#hasPart>
    Domain:
        <#Personnel>
    Range:
        <#Designer>
         or <#Manufacturer>
         or <#Supervisor>
ObjectProperty: <#hasProcess>
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#isDFM>
    Characteristics:
    Domain:
        <#hasProcess> some <#Broaching>,
        <#hasProcess> some <#Reaming>,
        <#hasProcess> some <#Center-type_and_center-less>,
        <#hasProcess> some <#MachineReaming>,
        <#hasProcess> some <#Burnishing>,
        <#Process>,
        <#hasProcess> some <#Cylindrical>,
        <#hasProcess> some <#Hobbing>,
        <#MillingSingleCut>,
        <#hasProcess> some <#ExternalCylindrical>,
        <#hasProcess> some <#HandReaming>,
        <#Operation>,
        <#hasProcess> some <#Finish>,
        <#hasProcess> some <#Rough>,
        <#hasProcess> some <#InternalCylindrical>,
        <#hasProcess> some <#MillingSingleCut>,
        <#hasProcess> some <#Planning_and_shaping>,
        <#hasProcess> some <#Honing>,
        <#hasProcess> some <#Grinding>
    Range:
        <#End_(slot_widths)>
         or <#Face>
         or <#Slotting_(width)>
         or <#Straddle>
ObjectProperty: <#hasCost>
    DisjointWith:
        <#hasGeometry>
    Characteristics:
    Domain:
        <#Cost>
    Range:
        <#ManufacturerCost>,
        <#SupervisorCost>,
        <#MachineFixedCost>,
        <#DesignerCost>,
        <#MachineOperationalCost>
ObjectProperty: <#hasMaterialCharacteristic>
```

```
    SubPropertyOf:
        <#hasMaterial>
    Characteristics:
    Domain:
        <#Approximate_ratio>,
        <#Cutting_nature>,
        <#Precautions>,
        <#Expected_finish>
    Range:
        <#hasMaterialCharacteristic> some <#Expected_finish>,
        <#hasMaterialCharacteristic> some <#Approximate_ratio>,
        <#hasMaterialCharacteristic> some <#Cutting_nature>,
        <#hasMaterialCharacteristic> some <#Precautions>
ObjectProperty: <#hasAccessoryCost>
    SubPropertyOf:
        <#hasCost>
    Characteristics:
    Domain:
        <#JigCost>,
        <#FixtureCost>,
        <#ToolCost>
    Range:
        <#JigCost>,
        <#FixtureCost>,
        <#ToolCost>
ObjectProperty: <#hasToleranceDefined>
    SubPropertyOf:
        <#hasTolerance>
    Characteristics:
    Domain:
        <#Dimension>
    Range:
        <#DesignedLowerLimit>
         and <#DesignedUpperLimit>,
        <#NominalDimension>
         and <#ToleranceDimension>
ObjectProperty: <#hasMachineOperationalCostDefined>
    SubPropertyOf:
        <#hasProcessingCostDefined>
    Characteristics:
    Domain:
        <#Machine>
    Range:
        <#MachineOperationalCost>
ObjectProperty: owl:topObjectProperty
    Characteristics:
        Symmetric,
        Reflexive,
        Transitive
    InverseOf:
        owl:topObjectProperty
ObjectProperty: <#hasDesignerCostDefined>
    SubPropertyOf:
        <#hasPersonnelCostDefined>
    Characteristics:
    Domain:
        <#Personnel>
```

```
    Range:
        <#Designer>
ObjectProperty: <#isPart>
    Characteristics:
    Domain:
        <#Part>
    Range:
        <#HoleFeature>
         or <#PrismaticFeature>
         or <#RotationalFeature>
ObjectProperty: <#hasMachiningCharacteristics>
    SubPropertyOf:
        <#hasProcess>
    Characteristics:
    Domain:
        <#hasProcess> only <#Machining>
ObjectProperty: <#hasMaterialCost>
    SubPropertyOf:
        <#hasCost>
    DisjointWith:
        <#hasFixtureCost>
    Characteristics:
    Domain:
        <#hasMaterialCost> some <#Material>
ObjectProperty: <#isTolerancingFrom>
    SubPropertyOf:
        <#isDFMOf>,
         inverse (<#isDFM>)
ObjectProperty: <#isAssembly>
    SubPropertyOf:
        owl:topObjectProperty
    Domain:
        <#Product>
    Range:
        <#isAssembly> only <#Part>,
        <#isAssembly> min 1 <#Part>
    InverseOf:
        <#hasAssembly>
ObjectProperty: <#isDFM>
    Domain:
        <#DFM>
    Range:
        <#hasProcess> some <#GeometryFeature>,
        <#hasGeometryFeature> some <#PrismaticFeature>,
        <#hasFeature> some <#PrismaticFeature>,
        <#hasManufacturingFeature> some <#HoleFeature>,
        <#hasProcess> some <#RotationalFeature>,
        <#hasGeometry> some <#ManufacturingFeature>,
        <#hasProcess> some <#HoleFeature>,
        <#hasFeature> some <#HoleFeature>,
        <#DFMActivity>,
        <#hasGeometryFeature> some <#HoleFeature>,
        <#hasGeometry> some <#PrismaticFeature>,
        <#hasGeometryFeature> some <#GeometryFeature>,
        <#DFMPrinciples/rules>,
        <#hasManufacturingFeature> some <#PrismaticFeature>,
        <#hasFeature> some <#RotationalFeature>,
```

```
        <#hasManufacturingFeature> some <#ManufacturingFeature>,
        <#hasFeature> some <#ManufacturingFeature>,
        <#hasManufacturingFeature> some <#GeometryFeature>,
        <#hasProcess> some <#Process>,
        <#hasGeometryFeature> some <#ManufacturingFeature>,
        <#hasGeometry> some <#HoleFeature>,
        <#hasGeometry> some <#RotationalFeature>,
        <#hasProcess> some <#PrismaticFeature>,
        <#hasProcess> some <#ManufacturingFeature>,
        <#hasGeometry> some <#GeometryFeature>,
        <#hasManufacturingFeature> some <#RotationalFeature>,
        <#hasFeature> some <#GeometryFeature>,
        <#hasGeometryFeature> some <#RotationalFeature>
    InverseOf:
        <#isDFMOf>
ObjectProperty: <#hasFeature>
    SubPropertyOf:
        <#hasGeometry>
    Characteristics:
    Domain:
        <#Feature>,
        <#Part>
    Range:
        <#hasFeature> only
            (<#Feature>
             and <#Part>)
    InverseOf:
        <#isFeature>
ObjectProperty: <#hasAssembly>
    Domain:
        <#Assembly>
    InverseOf:
        <#isAssembly>
ObjectProperty: <#hasGeometry>
    DisjointWith:
        <#hasMachine>,
        <#hasOperation>,
        <#hasCost>,
        <#hasMaterial>,
        <#hasPart>
ObjectProperty: <#hasMachineAccessory>
    SubPropertyOf:
        <#hasMachine>
    Characteristics:
    Domain:
        <#Machine>
    Range:
        <#Tool>
         and (<#Fixture>
         or <#Jig>)
ObjectProperty: <#isOperation>
    SubPropertyOf:
         inverse (<#isDFMOf>),
        <#isDFM>
    Domain:
        <#Operation>
    Range:
```

```
        <#Face>,
        <#Blind>,
        <#Through>,
        <#Slotting_(width)>,
        <#End>,
        <#Hollow>,
        <#Straddle>,
        <#MillingSingleCut>,
        <#End_(slot_widths)>
ObjectProperty: <#hasPart>
    Annotations:
        rdfs:comment "A related resource that is included either
            physically or logically in the described resource."^^xsd:
            string,
        rdfs:comment "It describes widely what is involved in DFM."^^xsd:
            string
    DisjointWith:
        <#hasGeometry>,
        <#hasPersonnel>
    Characteristics:
    Domain:
        <#Product>
    Range:
        <#Assembly>,
        <#Part>
ObjectProperty: <#hasManufacturerCostDefined>
    SubPropertyOf:
        <#hasPersonnelCostDefined>
    Characteristics:
    Domain:
        <#Personnel>
    Range:
        <#Manufacturer>
```

B.4 Taxonomia anotada de indivíduos

Lista B.4 – Taxonomia anotada de indivíduos dos componentes DFM básicos.

```
Individual: <#millimeter>
    Types:
        <#Feature>,
        <#ManufacturingFeature>,
        <#Machine>,
        <#Precautions>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Tool>,
        <#MillingSingleCut>,
        <#Material>,
        <#DFMActivity>,
        <#Tolerance>,
        <#AgentAction>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#Cost>,
        <#PrismaticFeature>,
        <#DimensionalFlexibility>,
        <#Machining>,
        <#Expected_finish>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>,
        <#Tolerancing>,
        <#Dimension>,
        <#Approximate_ratio>,
        <#Concept>,
        <#TypeOfFeature>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#Product>,
        <#GeometryFeature>,
        <#Cutting_nature>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#NominalDimension>,
        <#Accessories>,
        <#MaterialCharacteristics>
    Facts:
     <#hasDimension>  <#millimeter>,
     <#isDimensionDefinedBy>  <#millimeter>,
     <#isTolerancingFrom>  <#millimeter>,
     <#hasTolerance>  <#millimeter>,
     <#isDFMOf>  <#millimeter>,
     <#hasNominalDimensionDefined>  <#millimeter>,
     <#hasToleranceDefined>  <#millimeter>,
     <#hasGeometry>  <#millimeter>,
     <#isDFM>  <#millimeter>
```

```
Individual: <#Real>
    Types:
        <#Feature>,
        <#ManufacturingFeature>,
        <#Machine>,
        <#Precautions>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Tool>,
        <#ToolCost>,
        <#MillingSingleCut>,
        <#InitialCost>,
        <#Material>,
        <#DFMActivity>,
        <#Tolerance>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Cost>,
        <#Rough>,
        <#PrismaticFeature>,
        <#Machining>,
        <#Expected_finish>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>,
        <#JigCost>,
        <#Dimension>,
        <#FixtureCost>,
        <#Approximate_ratio>,
        <#MaterialCost>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#Product>,
        <#GeometryFeature>,
        <#AccessoriesCost>,
        <#Cutting_nature>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#NominalDimension>,
        <#Accessories>,
        <#MaterialCharacteristics>
Individual: <#DFMindex>
    Types:
        <#Feature>,
        <#ManufacturingFeature>,
        <#Machine>,
        <#Precautions>,
        <#ToleranceDimension>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Tool>,
        <#MillingSingleCut>,
        <#Material>,
```

```
        <#DFMActivity>,
        <#Operation>,
        <#Tolerance>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#Cost>,
        <#PrismaticFeature>,
        <#Machining>,
        <#Expected_finish>,
        <#DFMPrinciples/rules>,
        <http://www.nist.gov/OntoSTEP/config_control_design#to_string>
            some xsd:integer,
        <#DesignedTolerance>,
        <#Dimension>,
        <#Approximate_ratio>,
        <http://www.nist.gov/OntoSTEP/config_control_design#to_string>
            some xsd:string,
        <#TypeOfFeature>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#Product>,
        <#GeometryFeature>,
        <#Cutting_nature>,
        <http://www.nist.gov/OntoSTEP/config_control_design#to_decimal>
            some xsd:string,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#NominalDimension>,
        <#Accessories>,
        <#MaterialCharacteristics>
Individual: <#Rotational>
    Types:
        <#Feature>,
        <#ManufacturingFeature>,
        <#Machine>,
        <#ToleranceDimension>,
        <#Precautions>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Tool>,
        <#MillingSingleCut>,
        <#Material>,
        <#DFMActivity>,
        <#Operation>,
        <#Tolerance>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#Cost>,
        <#PrismaticFeature>,
        <#Machining>,
        <#Expected_finish>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>,
        <#Dimension>,
```

```
        <#Approximate_ratio>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#Product>,
        <#GeometryFeature>,
        <#Cutting_nature>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#NominalDimension>,
        <#Accessories>,
        <#MaterialCharacteristics>
Individual: <#Predicate>
    Types:
        <#JADE-CLASS>,
        owl:Class
Individual: <#AgentAction>
    Types:
        <#ManufacturingFeature>,
        <#Machine>,
        <#Precautions>,
        <#AvailabilityOfMachinesAndTools>,
        <#Tool>,
        <#InitialCost>,
        <#JADE-CLASS>,
        <#AgentAction>,
        <#Cost>,
        <#PrismaticFeature>,
        <#Machining>,
        <#DFMPrinciples/rules>,
        <#Dimension>,
        <#Personnel>,
        <#RotationalFeature>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#GeometryFeature>,
        <#Designer>,
        <#NominalDimension>,
        <#ToolAccessibility>,
        <#Feature>,
        <#ToleranceDimension>,
        <#Surface>,
        <#ComputeCost>,
        <#MillingSingleCut>,
        <#DFMActivity>,
        <#Material>,
        <#Tolerance>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#Expected_finish>,
        owl:Class,
        <#DesignedTolerance>,
        <#Tolerancing>,
```

```
        <#Approximate_ratio>,
        <#Concept>,
        <#TypeOfFeature>,
        <#Process>,
        <#ProcessingCost>,
        <#Part>,
        <#Product>,
        <#Cutting_nature>,
        <#DFM>,
        <#Supervisor>,
        <#DesignedLowerLimit>,
        <#Accessories>,
        <#AID>,
        <#MaterialCharacteristics>
Individual: <#Prismatic>
    Types:
        <#ManufacturingFeature>,
        <#Feature>,
        <#Machine>,
        <#Precautions>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Tool>,
        <#MillingSingleCut>,
        <#Material>,
        <#DFMActivity>,
        <#Tolerance>,
        <#Operation>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#Cost>,
        <#PrismaticFeature>,
        <#Machining>,
        <#Expected_finish>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>,
        <#Dimension>,
        <#Approximate_ratio>,
        <#TypeOfFeature>,
        <#Process>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#Product>,
        <#GeometryFeature>,
        <#Cutting_nature>,
        <#DesignedLowerLimit>,
        <#DFM>,
        <#NominalDimension>,
        <#Accessories>,
        <#MaterialCharacteristics>
Individual: <#Axis>
    Types:
        <#ManufacturingFeature>,
        <#Feature>,
```

```
        <#Machine>,
        <#ToleranceDimension>,
        <#Precautions>,
        <#Surface>,
        <#AvailabilityOfMachinesAndTools>,
        <#Tool>,
        <#MillingSingleCut>,
        <#Material>,
        <#DFMActivity>,
        <#Operation>,
        <#AgentAction>,
        <#Tolerance>,
        <#DesignedUpperLimit>,
        <#Rough>,
        <#Cost>,
        <#PrismaticFeature>,
        <#Machining>,
        <#Expected_finish>,
        <#DesignedTolerance>,
        <#DFMPrinciples/rules>,
        <#Dimension>,
        <#Personnel>,
        <#Approximate_ratio>,
        <#Concept>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#Product>,
        <#GeometryFeature>,
        <#Cutting_nature>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#NominalDimension>,
        <#ToolAccessibility>,
        <#Accessories>,
        <#MaterialCharacteristics>
Individual: <#Hole>
    Types:
        <#Feature>,
        <#ManufacturingFeature>,
        <#Machine>,
        <#Precautions>,
        <#ToleranceDimension>,
        <#AvailabilityOfMachinesAndTools>,
        <#Surface>,
        <#Tool>,
        <#MillingSingleCut>,
        <#DFMActivity>,
        <#Material>,
        <#Operation>,
        <#Tolerance>,
        <#DesignedUpperLimit>,
        <#Cost>,
        <#Rough>,
```

```
        <#PrismaticFeature>,
        <#Machining>,
        <#Expected_finish>,
        <#DFMPrinciples/rules>,
        <#DesignedTolerance>,
        <#Dimension>,
        <#Approximate_ratio>,
        <#Process>,
        <#TypeOfFeature>,
        <#RotationalFeature>,
        <#Part>,
        <#HoleFeature>,
        <#MachineMfgLimit>,
        <#Product>,
        <#GeometryFeature>,
        <#Cutting_nature>,
        <#DFM>,
        <#DesignedLowerLimit>,
        <#NominalDimension>,
        <#Accessories>,
        <#MaterialCharacteristics>
Individual: <#Concept>
    Types:
        <#JADE-CLASS>,
        owl:Class
Individual: <#AID>
    Types:
        <#JADE-CLASS>,
        owl:Class
Individual: <#$>
    Types:
        <#JigCost>,
        <#Material>,
        <#FixtureCost>,
        <#AccessoriesCost>,
        <#Cost>,
        <#MaterialCost>,
        <#ToolCost>
```

B.5 Taxonomia anotada de dados

Lista B.5 – Taxonomia anotada dos dados dos componentes DFM básicos.

```
Datatype: xsd:decimal
Datatype: xsd:short
Datatype: rdf:PlainLiteral
Datatype: xsd:boolean
Datatype: xsd:int
Datatype: xsd:long
Datatype: xsd:string
Datatype: xsd:float
Datatype: xsd:double
Datatype: xsd:integer

DataProperty: <#is_tolerance_limit>
    Characteristics:
    Domain:
        (<#is_tolerance_limit> some xsd:double)
         or (<#is_tolerance_limit> some xsd:float)
    Range:
        (xsd:double or xsd:float)
DataProperty: <#is_numeric_array>
    Annotations:
        rdfs:comment "Precautions and Approximate_ratio qualifiers."^^xsd
            :string
    Characteristics:
    Domain:
        <#Approximate_ratio>
         or <#Precautions>,
        <#is_numeric_array> only (xsd:double or xsd:float)
    Range:
        xsd:float,
        xsd:double

    SubPropertyOf:
        <#is_array>
DataProperty: <#is_tol_mfg_interval>
    Characteristics:
    Domain:
        <#ToleranceDimension>,
        <#DesignedTolerance>
    Range:
        xsd:float
    SubPropertyOf:
        <#is_float>
DataProperty: <#to_crisp>
    Characteristics:
    Range:
        xsd:decimal,
        xsd:short,
        xsd:int,
        xsd:long,
        xsd:double,
        xsd:float,
        xsd:integer
DataProperty: <#is_array>
```

```
DataProperty: <#JADE-JAVA-BASE-CLASS>
    Characteristics:
    Domain:
        <#JADE-CLASS>
    Range:
        xsd:string
DataProperty: <#is_string_array>
    Annotations:
        rdfs:comment "Cutting_nature and Expected_finish qualifiers."^^
            xsd:string
    Characteristics:
    Domain:
        <#is_string_array> only xsd:string,
        <#Cutting_nature>
         or <#Expected_finish>
    Range:
        xsd:string
    SubPropertyOf:
        <#is_string>,
        <#is_array>
DataProperty: <#to_define_limit>
    Characteristics:
    Range:
        (xsd:double or xsd:float)
DataProperty: <#JADE-UNNAMED-SLOT>
    Characteristics:
    Domain:
        <#JADE-SLOT>
    Range:
        xsd:boolean
DataProperty: <http://www.nist.gov/OntoSTEP/config_control_design#
    to_decimal>
DataProperty: <http://www.nist.gov/OntoSTEP/config_control_design#
    to_integer>
    Characteristics:
    Range:
        xsd:integer
DataProperty: <#is_precautions>
    Domain:
        <#Precautions>
DataProperty: <#JADE-IGNORE>
    Characteristics:
    Domain:
        <#JADE-CLASS>
    Range:
        xsd:boolean
DataProperty: <#JADE-JAVA-CODE>
    Characteristics:
    Domain:
        <#JADE-CLASS>
    Range:
        xsd:string
DataProperty: <http://www.nist.gov/OntoSTEP/config_control_design#
    to_string>
DataProperty: <#is_string>
    Annotations:
        rdfs:comment "Regarding materials:
```

```
        The string can be either a base metal, alloy or UNS."^^xsd:string
    Characteristics:
    Domain:
        <#MaterialIdentifier>
    Range:
        xsd:string
DataProperty: <#string>
    Characteristics:
    Domain:
        <#is_string> only xsd:string
DataProperty: <http://www.nist.gov/OntoSTEP/config_control_design#
    to_boolean>
    Characteristics:
    Range:
        xsd:boolean
DataProperty: <#JADE-NAME>
    Characteristics:
    Domain:
        <#JADE-CLASS>
         or <#JADE-SLOT>
    Range:
        xsd:string
DataProperty: <#to_fuzzy>
    Characteristics:
    Range:
        xsd:string
DataProperty: <#is_float>
    Characteristics:
    Range:
        xsd:float
```

ANEXO A – LÓGICA MODAL E TAXONOMIA

Este apêndice descreve uma breve introdução a um conceito relevante para a compreensão da utilização do contexto da informação: Lógica Modal [Platzer, 2010]. Adicionalmente, a lógica modal serve como fundação para a sintaxe do conceito de ontologias utilizado nesta pesquisa através de Lógica Descritiva, ou *Description Logics – DL*. A lógica modal permite expandir o fundamento básico de lógica clássica através da utilização de interpretações sobre conceitos. O conceito básico de lógica clássica define que a partir de raciocínio baseado em fundamentos válidos sempre ocasionarão conclusões válidas. Se este fundamento é incorreto existe algum tipo de equívoco, ou erro, na estrutura de raciocínio.

De forma oposta, em lógica modal diferentes contextos podem ser analogamente representados através de "mundos possíveis". Consequentemente, diferentes interpretações sobre um determinado conceito podem existir. Dekker postula, através de uma similaridade análoga com esta pesquisa, que a forma mais simples e adequada para compreender lógica modal e seu posicionamento neste trabalho é através de um exemplo demonstrando seu contexto de funcionamento, descrito a seguir [Dekker, 2004].

A.1 Aspectos Iniciais

A lógica clássica tem por objeto de estudo a validade dos argumentos sob um ponto de vista formal, portanto sem interpretação. Um argumento é composto de premissas e conclusão. Sua validade consiste que sua estrutura garanta que, supondo que todas as premissas são verdadeiras, sua conclusão também será verdadeira. Desta forma, transfere o valor–de–verdade das premissas para a conclusão baseado apenas na virtude da sua forma ao invés de seu conteúdo. As premissas e conclusões são proposições – sentenças bipolares bem formadas, ou seja, elas possuem apenas dois polos (valores): V ou F *e somente* dois polos (lei do terceiro excluído). Eles têm somente um dos dois valores: V ou F. Neste contexto, uma premissa (seu fundamento ou *inputs* do argumento) ou uma conclusão (*output*) não são válidos se existe algum erro. O conceito de validade é técnico, ou seja, só se aplica aos argumentos porém não à proposição. As lógicas não clássicas, tais como a lógica modal, não são bipolares pois admitem outros valores–de–verdade.

A Lógica Modal permite utilizar extensões para uma determinada afirmação, por exemplo o conceito o "Custo é alto" (custo=*C*). Esta é uma afirmação verdadeira, ou SIM. Desta forma, existem diferentes formas de definir este conceito (ou *modalidade*) tais como,

- "Custo é supostamente alto".

- "Custo é admitido como alto".
- "Custo deve ser alto".
- "Custo é eventualmente alto".
- "Custo é necessariamente alto".

As formas distintas de definir um conceito são relacionadas com o conceito real de um determinado contexto formando *mundos possíveis*. Um mundo possível é aquele cujas características são diferentes de um padrão aceito em grau maior ou menor. O aspecto relacionado com a validade, ou existência, de mundos possíveis é uma questão metafísica profunda fortemente relacionada com teologia e física. Entretanto, neste trabalho é assumido que qualquer coisa que possa ser imaginada sem contradições é um mundo possível.

A.2 Opinião

A forma mais simples de introduzir o conceito de mundos possíveis é através da modalidade, ou opinião. Logo, é possível imaginar uma determinada pessoa particular: Roberto, por exemplo). A especificação que Roberto acredita que o custo é alto infere que em todos mundos possíveis onde Roberto possa existir o custo é alto.

A lógica clássica impõe de forma bastante restrita aspectos sobre a forma de pensar de Roberto pois ela está baseada em dois axiomas básicos sobre uma opinião. O primeiro axioma é definido por,

Em qualquer mundo possível w Roberto acredita em T para todas tautologias T.[1] (A.1)

As tautologias incluem todas leis da lógica, tais como[2],

- P e Q implica P ou Q (ou seja, P e $Q \Rightarrow P$ ou Q).
- $((P \Rightarrow R)$ e $(Q \Rightarrow R)$ e $(P$ ou $Q)) \Rightarrow R$.
- P ou não $-P$.

O segundo axioma, ou regra de inferência, define que Roberto aceita todas as consequências lógicas de sua opinião é definido por,

[1] Uma tautologia é linguisticamente equivalente a um pleonasmo.

[2] $\Rightarrow$ é tradicionalmente conhecido como implicação material, ou implicação, embora também seja utilizado como consequência lógica.

Em qualquer mundo w se Roberto acredita no custo C e em $C \Rightarrow Z$ então Roberto acredita em Z. (A.2)

Estes axiomas não são realísticos pois Roberto é definido como um quase–Deus. Entretanto, é necessário considerar o fato que não é possível, ou muito difícil, definir limites formais rígidos com relação à habilidade mental de Roberto. Consequentemente, assumir as hipóteses A.1 e A.2 é uma maneira de definir a racionalidade de Roberto. Adicionalmente, é relevante descrever que não existem obrigações sobre a verdade ou o quão completa é uma determinada opinião em um mundo possível.

A.3 Semântica Kriptke

Kriptke [Cocchiarella e Freund, 2008] definiu uma forma de relacionar mundos possíveis através de setas, por exemplo $w_1 \rightarrow w_2$; esta relação denomina–se *acessibilidade*. Utilizando a opinião de Roberto é válido afirmar que se Roberto vive em um mundo possível w_1 então w_2 consiste em um mundo possível segundo ele. Assumindo que Roberto seja de um mundo possível v e $v \rightarrow w_1$, $v \rightarrow w_2$, etc. (ou seja, w_{is} são mundos possíveis) é possível especificar que "Roberto acredita em C" é verdadeiro no mundo v exatamente quando C é verdadeiro em todos mundos possíveis w_i[3],

Em qualquer mundo v Roberto acredita no custo alto $C \Leftrightarrow C$ é verdadeiro em todos w_i com $v \rightarrow w_i$ (A.3)

Analogamente é possível concluir que para uma descrição lógico–semântica $w_1 \leftarrow v \rightarrow w_2 \rightarrow w_3$ na qual Roberto vive em w_1 não existe nenhuma interpretação de mundo possível. Consequentemente, qualquer definição de C alto é trivialmente possível considerando que qualquer mundo possível pode ser definido.

No mundo possível w_1 "Roberto acredita no custo" C para qualquer definição de C (A.4)

A.4 Relacionamentos

Os relacionamentos em lógica modal descrevem as possíveis abstrações de conceitos entre mundos possíveis. Tipicamente, os relacionamentos mais utilizados são,

[3] É relevante ressaltar que não existe obrigatoriedade que C seja verdadeiro em todos mundos possíveis.

Reflexividade existe uma seta $w \rightarrow w$ de um mundo possível para ele próprio.

Transitividade se existem setas $w_1 \rightarrow w_2 \rightarrow w_3$ também existe uma seta $w_1 \rightarrow w_3$.

Continuação sequências de setas não tem fim; a demonstração mais típica deste relacionamento é através de reflexividade.

Desta forma, é viável obter uma conclusão: é permissível imaginar o mundo corrente como possível. Logo as opiniões de Roberto são sempre verdadeiras no mundo[4] que ele está. Em notação de lógica modal,

$$\text{Em qualquer mundo possível } w \text{ “Roberto acredita no custo” } C \Rightarrow C \tag{A.5}$$

A.5 Lógica Doxástica

A lógica doxástica é uma extensão dos princípios da semântica Kriptke, mais especificamente o “raciocínio sobre opinião”. Entretanto, a demonstração neste caso necessita do relacionamento de transitividade. A transitividade significa que qualquer mundo acessível a partir de v inclui mundos acessíveis um passo adiante (w_1 ou w_2). Desta forma a transitividade corresponde à lei da introspecção,

$$\begin{gathered}\text{Em qualquer mundo possível } w \text{ “Roberto acredita no custo” } C \Rightarrow \\ \text{Roberto acredita que “Roberto acredita no custo” } C\end{gathered} \tag{A.6}$$

A lógica doxástica é bastante útil para formalização do raciocínio. Um exemplo típico é o serviço de comércio eletrônico, por exemplo: como é possível provar que um cliente é quem ele especifica quem é na verdade?

A.6 Lógica Epistemológica

A lógica epistemológica combina os relacionamentos de reflexividade com transitividade obtendo a lógica do conhecimento. Por exemplo, se forma de raciocínio de Roberto é reflexiva e transitiva ocasionará uma opinião baseada na modificação[5] do axiomas A.1, A.2, A.5, A.5 e A.6. Desta forma, a sua utilização por si própria do princípio de conhecimento sobre conhecimento pode ocasionar uma solução verdadeira porém incompleta. Em geral, neste caso, são utilizados axiomas e regras para suplantar esta limitação.

[4] Um mundo possível é uma interpretação de um conceito.
[5] Alterando o verbo acreditar por saber.

A.7 Lógica Deôntica

A lógica deôntica define uma interpretação de melhoria em um mundo possível derivado de outro, ou seja, w_2 obtido de $w_1 \rightarrow w_2$ é uma interpretação "melhorada" do mundo w_1. Neste sentido é possível interpretar conceitos sobre conceitos como *o que deveria ser* ao invés do que o que é verdade. A propriedade de continuação torna–se relevante neste aspecto. Por exemplo, se não é possível imaginar como algo possa ser melhorado como é possível saber como ele deve ser feito?

Adicionalmente, a transitividade é utilizada como base para obtenção de novos axiomas A.1, A.2 e A.6 reinterpretados sob o contexto de obrigatoriedade. A reinterpretação destes axiomas permite especificar outro axioma,

$$\text{Em qualquer mundo } w \text{ "}C\text{ é obrigatório"} \Rightarrow \text{não "não–}C\text{ é obrigatório"} \tag{A.7}$$

Este axioma permite definir um nível de consistência lógica para o conceito de obrigação. A combinação da negação dupla é útil e pode ser abreviada como uma interpretação do conceito "C é permissível",

$$\text{Em qualquer mundo } w \text{ "}C\text{ é permissível"} \Leftrightarrow \text{não "não–}C\text{ é obrigatório"} \tag{A.8}$$

A lógica deôntica é interpretada por alguns autores como inconsistente. Geralmente, este paradoxo pode ser removido através de melhor especificação gramatical e sintática das hipóteses.

A.8 Necessidade

Se as setas de conexão unem *todos mundos possíveis* é possível interpretar este fatos como lógica da necessidade. Portanto um determinado conceito C é necessariamente verdadeiro se ele é verdadeiro em todo mundo possível. Segundo Kriptke, a regra torna–se,

$$\text{"necessariamente } C\text{"} \Leftrightarrow C \text{ é verdadeiro em cada mundo possível } w \tag{A.9}$$

Desta forma, por definição, se "necessariamente C" é verdadeiro em um mundo ele será verdadeiro em todos mundos. Os axiomas A.1, A.2, A.5 e A.6 podem ser definidos por, respectivamente,

necessariamente T para toda tautologia T (A.10)

(“necessariamente C” e necessariamente “$C \Rightarrow Z$”) $\Rightarrow$ necessariamente Z (A.11)

“necessariamente C” $\Rightarrow$ C (em qualquer mundo possível w) (A.12)

“necessariamente C” $\Rightarrow$ necessariamente “necessariamente C ” (A.13)

O conceito “possivelmente C” pode ser definido analogamente ao caso de lógica deôntica (ou “permissível”) como,

não (necessariamente não–C) (A.14)

Esta definição especifica que embora C não precise ser verdadeiro em todos mundos possíveis ele necessita ser verdadeiro em pelo menos um mundo possível. Adicionalmente, é possível deduzir facilmente que se C é necessário ele também é possível.

A.9 Lógica Descritiva

A representação lógica do conhecimento, consequentemente informação, é de interesse fundamental em uma arquitetura como a proposta nesta pesquisa [Gašević, Djurić e Devedžić, 1998]. A semântica de uma representação identifica os conceitos de uma linguagem de informação permitindo que eles sejam identificados como aspectos decidíveis em problemas de Lógica de Primeira Ordem (FOL). A Lógica de Primeira Ordem é mais expressiva que a tradicional Lógica Proposicional pois possui estrutura, ou linguagem, interna através de identificadores e operações dentre eles.

O desenvolvimento deste tipo de representação envolve a identificação do vocabulário de um domínio de conhecimento, ou terminologia, em um **recipiente** (*container*) denominado T–Box e **asserções** sobre os *indivíduos* do domínio em parte da base de conhecimento denominada A–Box. O vocabulário, portanto, consiste de conceitos e sua utilização, ou relações (papel, *role*). Os conceitos formam conjuntos de *indivíduos*. Assumindo que um conceito C tenha uma relação R, as expressões $\exists R.C$ ou $\forall R.C$ significam, respectivamente, que existe um conjunto de indivíduos no conceito C para os quais a relação R existe.

O *dual* T–Box/A–Box permite que seja efetuado raciocínio utilizando asserções e terminologia. Desta forma é possível determinar se um T–Box é *satisfeito*, ou *sem contradição*, e se existe uma *subsunção* (redefinição) entre dois conceitos, ou seja, se um é mais geral que o outro. As asserções associadas com A–Box também permitem inferir se um indivíduo específico é uma *instância* de um determinado conceito bem como se um dado conjunto de asserções é *consistente* (passível de ser modelado). A análise da consistência permite

verificar o conjunto de asserções e a satisfazibilidade confere o conjunto de conceitos em relação a quão significativa e válida é a base de conhecimento.

Finalmente, existe um subconjunto de Lógica Descritiva que permite que regras adicionais sejam definidas junto com o *dual* T–Box/A–Box. Uma regra sob forma operacional é definida como $C \Rightarrow D$: se um indivíduo é uma instância de C logo ele também é uma instância de D. Uma regra na forma declarativa é dada por $\mathbf{K}C \subseteq D$, ou axioma da inclusão, onde **K** restringe a utilização de um axioma a apenas indivíduos que estão no A–Box e nos quais o *dual* T–Box – A–Box infere que eles são instâncias de C.

VITA

Meu *curriculum vitae* é relativamente curto porém sempre focado em engenharia. Eu obtive meu diploma de graduação em engenharia cursado na Universidade Federal do Rio Grande do Sul – UFRGS de Porto Alegre, RS. Quando ainda era estudante realizei estágios nas empresas Sultepa S/A e Cotasul Engenharia. Após a graduação trabalhei na PAVIMAR Engenharia como responsável por projeto estrutural de prédios e construção. Subsequentemente à formatura na graduação, recebi bolsa do IAESTE e fui fazer um estágio prático de projeto e execução na Vägverket Projekteringskontor de Jönköping, Suécia.

Quando o estágio terminou realizei mestrado profissional no CEDEX de Madrid, Espanha como bolsista do Instituto de Cooperación Ibero–Americano (ICI). Após o retorno para o Brasil, ingressei no segundo mestrado: Engenharia de Produção e Sistemas, área de Pesquisa Operacional, na Universidade Federal de Santa Catarina – UFSC em Florianópolis, SC. O doutoramento em *Industrial and Systems Engineering* foi iniciado na Virginia Tech de Blacksburg, Va – EUA infelizmente não foi concluído, pois estava em processo de conclusão após o retorno para o Brasil quando foi interrompido por uma razão externa. Ambos cursos foram realizados como bolsista do CNPq. Desta feita, posteriormente, concluí o doutoramento no Programa de Pós–Graduação em Engenharia Mecânica (PROMEC) da Universidade Federal do Rio Grande do Sul – UFRGS.

Após o retorno ao Brasil, comecei a trabalhar como professor e pesquisador na Faculdade de Engenharia, em cursos/disciplinas de Engenharia de Controle e Automação e Engenharia Mecânica, da PUCRS em Porto Alegre, RS, onde vivo atualmente.

COLOPHON

O colofão descreve as opções utilizadas para geração deste documento, bem como da pesquisa. Inicialmente, vale a pena destacar que somente ferramentas gratuitas e indepedente de sistema operacional foram usadas.

- GNU Emacs 24.5.1 (x86_64-pc-linux-gnu, GTK+ versão 3.16.6) de 17-09-2005 utilizando lgw01-52, modificado por Debian.
- pdfTeX 3.14159265-2.6-1.40.16 (TeX Live 2015/Debian).
- abnTeX2 documentos técnicos e científicos em LaTeX para o formato ABNT com formato BibTeX (<http://www.abntex.net.br/>).

A ontologia foi desenvolvida utilizando Protégé v4.3 com *plug-in* -ontop- para acesso ao banco de dados PostgreSQL v9.4.5.