UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Gerenciamento de Documentação Técnica para Ambientes de Engenharia/CAD com Suporte a Versões

por

JOACIR GIARETTA

Dissertação submetida à avaliação como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Clesio Saraiva dos Santos
Orientador

Porto Alegre, junho de 2001.

CIP – Catalogação na Publicação

Giaretta, Joacir

Gerenciamento de Documentação Técnica para Ambientes de Engenharia/CAD com Suporte a Versões / por Joacir Giaretta. – Porto Alegre: PPGC da UFRGS, 2001.
90 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2001. Orientador: Santos, Clesio Saraiva dos.

1. Aplicações de Engenharia (CAD). 2. Versões. 3. Frameworks. 4. Documentação dos Passos de Projetos. I. Santos, Clesio Saraiva dos. II. Título.

Agradecimentos

Durante dois anos e meio estive envolvido com o mestrado, e muitas coisas aconteceram nesse período. Gostaria de poder agradecer pessoalmente a todos que, direta ou indiretamente, contribuíram para que eu pudesse atingir esse estágio tão almejado. Infelizmente, isso não é possível, por isso agradecerei àqueles que mais estiveram em contato comigo nesse tempo.

Certamente, o primeiro agradecimento deve ser feito a DEUS, que me guiou e me deu a certeza que teria condições de cumprir meus objetivos e não desistir de alcança-los, em nenhum momento.

Após, gostaria de agradecer ao meu orientador, Prof. Clesio Saraiva dos Santos, pelas dicas e sugestões construtivas acerca do trabalho desenvolvido. A todos os meus colegas de mestrado, pelas horas de sala de aula e pelas festas realizadas. Aos meus funcionários, em especial ao Toni, pelas muitas dicas que me foram passadas em relação à linguagem de programação utilizada para o desenvolvimento do *GerDoc Ábacus*.

Um agradecimento especial àqueles que fazem parte do meu convívio diário, e que me "agüentaram" durante esse período: aos meus pais Moacir e Lorena, que me deram o prazer de conhecer esse mundo maravilhoso; aos meus irmãos Eder e Rono, que fazem tanta falta quando estão longe; à minha ex-namorada Diana, que, apesar de ser ex, me acompanhou e me deu carinho durante a maior parte dessa caminhada. Obrigado, Di!

Finalmente, devo agradecer a mim mesmo por manter a cabeça erguida em muitas situações difíceis que enfrentei. Tenho certeza que, além do aprendizado advindo do estudo realizado, aprendi muitas lições de auto-confiança e auto-suficiência.

Sumário

Lista de Abreviaturas

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
BD	Banco de Dados
CAD	Computer Aided Design
CIM	Computer Integrated Manufacturing
CTKO	CAD Tool Knowledge Object
DBV	Database Version
DPE	Design Planning Engine
DO	Design Object
MHD	Multiversion HyperData
OID	Object Identifier
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language

Lista de Figuras

Lista de Tabelas

Resumo

Ambientes de engenharia apresentam a forte característica da necessidade de cooperação entre projetistas na concepção de projetos CAD, o que provoca uma série de problemas em relação aos ambientes usuais encontrados em aplicações convencionais.

Na busca de solucionar tais problemas, vários recursos e mecanismos relativos às gerências de dados e do processo de projeto são apresentados em vários estudos encontrados na literatura. Boa parte desses recursos estão embutidos nesse trabalho, que visa apresentar um sistema gerenciador de documentação técnica para ambientes de engenharia/CAD chamado *GerDoc Ábacus*.

A proposta da construção do *GerDoc Ábacus* é baseada na busca da solução dos problemas relativos à consistência dos dados de projetos e da integração de tarefas de projetistas que interagem em ambientes distribuídos de projeto. Unindo vários mecanismos, é proposta uma interface totalmente interativa, objetivando manter a harmonia entre projetistas que fazem parte de equipes de projetos que são mantidos em atividade durante longos períodos de tempo, além de documentar todos os passos realizados acerca de cada um desses projetos.

Dessa forma, o *GerDoc Ábacus* é uma ferramenta organizacional e administrativa para projetos de engenharia, sendo de fácil operacionalização, buscando altos níveis de integridade dos dados mantidos.

Palavras-chave: Aplicações de Engenharia (CAD), Versões, Frameworks, Documentação dos Passos de Projetos.

TITLE: "MANAGEMENT OF TECHNICAL DOCUMENTATION FOR DESIGN ENVIRONMENTS, WITH VERSIONS SUPPORT"

Abstract

Design environments require support for the cooperation between designers in the project conception, which is not provided by the environments used in conventional applications.

To provide such support, several resources and mechanisms for data and project management are presented in the literature. Part of those resources is included in the present work, as a management system called GerDoc Ábacus, designed to support technical documentation in project environments.

The main objective of GerDoc Ábacus is to support the project data consistency and the integration of project tasks allocated to designers, which cooperate in a distributed environment. An interactive interface is proposed, and supported by a set of mechanisms able to provide an adequate level of integration between the designers that cooperate in the same project, working together for long periods of time. The system supports also the documentation of all project steps.

So, the GerDoc Ábacus is a tool directed to the organization and the management of engineering projects, designed to be easy to implement and operate, and to provide high level of project data integrity.

Keywords: Design Applications (CAD), Versions, Frameworks, Documentation of the projects steps.

1 Introdução

Projetos CAD (*Computer Aided Design*), ou de engenharia, sempre se caracterizaram por serem compostos por objetos com estruturas complexas, criados por grupos de projetistas trabalhando em ambientes de cooperação. A complexidade desses projetos faz com que vários problemas sejam encontrados, como as múltiplas representações possíveis para um único objeto de projeto. Comum em *ambientes de projetos de engenharia*, a *modularização* separa objetos em sub-objetos, garantindo maior facilidade na sua manipulação, mas cria a necessidade de controlar tais módulos, de maneira que o ambiente de cooperação entre os projetistas não seja afetado.

Pelos fatores já citados e por outros problemas existentes, ambientes de projeto não apresentam características comuns de ambientes usuais, sendo necessário integrar as tarefas dos projetistas através de ambientes distribuídos, controlando o acesso dos projetistas, conforme sua hierarquia. É fundamental, também, controlar os passos dos projetistas e auxilia-los em todas etapas de seus trabalhos. Tais controles são alcançados através de mecanismos adequados de *gerência do processo de projeto* e de *gerência de dados.*

Nesse contexto, surge o conceito de *framework*, que são plataformas que provêem um ambiente operacional que disponibiliza facilidades para a manutenção e o gerencimento dos dados criados, além do conhecimento a respeito do processo de projeto, sendo capazes de conduzir o projetista no desenvolvimento de suas tarefas [SOA96]. Vários *frameworks* destinados a ambientes de engenharia são encontrados na literatura, como o STAR [WAG92], JESSI-Common-Frame [KAT92] e NELSIS [BOS91]. Todos possuem a preocupação de oferecer recursos de gerência do processo de projeto e de dados para garantir a consistência dos ambientes integrados de projeto.

Característica presente em *frameworks*, e principal objeto desse trabalho, a *documentação dos passos de projeto* visa unir os recursos de gerência de dados e do processo de projeto, objetivando manter o histórico do ciclo de vida dos projetos, controlando atividades de projetistas, não permitindo acessos de projetistas não ligados a projetos particulares, gerenciando a modularização dos projetos e o versionamento desses módulos, além de oferecer uma interface totalmente interativa, garantindo facilidades de operacionalização na busca de versões de objetos específicos e no controle da execução das tarefas necessárias ao cumprimento dos objetivos fixados.

A *gerência de dados* de projetos complexos deve, então, garantir a integridade do grande volume de dados gerado nos projetos CAD, que tendem a ser complexos, para que projetistas possam manipula-los de forma consistente, dentro dos projetos em que estão inseridos. Três são os recursos fundamentais para cumprir esse requisito:

① *Controle de Acesso*: o compartilhamento da base de dados entre projetistas de equipes de projetos distintos, ou até mesmo da mesma equipe, e a hierarquia existente entre projetistas, onde projetistas-chefes são responsáveis por equipes de projeto, possuindo, portanto, tarefas específicas, leva à necessidade de um controle de acesso preciso, com perfis de usuários e níveis de acesso que garantem o bom andamento dos trabalhos e evitem acessos indesejados. A granularidade dos direitos de acesso *varia* muito entre os modelos encontrados na literatura. O

ambiente AMPLO [WAG91], por exemplo, possui um controle relacionado às bases de dados existentes;

② *Gerência de Versões e Configurações*: a complexidade dos projetos leva a modularização destes em objetos de projeto, que, por sua vez, podem possuir várias versões, que são estados distintos do mesmo objeto alcançados em tempos distintos. A importância do gerenciamento das versões de objetos de projeto está, exatamente, na possibilidade de retornos a pontos anteriores, na busca de novas alternativas, a fim de alcançar o objetivo traçado. Além disso, várias versões do mesmo objeto podem ser consideradas consistentes, podendo ser utilizadas em configurações diferentes de um mesmo projeto. A área de engenharia é pioneira no estudo de versões e configurações, e vários modelos podem ser encontrados na literatura, como os modelos históricos [BAT85, CHO86, KAT86, DIT88, KAT90] e outros, como [CEL91a, RYK96, GOL95a];

③ *Mecanismos de Navegação e Consulta*: Projetos de engenharia tendem a ter muitos objetos de projeto, e cada um desses tende a possuir várias versões, criadas a partir de tentativas de alcançar consistência. Os projetistas responsáveis pela criação e manutenção dessas versões realizam tarefas delegadas por outros projetistas, de forma que, é necessário que exista um mecanismo para visualizar, através de uma interface gráfica e interativa, as próximas tarefas a serem executadas. Da mesma forma, versões específicas devem ser encontradas de maneira rápida e eficiente. Mecanismos de navegação e consulta realizam essas tarefas, de modo que os projetistas possam encontrar dados desejados seqüencialmente, com ordenações selecionadas, ou através de consultas específicas, atribuindo critérios de seleção.

A *gerência do processo de projeto* deve garantir o total controle das equipes de projetistas que interagem em projetos específicos. Projetos possuem metodologias de desenvolvimento, que devem ser seguidas pelos seus integrantes. Para guiar no cumprimento de tais metodologias, os seguintes recursos são utilizados:

① *Controle do Fluxo de Tarefas*: Visando disciplinar as ações dos projetistas, além de guia-los no cumprimento destas ações, o controle do fluxo de tarefas se torna imprescindível em ambientes complexos de projeto, já que garante que as regras dos projetos sejam rigorosamente seguidas. Destacam-se dois tipos de controles: (1) os *sintáticos*, que estabelecem seqüências pré-definidas de etapas a serem seguidas, não interessando o estado do projeto, e; (2) os *semânticos*, que, dependendo da situação em que o projeto se encontra, apontam o melhor caminho a ser seguido, a partir de regras pré-estabelecidas. A literatura apresenta vários modelos, como o modelo histórico de Chiueh e Katz [CHI90], o esquema de tarefas do framework Odyssey [BRO92a, BRO92b], o sistema Cadweld [DAN89] e o sistema especialista ADAM [KNA86];

② *Gerência de Cooperação entre Projetistas*: Aplicações CAD, denominadas não convencionais, possuem transações de longa duração e necessitam de suporte à cooperação entre projetistas, diferente das aplicações tradicionais. A partir desse enfoque, vários requisitos se tornam fundamentais, como os suportes à longa duração, ao trabalho em grupo, a dados complexos e à evolução de dados.

Visando solucionar a problemática da complexidade de projetos de engenharia e a interação destes com os projetistas, o *GerDoc Ábacus*, um gerenciador de documentação técnica para ambientes de engenharia/CAD, é apresentado neste trabalho como uma ferramenta que assimila recursos de *gerência de dados* e de *gerência do processo de projeto*.

Relativo à *gerência de dados*, o *GerDoc Ábacus* apresenta recursos considerados importantes à manutenção da consistência dos dados mantidos para cada projeto. São eles:

① *Controle de usuários* baseado em *perfis* diferentes, identificados a partir da função exercida por cada projetista dentro da equipe de projeto em que ele faz parte, possuindo *áreas de acesso* distintas, controlando acessos indesejados em áreas impróprias;

② *Mecanismos de navegação e consulta* disponibilizados através do ambiente gráfico do *GerDoc Ábacus*. O principal navegador existente possibilita visualizar listas seqüenciais de dados de projetos, ordenados a partir de várias formas de classificação. Há ainda o navegador de versões, que mostra o grafo de versões de determinado objeto de projeto, e o de fluxo de tarefas, que permite visualizar as próximas tarefas a serem executadas em um objeto de projeto específico. Em relação às consultas, além de uma consulta textual simples, o *GerDoc Ábacus* apresenta uma consulta por predicados, onde o usuário cria e executa suas próprias sentenças de seleção de dados;

③ *Gerência de versões de objetos de projeto*. No *GerDoc Ábacus,* os dados são armazenados na forma <*projeto, objeto de projeto, versão do objeto de projeto*>, onde cada objeto de projeto criado é considerado um objeto versionado. Cada versão criada possui um *documento CAD* associado e pode possuir versões sucessoras. Um *status* indica seu estado atual e apenas uma delas é considerada *versão corrente* do objeto de projeto;

A *gerência do processo de projeto* do *GerDoc Ábacus* possui mecanismos para controlar o trabalho cooperativo dos projetistas envolvidos em projetos específicos. São eles:

① *Controle do fluxo de tarefas* de objetos de projeto, que se trata de um mecanismo meramente *sintático*, guiando projetistas inseridos em projetos nos passos a serem seguidos durante o ciclo de vida de cada objeto de projeto criado. Cada projeto pode ter um *modelo de fluxo de tarefas* distinto, e cada objeto de projeto seguirá o modelo relacionado ao projeto em que está inserido. O *GerDoc Ábacus* exige que seja seguido o fluxo pré-determinado de tarefas;

② *Gerência de dados complexos*. Objetos de projetos de engenharia são, usualmente, compostos por vários outros objetos, que podem estar inseridos em outros projetos. Portanto, cada objeto de projeto é considerado um objeto complexo, e o *GerDoc Ábacus* permite o cadastro da composição de cada versão de objeto de projeto existente, já que versões diferentes de um mesmo objeto podem possuir composições diferentes.

O presente trabalho possui duas propostas: (1) apresentar conceitos e modelos encontrados na literatura acerca dos tópicos anteriormente relacionados,

objetivando identificar os elementos fundamentais para se ter uma eficiente documentação dos passos de projetos de engenharia, e; (2) introduzir o modelo geral do sistema gerenciador de documentação técnica para ambientes de engenharia/CAD *GerDoc Ábacus*, explicitando cada um de seus sub-modelos, além de apresentar sua interface e sua forma de operacionalização.

A finalidade principal do *GerDoc Ábacus* está relacionada com a problemática do gerenciamento de projetos de engenharia em ambientes com características próprias, que possuem, como fatores críticos, a necessidade de modularização dos projetos e a participação de vários projetistas na construção destes módulos. O *GerDoc Ábacus* objetiva organizar equipes de projetistas responsáveis por projetos específicos, a fim de que seja possível controlar todas as tarefas realizadas por cada elemento, através de uma interface amigável e interativa.

O texto está organizado da seguinte forma: o *capítulo 2* descreve as características encontradas em ambientes de projetos de engenharia, destacando a complexidade existente nos projetos, realizando comparações com ambientes usuais, citando problemas e apontando soluções. Nesse mesmo capítulo é dada uma introdução ao tópico *framework* e sua relação com o gerenciamento de documentação dos passos de projetos, onde começa a ficar evidente a importância de mecanismos de *gerência do processo de projeto* e de *gerência de dados*, objetivando garantir a integridade dos dados nos ambientes cooperativos existentes em projetos de engenharia.

O *capítulo 3* destina-se a explicitar os recursos possíveis de *gerência de dados* de projetos complexos, além de exemplificar através de modelos encontrados na literatura. A gerência de versões e configurações é detalhada como solução para as várias representações possíveis de um objeto de projeto em períodos distintos de tempo. Após, mecanismos de controle de acesso são identificados como fundamentais para garantir a segurança nos dados de projetos. Como último tópico, os mecanismos de navegação e consulta são apresentados, salientando a importância de uma interface gráfica e interativa, visando facilitar o trabalho dos projetistas.

O *capítulo 4* aborda recursos fundamentais para o trabalho cooperativo, através de mecanismos de *gerência do processo de projeto complexos*, introduzidos pela gerência do fluxo de tarefas a serem executadas por projetistas integrantes de projetos específicos. Além de conceitos, vários modelos são citados nesse item, tanto de ordem semântica, quanto sintática. A gerência de cooperação em ambientes de projeto fecha o capítulo, mostrando os requisitos para um bom controle do trabalho cooperativo realizado entre projetistas.

O *modelo conceitual do GerDoc Ábacus* é apresentado no *capítulo 5*, onde, através de pequenos modelos, é construído o modelo geral do sistema. As funcionalidades previstas para o *GerDoc Ábacus* são apresentadas e detalhadas, desde o controle de acesso até a forma de armazenamento dos dados no sistema. Esse capítulo visa conceituar todos os mecanismos existentes, para que possam ser encaixados na descrição da sua interface, que é assunto do *capítulo 6*.

O *capítulo 6* apresenta toda a interface do *GerDoc Ábacus*, caracterizando cada elemento componente e relacionando com os modelos explicitados no *capítulo 5*. A área de trabalho do sistema é introduzida como ambiente de operacionalização dos demais módulos. Após, os módulos de criação de modelos de fluxos de tarefas e de criação de usuários e áreas de acesso são detalhados. Como ferramenta importante aos projetistas, é demonstrado o funcionamento do visualizador de documentos CAD. A principal seção do capítulo trata do próprio gerenciamento dos

projetos, indicando as etapas a serem seguidas, desde a criação dos projetos, seus objetos e as versões de cada objeto. O controle do fluxo de tarefas e o gerenciamento das versões também estão inseridos nesse mesmo tópico. Fechando o capítulo, é mostrada a interface dos módulos de navegação e consulta e de relatórios.

Por fim, o *capítulo 7* apresenta conclusões e considerações finais acerca do estudo realizado e da aplicação dos conceitos no sistema gerenciador de documentação técnica para ambientes de engenharia/CAD *GerDoc Ábacus*.

2 Ambientes de Projetos de Engenharia

Este capítulo apresenta as características de ambientes de projetos complexos de engenharia, citando problemas encontrados e apresentando soluções para estes.

A complexidade de projetos é o primeiro tópico abordado, mostrando exatamente os problemas encontrados e possíveis soluções.

Após, a caracterização de ambientes usuais e a necessidade de se ter bons ambientes de projeto são referidos, com o objetivo de conceituar ambientes integrados e abertos de projeto, introduzindo então o principal tópico, que trata de aspectos gerais sobre *frameworks*, referenciando modelos já existentes na literatura, suas características funcionais e estruturais, e a relação existente com o *gerenciamento de documentação dos passos de projeto*, que é o ponto chave do trabalho apresentado.

2.1 Complexidade de projetos

Projetos de engenharia sempre se caracterizaram pela sua complexidade. O processo de um projeto complexo de engenharia cria múltiplas representações para cada objeto que o compõe, em níveis de abstração diferentes. Torna-se necessária, então, a existência de ferramentas adequadas para manipulação destes objetos, garantindo autonomia a projetistas e total controle sobre o projeto em si.

Uma forma usual de dominar o problema da complexidade é a *modularização* [WAG94]. Cada projeto considerado complexo é separado em diversos módulos, permitindo aos projetistas a manipulação individual de cada um e, através da *hierarquização*, dividir módulos em sub-módulos, até que se atinja um nível de simplicidade satisfatória para a manipulação por parte dos projetistas.

Através de um controle dos módulos e sub-módulos de um projeto específico, é possível dividir o trabalho em *equipe*, onde vários projetistas podem trabalhar em módulos distintos, considerando dois fatores que são importantes para garantir a funcionalidade do projeto como um todo: (1) os módulos precisam conhecer a sua hierarquia e; (2) os projetistas precisam trabalhar em ambiente de *cooperação*.

Baseado no fato de que projetistas, trabalhando em ambiente de cooperação, precisam interagir entre si e de que os módulos são manipulados por vários projetistas, torna-se fundamental a *exploração do espaço do projeto*, garantindo o retorno a estados anteriores e, a partir daí, o uso de um conjunto de recursos (ferramentas, dados e opções) de *refinamento de alternativas* para a obtenção de resultados mais próximos dos desejados.

Considera-se também importante a otimização dos passos de um projeto, levando-se em conta o grande volume de dados que um projeto complexo pode gerar. Tal fator somente consegue ser alcançado através de uma exploração inteligente do espaço de projeto.

2.2 Ambiente de Projeto

Um ambiente de projeto deve possuir funcionalidades que possam eliminar os problemas verificados nos ambientes usuais [WAG94, SOA96]. Ambientes usuais possuem fatores que diminuem a produtividade global do processo de projeto:

① Não existe um modelo único de representação de dados. Várias ferramentas são utilizadas, resultando em saídas heterogêneas e possíveis inconsistências e redundância de dados;

② Ausência de recursos para gerência de dados, onde os projetistas armazenam os arquivos dos projetos em pastas específicas, não possuindo qualquer informação adicional do arquivo a não ser a sua descrição. A própria hierarquia do projeto se torna desconhecida;

③ Ausência de recursos para gerência do processo do projeto, onde os projetistas não possuem nenhum controle do fluxo de tarefas a ser realizado. Projetistas-chefes não conseguem administrar as atividades de seus subordinados, gerando um grave risco de inconsistência;

Para solucionar os vários problemas apresentados, na busca da redução do tempo de desenvolvimento de um projeto e conseqüente redução de custos, um ambiente de projeto deve:

① Possuir uma boa interface;

② Guiar o projetista no seu trabalho, através de recursos de gerência de projeto;

③ Permitir acessos diferenciados, dependendo da hierarquia de comando dos projetistas, garantindo o bom trabalho cooperativo;

④ Gerenciar e eliminar possíveis inconsistências de dados e controlar as informações da hierarquia dos módulos de um projeto, incluindo a gerência de versões e configurações destes, através de recursos de gerência de dados;

⑤ Possibilitar o trabalho em ambientes distribuídos.

As características citadas garantem um ambiente de projeto dito *ambiente integrado*. O ambiente AMPLO [FRE88] apresenta uma metodologia para o desenvolvimento de um ambiente integrado, conforme *tabela 2.1*.

TABELA 2.1 - Metodologia para desenvolvimento de um ambiente integrado

1	*Definir funcionalidades básicas.* *processo de projeto* *recursos de gerência de dados* *recursos de gerência do processo de projeto*
2	*Definir esquema conceitual que suporte as funcionalidades básicas.*
3	*Implementar o esquema conceitual através de um SGBD.*
4	*Definir modelo uniforme de interação com o usuário.*
5	*Definir e construir ferramentas de acordo com o esquema conceitual e com o modelo de interação.*
6	*Tornar todas as ferramentas acessíveis através de um* ***cockpit*** *(interface a partir da qual o usuário tem acesso a todas as ferramentas).*

Um ambiente de projeto deveria ser, além de integrado, *aberto*. Nem sempre um ambiente integrado é totalmente aberto (AMPLO), e nem sempre um ambiente aberto é totalmente integrado. As características a seguir garantem um ambiente dito *ambiente aberto:*

① Permite integração de novas ferramentas, seja através de novos produtos, ou formatos diferentes criados em uma ferramenta existente;

② Provê comunicação entre ferramentas;

③ Provê uma interface procedural de acesso aos recursos de gerência de dados e de projeto, bem como à base de dados.

Tornar um ambiente totalmente aberto é uma tarefa que pode ser muito complicada, já que existem muitas ferramentas no mercado, cada uma com características particulares e muitas delas são totalmente proprietárias, não permitindo integração com outras ferramentas.

2.3 *Frameworks* e a Relação com a Documentação de um Projeto

A procura de alternativas para redução de tempo de desenvolvimento e custo de projetos complexos levou ao surgimento, na década de 80, dos *frameworks*, que são plataformas que provêem um ambiente operacional comum a ferramentas de projeto (os programas utilizados), disponibilizando facilidades para a manutenção e o gerenciamento dos dados criados (os objetos de projeto), e conhecimento a respeito do processo de projeto, sendo capazes de conduzir o projetista no desenvolvimento de suas tarefas [SOA96].

O objetivo dessa seção não é discutir as tecnologias necessárias ao desenvolvimento de um *framework*, e sim apresentar características esperadas em um *framework* para que ele suporte o desenvolvimento de ambientes de projeto com as propriedades desejadas. Tópicos importantes como gerência de dados e do processo de projeto são abordados e, especificamente, uma técnica utilizada em diversos trabalhos relacionados a *frameworks*, a *documentação dos passos de projeto*. Em tal tópico são mostradas alternativas possíveis na busca de soluções satisfatórias a problemas de gerência de dados e do processo de projeto.

2.3.1 Conceitos Básicos sobre *Frameworks*

Várias definições de *framework* são encontradas na literatura. De forma geral, um *framework* é uma plataforma para a construção de ambientes integrados e abertos de projetos. Ele não pode ser um ambiente em particular para uma determinada aplicação. Ele deve compreender uma coleção de serviços de propósitos gerais que podem ser configurados de maneira ótima para um largo espectro de aplicações.

O principal objetivo de um *framework* é potencializar o aumento da produtividade global do processo de projeto obtido através de ambientes integrados e abertos. Ele deve permitir aos usuários finais [WAG94]:

① Disparar e gerenciar a execução das ferramentas;

② Criar, organizar e gerenciar dados;

③ Visualizar (graficamente) todo o processo de projeto;

④ Realizar tarefas diversas de gerência de projeto, inclusive relativas ao trabalho em equipe.

2.3.2 Visão Funcional de um *Framework*

Em [HEA92] encontram-se várias funcionalidades desejáveis para obtenção de *frameworks* que garantam ambientes de projeto realmente integrados e abertos. Dentre essas funcionalidades, duas são de grande importância ao escopo deste trabalho: (1) *gerência de dados* e; (2) *gerência do processo de projeto*. Ambas são conceituadas nos próximos parágrafos e melhor detalhadas em capítulos posteriores.

A *gerência de dados* deve apresentar recursos para garantir o controle do grande volume de dados gerado por projetos complexos. A relação entre esses dados é de extrema importância para projetistas que trabalham em ambiente de cooperação. A não organização desses dados pode acarretar uma queda de produtividade, uma vez que dados deverão ser procurados, além de gerar um grande risco de utilização de dados incorretos ou desatualizados.

Muitos recursos podem existir, dentre eles:

① Políticas de controle de acesso de diferentes grupos de projetistas aos dados com direitos de acesso configuráveis;

② Controle de versões e configurações de objetos de projeto;

③ Consultas interativas aos dados e a outras informações de gerência de projeto através de consultas textuais e navegadores gráficos;

④ Organização de espaços de trabalho, organizando os dados segundo critérios distintos;

Cabe ressaltar que os recursos citados não devem seguir modelos fixos, deixando por conta dos usuários a definição de suas próprias políticas. Isso faz com que muitos ambientes não se caracterizem totalmente como *frameworks*, apesar de possuir muitas das características apresentadas.

A *gerência do processo de projeto* tem por objetivo básico controlar a execução de um conjunto de tarefas (ou passos de projeto) por uma equipe de projetistas. Cada novo projeto possui uma especificação inicial e um conjunto de requisitos a serem atendidos. Para alcançar tais requisitos, existe uma série de alternativas que devem ser indicadas pelos recursos de gerência do processo de projeto. Assim, projetistas sabem os passos que devem e os que não devem tomar, objetivando automatizar seqüências tediosas de tarefas, forçando disciplina de projeto. Além disso, é útil no auxílio a projetistas inexperientes. Ao conjunto ordenado de tarefas dá-se o nome de *metodologias de projeto*.

Além do controle do fluxo de tarefas, a gerência do processo de projeto deve controlar também as equipes envolvidas em cada projeto, dando suporte ao *trabalho cooperativo*, permitindo o compartilhamento de dados.

A *tabela 2.2* resume os serviços da gerência de dados e da gerência do processo de projeto.

TABELA 2.2 - Serviços das gerências

Gerência de Dados	*- Controle de acesso aos projetistas* *- Controle de versões e configurações* *- Consultas interativas aos dados dos projetos* *- Organização de espaços de projetos*
Gerência do Processo de Projeto	*-Execução de metodologias de projeto* *- Suporte ao trabalho cooperativo*

2.3.3 Visão Estrutural de um *Framework*

Algumas estruturas de *frameworks* são mostradas como exemplos, objetivando apenas ter uma visão do que é necessário para alcançar as várias funcionalidades que garantam ambientes de projeto realmente integrados e abertos.

O primeiro, mostrado na *figura 2.1*, é o *framework* STAR [WAG92a], onde os diversos gerentes realizam as funcionalidades desejadas em camadas consecutivas, de forma que os gerentes das camadas superiores utilizam os recursos dos gerentes das camadas inferiores. Um SGBD com recursos de modelagem, armazenamento e recuperação de objetos complexos forma a camada inferior.

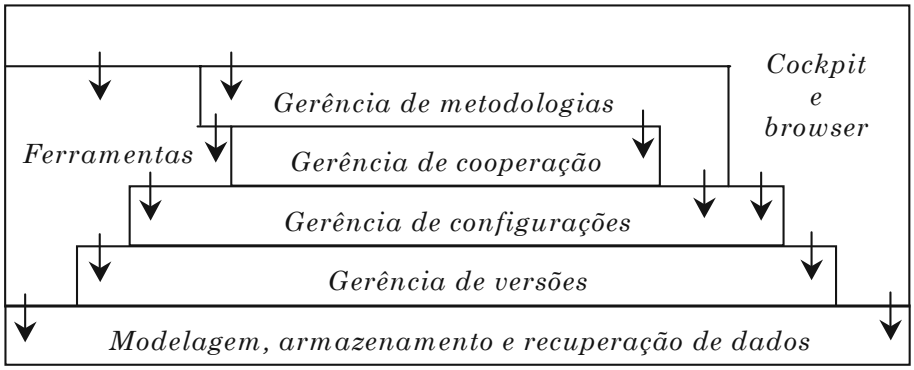


FIGURA 2.1 - Arquitetura do framework STAR

O segundo, mostrado na *figura 2.2*, apresenta módulos de software que se interligam formando os diversos serviços oferecidos pelo *framework* JESSI-Common-Frame [KAT92]. A partir do sistema operacional, os demais serviços surgem, cada um com responsabilidades, como é o caso da gerência de dados, responsável pelo armazenamento, manipulação e recuperação de dados. A camada superior é responsável pela interação com o usuário.

O terceiro, mostrado na *figura 2.3*, apresenta o *framework* NELSIS [BOS91], onde os objetos de projeto e os meta-dados são organizados separadamente. O segundo realiza as relações entre objetos de projeto, tais como versionamento, hierarquia de composição e equivalência funcional.

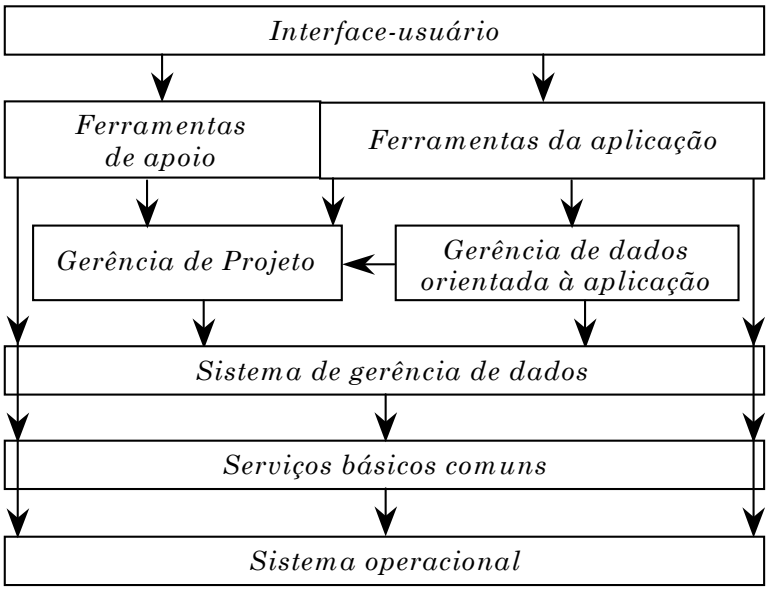


FIGURA 2.2 - JESSI-Common-Frame

É importante perceber que as três arquiteturas apresentadas possuem sistemas de gerência de dados específicos, moldados conforme as necessidades de cada *framework*. Isso é necessário porque os sistemas de gerenciamento de bancos de dados (SGBD's) usuais apresentam deficiências quando aplicados em ambientes de projeto, como [WAG94]:

① Inadequação dos recursos para modelagem, armazenamento, recuperação e consulta de objetos complexos;

② Falta de recursos para controle das várias representações alternativas de um mesmo objeto, através de recursos para gerência de versões e configurações;

③ Inadequação dos mecanismos de verificação de restrições de integridade;

④ Inadequação dos modelos de transações como unidades de consistência, restauração e sincronização;

⑤ Falta de recursos para suporte à cooperação entre projetistas;

⑥ Inadequação dos modelos de controle de concorrência de acessos.

Algumas dessas deficiências são abordadas nos próximos capítulos e o próprio sistema gerenciador de documentos que será apresentado descreverá mecanismos para solução de alguns desses problemas.

2.3.4 Documentação dos Passos de Projeto

Vários trabalhos relativos a *frameworks* apresentam técnicas de *documentação dos passos de projeto*, visando resolver problemas ligados às gerências de dados e do processo de projeto de engenharia/CAD, que se caracterizam por possuírem estruturas complexas e exigirem transações de longa duração, coordenadas por grupos de projetistas em ambientes de cooperação.

Nesse contexto, a documentação dos passos de projeto se torna recurso fundamental dentro de ambientes CAD. Por exemplo, a exploração do espaço de projeto, que permite o retorno a um ponto anterior do projeto para o teste de alternativas, que é de extrema importância para a obtenção dos melhores resultados, não poderia ser feita sem a manutenção do estado do projeto na documentação dos passos de projeto.

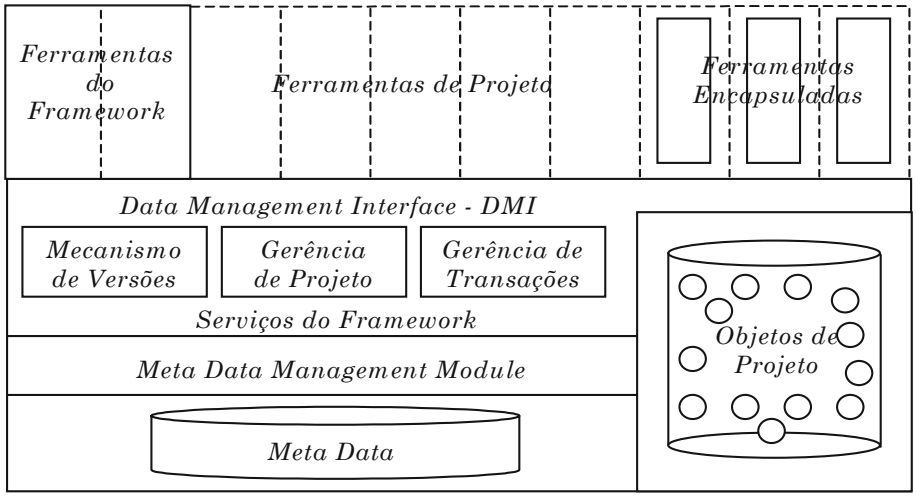


FIGURA 2.3 - Arquitetura do framework NELSIS

A documentação de projetos é um recurso usado para a manutenção de um histórico de tudo o que acontece no desenrolar do processo de projeto: quais as ferramentas utilizadas, quando, por quem, em que ordem, as opções usadas, quais os dados de entrada, quais os dados de saída, entre outros [SOA96]. Esta documentação é aproveitada para:

① O acesso a versões e configurações de objetos pelos projetistas, através de navegadores (*browsers*) gráficos, garantindo a visualização das alterações realizadas durante o tempo de vida do projeto;

② O controle do fluxo de execução de tarefas, garantindo a reutilização de seqüências já usadas com sucesso por outros projetistas, além de dar maior possibilidade de controle pelos projetistas-chefe de projetos específicos;

③ A exploração do espaço de projeto, garantindo o retorno a estados anteriores e, a partir daí, o uso de um conjunto alternativo de recursos (ferramentas, dados e opções) para a obtenção de resultados, quem sabe, mais próximos dos desejados.

Uma das principais carências existentes na área de engenharia é exatamente o controle de documentação da fase de projeto. Para suprir essa carência é necessária uma aplicação que gerencie essa documentação, possuindo um controle eficiente dos dados de projeto e das atividades de projetistas que consiga atingir um bom nível de consistência, garantindo assim, além das facilidades listadas acima, a redução de tempo de desenvolvimento e custo de projetos complexos.

O gerenciamento de documentação de projetos busca manter o histórico de projetos, que em ambientes de engenharia, tendem a ser de longa duração. As informações relativas a cada projeto, como: material utilizado, autores, datas, situação atual, tarefas a executar, relações entre versões, devem ser armazenadas para que todos os projetistas, sem interessar o grau de hierarquia, possam ter as diretrizes para passos futuros.

A implementação de um sistema de gerenciamento de documentação de projetos deve respeitar as duas áreas já citadas anteriormente: (1) *gerência de*

dados, tratando da forma como os dados são acessados e o próprio gerenciamento das versões de projetos, e; (2) *gerência de projeto*, que será responsável pela administração do trabalho em equipe, controlando o fluxo de execução de tarefas.

2.4 Algumas Considerações

Esclarecer aspectos importantes referentes a *ambientes de projetos de engenharia* é fundamental para que se entenda a real necessidade de se ter ambientes integrados e abertos de projeto. Nesse capítulo, através da apresentação da complexidade de projetos de engenharia, foi possível vislumbrar problemas e soluções existentes e a utilização de *frameworks* como alternativa, para conduzir e organizar projetistas no desenvolvimento de tarefas relacionadas a projetos complexos.

Porém, o assunto *frameworks* não é o principal objeto desse trabalho, e sim o *gerenciamento da documentação dos passos de projetos*, que possui relação direta com *frameworks*, por ser uma técnica que absorve as principais funcionalidades neles encontradas, como a gerência de dados e do processo de projeto, assuntos já abordados e aprofundados nos próximos capítulos.

3 A Gerência de Dados de Projetos Complexos

Conforme já visto em seções anteriores, um processo de projeto complexo de engenharia necessita tratar o grande volume de dados gerado e a relação entre esses dados que são manipulados por projetistas trabalhando em ambientes de cooperação. Às facilidades de manipulação de dados dá-se o nome de *gerência de dados*.

Como tópico inicial desse capítulo, são apresentados alguns conceitos e requisitos básicos da gerência de dados, introduzindo os tópicos seguintes, que abordam a gerência de versões e de configurações, o controle de acesso aos dados e os mecanismos de navegação e consulta.

3.1 Conceitos e Requisitos Básicos de Gerência de Dados

3.1.1 Objetos Complexos

Na *seção 2.1* foi caracterizada a complexidade de projetos de engenharia, citando a modularização como forma usual de resolver esse problema. Tal processo resulta em uma composição de subobjetos, de modo que cada subobjeto possa ser projetado e analisado em separado. Tem-se, então, a primeira propriedade de projetos complexos: a *composição*, para uma estratégia *top-down* e *decomposição*, para uma estratégia *bottom-up*.

A segunda propriedade é a *hierarquia*, onde todo objeto pode conter, construída através da composição (ou decomposição), uma hierarquia de subobjetos, formando assim *objetos complexos*.

3.1.2 Estratégias de Projeto e a Relação com Projetistas

Diferentes estratégias podem ser utilizadas para guiar as seqüências de transformações de projetos complexos: (1) estratégia *top-down*, seguindo dos mais altos níveis de abstração para os mais baixos; (2) estratégia *bottom-up*, que agrega subobjetos na busca de um objeto final e; (3) estratégia *meet-in-the-middle*, quando se parte de um nível mais alto de abstração até se obter uma composição de módulos previamente projetados e armazenados em bibliotecas.

O processo todo, não interessando a estratégia adotada, é seguido por projetistas que, a partir de um ponto inicial, vão acrescentando informações de composição ou refinamento. Explorar o espaço de projeto na busca de melhores soluções se torna ponto fundamental nesse instante. Requisitos são estabelecidos, e o projetista terá várias descrições alternativas de módulos para comparações na busca da solução destes requisitos. Tem-se então, além das descrições de um objeto em níveis diferentes de abstração (*vistas*), descrições alternativas do mesmo objeto para fins de comparações (*alternativas*).

O termo *versão* é utilizado para designar uma generalização dos conceitos de *vista* e *alternativa*. A gerência de versões é importante para suportar a evolução de projetos, considerando suas peculiaridades, segundo vistas e alternativas.

O termo *configuração* aparece de formas diferentes em modelos diversos. O framework STAR [WAG92b] adota a seguinte definição: uma representação composta Y^k de um objeto Y contém subobjetos C_i ,..., C_j que são instâncias de

outros objetos X_i ,..., X_j. Como um objeto X_n pode possuir múltiplas versões, deve-se selecionar para sua instância em Y^k uma destas versões, para que a representação Y^k possa ser processada por uma ferramenta de análise ou síntese. Uma *configuração* de Y^k é uma seleção de uma versão para o objeto X_n instanciado por cada subobjeto C_n. A este processo de seleção chama-se de *resolução* configuração. A resolução de configurações pode ser implementada de diversas formas, conforme o mecanismo de controle de configurações empregado. A gerência de versões e configurações é assunto do próximo tópico.

3.2 Gerência de Versões e Configurações

A tecnologia de banco de dados endereçada a aplicações não convencionais, como é o caso de aplicações de engenharia/CAD (Computer Aided Design), tem evoluído muito e, além de permitir o armazenamento de vários estados (versões) dos objetos em tempos distintos, fornece mecanismos para garantir a integridade do banco de dados e evitar redundâncias de dados.

Pode-se conceituar *versionamento de dados* como sendo a habilidade que um sistema gerenciador de banco de dados tem para criar, organizar, gerenciar e manter *versões* distintas de dados ou objetos individuais em certos períodos de tempo. Tal conceito é importante para o domínio de aplicações de engenharia, onde usuários precisam experimentar diferentes versões de objetos, a fim de explorar os vários caminhos possíveis até atingir estados consistentes e concluir com êxito projetos de produtos específicos.

A estrutura complexa dos objetos de projetos e o fato de que projetos exigem transações de longa duração, coordenadas por grupos de projetistas em um ambiente de cooperação, foram pontos fundamentais para o surgimento de diversos estudos na busca de criar mecanismos que suportem tais requisitos. Os modelos pioneiros surgiram dentro da área de aplicações de projeto [BAT85, CHO86, KAT86, DIT88, KAT90], objetivando representar os objetos de projeto e suas necessidades de versionamento. A maioria dos modelos não apresenta maneiras de evitar redundância de dados, buscando apenas garantir a integridade do banco de dados, como acontece em [DIT88], onde as versões são armazenadas em sua totalidade.

Nas próximas seções, alguns modelos encontrados na literatura são descritos, objetivando apresentar algumas contribuições aparentes: (1) mostrar a importância fundamental do controle de versões em ambientes cooperativos, necessários para projetos de engenharia; (2) apontar problemas comuns encontrados nesses ambientes e suas possíveis soluções; (3) apresentar mecanismos úteis para o gerenciamento de versões e objetos versionados, na tentativa de preservar a integridade do banco de dados. A maioria deles é voltado para ambientes CAD, com exceção de [GOL95a], que se trata de um modelo mais genérico.

3.2.1 Suporte a Versões para Sistemas de Banco de Dados de Engenharia

A noção de versão em um sistema de banco de dados pode ser interpretada de diferentes formas, segundo [DIT88]. Em geral, manter versões de uma entidade significa manter representações diferentes dos mesmos aspectos da entidade, tendo independência entre elas. Dessa forma, uma versão é a descrição completa de uma entidade com o grau de abstração escolhido pelo projetista.

Em aplicações de engenharia, um produto tem origem a partir da criação de objetos hipotéticos que, passo a passo, vão sendo modificados para atingir uma descrição completa do produto final que será produzido. Após essa etapa, podem surgir revisões advindas de vários fatores, como: correção de erros, melhores idéias, etc. Projetistas precisam ter informações das versões do produto, pois pode ser necessário retornar a estágios anteriores para a criação de uma nova versão. Por isso, em ambientes de engenharia, é indispensável manter descrições das versões de objetos.

Para o modelo apresentado, um *objeto de projeto* (*DO – Design Object*) é um conjunto de versões com uma chave identificadora, conforme mostra a *figura 3.1*. Uma versão é unicamente identificada por um número de versão dentro do conjunto de versões de um objeto determinado pelo par (*identificador DO*, *número da versão*), que são geradas automaticamente na sua criação. Apenas uma versão pode ser denominada *versão corrente* e uma versão pode ser declarada *congelada* quando o projetista determinar que ela é consistente e pode ser compartilhada com outros usuários, o que significa que exclusões e alterações a partir desse ponto são consideradas ilegais, a não ser que elas sejam *descongeladas*.

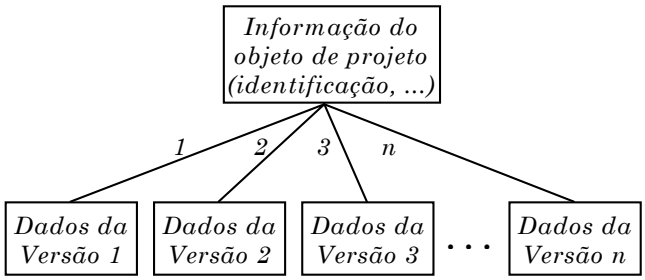


FIGURA 3.1 - Objeto de Projeto

Um conjunto de operações define e manipula versões de acordo com o modelo da *figura 3.1*, divididos em cinco grupos: (1) criação e exclusão de objetos de projeto; (2) criação e exclusão de versões; (2) congelamento e descongelamento de versões; (4) restauração de objetos de projeto e suas versões e; (5) restauração de informações úteis sobre objetos de projeto e suas versões.

Ainda em [DIT88] são encontradas soluções para problemas encontrados no versionamento de projetos complexos, definidos como *projetos hierárquicos*. Em uma segunda parte, o modelo apresenta conceitos referentes a *grupos de versões lógicas*, onde versões podem ser agrupadas em *revisões*, e revisões em *alternativas*.

3.2.2 Um *Framework* para Controle de Versões em um Ambiente CAD

Em [CHO86], como em outros modelos, fica explícita a idéia de que o controle de versões é uma das mais importantes funções que precisam ser suportadas em sistemas CAD integrados, pelo fato de que um projeto, até satisfazer seus requisitos, precisa ter múltiplas versões que devem ser testadas pelos usuários. Isso torna os projetos complexos, pois cada um é composto por componentes de baixo nível, que podem ser compartilhados por outros projetos contidos em bancos de dados distintos ou não.

No modelo proposto foram desenvolvidas soluções para problemas encontrados através de um *framework*, na tentativa de incorporar as características de ambientes CAD e bancos de dados CAD.

O modelo apresenta *ambientes CAD* como ambientes que possuem servidores centrais de redes locais. Os projetistas realizam os seus projetos em suas próprias estações de trabalho. Existem os servidores operacionais, que processam os *jobs* enviados pelas estações de trabalho, e os servidores de banco de dados, que gerenciam os seus *bancos de dados públicos*, além de compartilhar os dados entre os sistemas de *bancos de dados privados* das estações de trabalho.

Um modelo de transações para ambientes CAD foi definido em [BAN85], onde os objetos de projeto (versões) do banco de dados público estão acessíveis para qualquer projetista, e os do banco de dados privado são acessados apenas pelo projetista responsável, que pode compartilhar dados com outros projetistas que pertençam ao mesmo projeto. Isso conduz a noção de *banco de dados de projeto*, que serve como um repositório de objetos de projeto que são utilizados por vários projetistas.

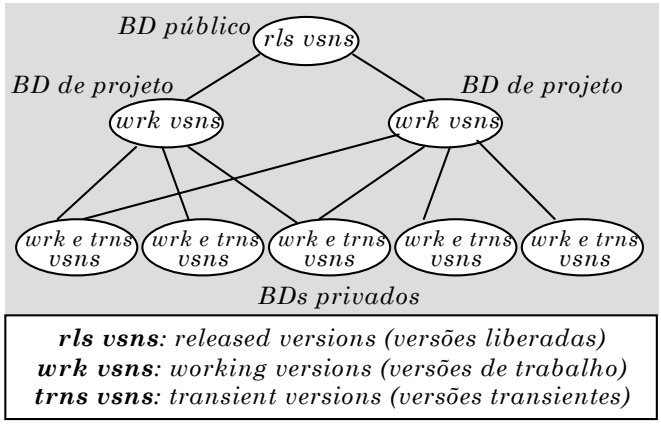


FIGURA 3.2 - Hierarquia do Banco de Dados

É necessário haver uma hierarquia entre os três bancos de dados, conforme mostra a *figura 3.2*, onde o *banco de dados público* mantém os *objetos liberados*, que não podem ser atualizados ou apagados e estão acessíveis para todos os projetistas; os *bancos de dados de projetos* mantêm os *objetos em trabalho*, que estão acessíveis apenas aos projetistas ligados ao projeto e são considerados estáveis, mas não foram testados e, por isso, não podem ser liberados para o *banco de dados público*; os *bancos de dados privados* mantêm *objetos provisórios* e *em trabalho*, que estão acessíveis apenas ao projetista responsável (administrador do banco de dados), sendo que os objetos provisórios podem ser promovidos a objetos em trabalho.

O *modelo de objetos CAD* apresentado foi desenvolvido em [KAT86] e apresenta três tipos de relacionamentos existentes entre os objetos CAD em um banco de dados. Primeiro, um objeto CAD pode ter qualquer número de versões, que podem ser derivadas de uma versão já existente. A descrição de um objeto CAD consiste de duas partes: sua *interface* e *implementação*. Versões são objetos que compartilham a mesma interface, mas tem implementações diferentes. Segundo, um objeto CAD de qualquer complexidade é representado como hierarquia de objetos mais primitivos. A composição hierárquica de um objeto CAD complexo é chamado de *configuração* do objeto e um objeto componente pode ser referenciado por qualquer número de outros objetos. Terceiro, uma versão de um objeto CAD pode ter um número de representações equivalentes, uma derivada da outra.

O modelo de versões apresentado define três tipos de versões distintas, todas em termos de capacidades operacionais, cuja origem está diretamente ligada

aos três tipos de bancos de dados citados nos conceitos acima. São elas: versões *liberadas*, *em trabalho* e *transientes*.

Após a criação de um objeto de projeto, novas versões do objeto podem ser derivadas dele, formando uma *hierarquia de derivação de versão* para o objeto, que captura a evolução do projeto e indica uma ordenação parcial das versões do objeto.

TABELA 3.1 - Ações permitidas aos tipos de versões

	criação	***exclusão***	***atualização***
liberadas	✓	✘	✘
de trabalho	✓	✓	✘
	✓	✓	✓

É necessário haver semântica nas versões, para que não se perca a integridade do banco de dados. Foram criadas regras para as ações de criação, exclusão e atualização aplicáveis sobre os tipos de versões, conforme mostra a *tabela 3.1*.

No modelo ainda são tratadas questões referentes à *notificação de mudanças baseadas em mensagem e baseadas em flag*, com o objetivo de garantir a integridade da hierarquia dos bancos de dados.

3.2.3 Gerência de Versões e Configurações no *Framework* STAR

Através de uma hierarquia flexível de alternativas e vistas, que pode ser adaptada para diferentes aplicações ou objetos de projeto, o modelo STAR gerencia versões de seus objetos, como mostra a *figura 3.3*. Um *Design* pode reunir um número arbitrário de *ViewGroups* e *Views*. Os *ViewGroups*, por sua vez, podem também reunir um número qualquer de outros *ViewGroups* e *Views*, formando-se assim uma hierarquia com qualquer profundidade.

Um *ViewGroup* é, portanto, apenas um nodo intermediário na hierarquia, destinado a agrupar outras representações, seguindo um critério definido a cada projeto. Uma metodologia de projeto, adequada a uma dada aplicação, pode definir um *esquema de objeto* para cada objeto de projeto, estabelecendo uma hierarquia particular de *ViewGroups* e *Views*.

O modelo STAR tem um mecanismo duplo de revisões. Para cada *View* podem ser criados múltiplos *ViewStates*, que também podem acrescentar sinais de interface e atributos próprios. A descrição interna de um objeto encontra-se no interior dos *ViewStates*. Atributos definidos nos nodos do esquema correspondentes a *Design*, *ViewGroup* e *View* também podem ser modificados, criando-se assim revisões consecutivas destes nodos. O sistema gerencia automaticamente as relações entre *ViewStates* e revisões dos demais nodos.

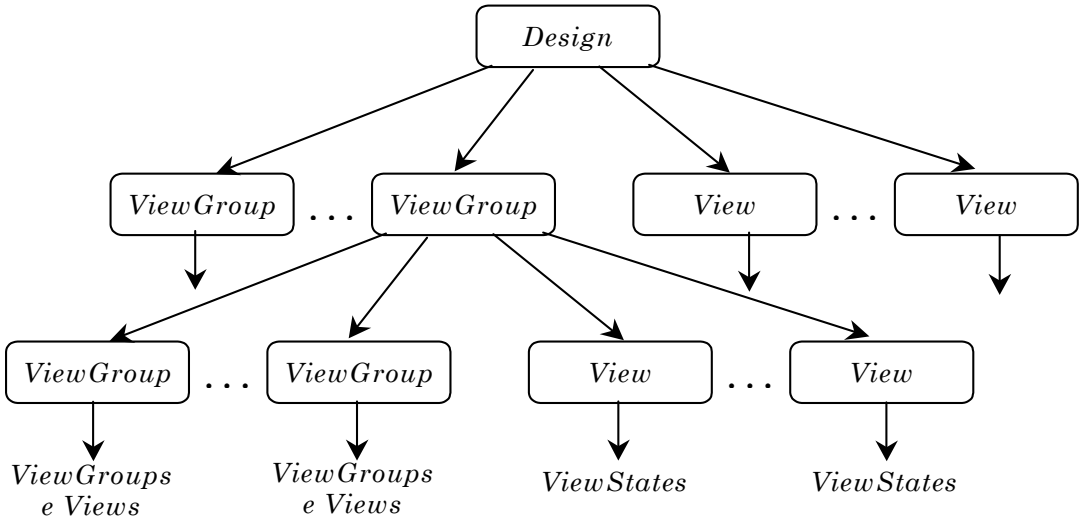


FIGURA 3.3 - Gerência de versões no framework STAR

A gerência de configurações segue a apresentada na seção 3.1.2, onde uma configuração está associada a uma representação composta particular Y^k de um objeto Y, consistindo, portanto, de um conjunto de seleções de objetos e versões destes para os subobjetos existentes em Y^k. Para cada representação Y^k podem existir diversas configurações.

3.2.4 Constelações de Objetos Muiltiversionados para Bancos de Dados CAD

O modelo de [CEL91a] apresenta um *framework* para bancos de dados CAD baseado em *constelações*, que são conjuntos de objetos de projeto multiversionados, que podem ser versionadas em *configurações*. As definições desse modelo foram baseadas em três facetas: (1) objetos de projeto são compostos e existem em múltiplas versões; (2) um processo de projeto é evolucionário e cooperativo, com muitas pessoas envolvidas, de níveis distintos de especialidade e competência, e; (3) não há noção pré-definida de consistência de banco de dados. O modelo [CEL91b], além de tratar de objetos multiversionados, também apresenta esquemas de banco de dados multiversionados.

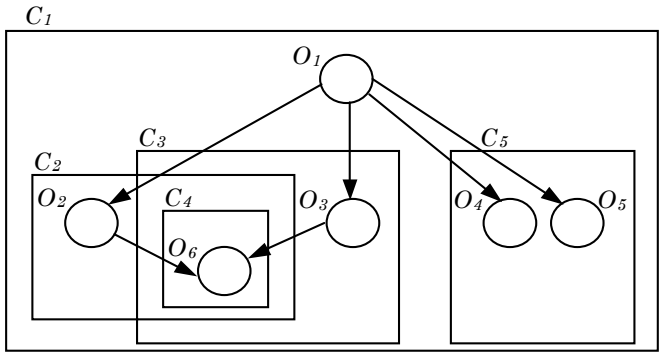


FIGURA 3.4 - Constelações de Objetos

Segundo [CEL91a], usuários de um banco de dados de projeto formam uma hierarquia de projetistas, formando um grupo de trabalho em um projeto comum, que pode ter vários subprojetos. Tais projetos criam objetos de projeto, que podem ter *versões* diferentes, criando assim *objetos versionados* que podem existir em

diversas *variantes*, e são a toda hora combinados em *configurações*. O esquema de um banco de dados CAD não é totalmente definido inicialmente, o que é conhecido por *schema-less*. Isso significa que não há uma noção de consistência pré-definida para objetos de projeto, pois eles mudam dinamicamente no curso de sua transação.

O modelo propõe a organização de bancos de dados CAD na forma de *constelações* de objetos, que podem ser aninhadas, podem se cruzar ou podem ser independentes, como mostra a *figura 3.4*, onde são apresentados seis objetos: O_1 é um artefato composto de quatro objetos (O_2 a O_5), sendo que O_2 e O_3 são compostos (possuem o componente O_6) e O_4 e O_5 são primitivos. O banco de dados está dividido em cinco constelações (C_1 a C_5) que contém objetos, sendo que um objeto pode estar em várias constelações. No exemplo, C_3 está aninhada em C_1, cruzada com C_2, possui C_4 aninhada em si e é independente de C_5. As constelações são consideradas unidades de alocação para os projetistas e cada constelação possui um projetista responsável. Relacionamentos entre constelações implicam em relacionamento entre projetistas. Uma constelação pode ser criada dinamicamente durante o projeto, desde que tenha um projetista responsável por ela.

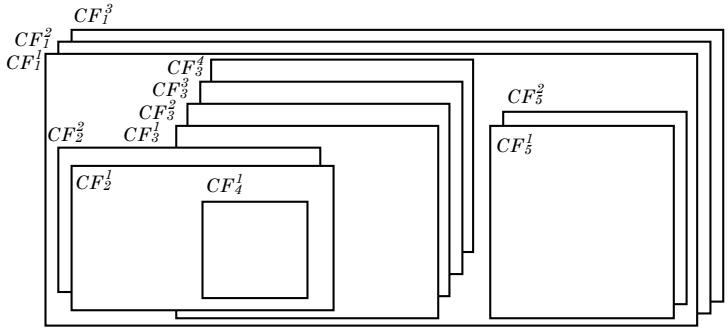


FIGURA 3.5 - Configurações

Ao invés de versionar objetos isolados, constelações inteiras são versionadas, como mostra a *figura 3.5*. Uma versão de uma constelação é chamada de *configuração*, que é composta de versões de todos os objetos que compõem sua constelação. Porém, em muitos casos o escopo da configuração tem que ser reduzida ao subconjunto dos objetos que compõem sua constelação e, por isso, o valor *nil* (nulo) de uma versão é usado e interpretado como versão inexistente. Uma configuração é uma unidade de consistência e os requisitos de consistência dos objetos pertencentes à constelação devem ser impostos pelo projetista responsável. Se há uma constelação aninhada, o projetista da constelação que aninha a outra define requisitos de consistência para as duas (ou mais) constelações.

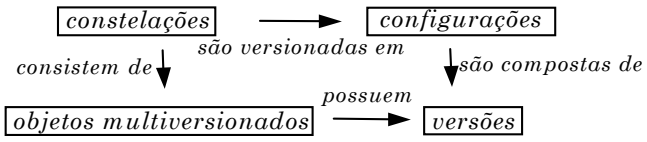


FIGURA 3.6 - Resumo do Modelo

Uma configuração é isolada de todas outras configurações da mesma constelação, ou seja, qualquer atualização em objetos não tem efeito nas outras configurações da mesma constelação. A *figura 3.6* sumariza o modelo apresentado.

3.2.5 Utilização de Bancos de Dados Orientados a Objetos Multiversionados em Sistemas CAD/CIM

O modelo [RYK96] apresenta claramente alguns requisitos de sistemas CAD/CIM[1]: (1) processos de projeto, produção e manutenção devem ser integrados e devem manter a característica de ser um processo cíclico, pois um produto sempre pode ter versões melhoradas, baseadas em versões anteriores do mesmo produto; (2) um produto é projetado em variantes que podem ser reutilizadas, criando a base de um conjunto de elementos unificados; (3) um elemento deve ser representado de formas diferentes, necessárias aos sucessivos estágios de projeto, produção e *marketing*, e; (4) um produto tem que ser projetado em paralelo por um grupo de especialistas de diferentes domínios, seguindo o paradigma da engenharia concorrente. A execução desses requisitos depende da forma de implementação do banco de dados orientado a objetos e o modelo [RYK96] trata disso através do modelo *MHD* (*Multiversion HyperData*), que é baseado no conceito de *versões de banco de dados*.

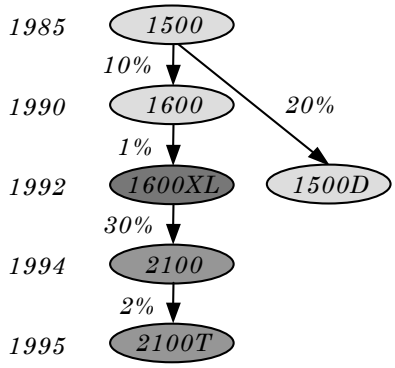


FIGURA 3.7 - Revisões de um carro

Os requisitos de um banco de dados de projeto e as características do modelo *MHD* em relação a eles são apresentados por [RYK96] através do exemplo de um carro, projetado e produzido continuamente durante dez anos, com dez mil partes de diferentes tamanhos, como mostra a *figura 3.7*. No início, a revisão *1500 cm³* é introduzida. Esta revisão foi modificada cinco anos depois, criando a revisão com um motor maior, e sete anos depois, criando uma revisão com um motor diesel. Em paralelo, a revisão *1600 cm³* foi levemente modificada para uma nova denominada *XL*. Dois anos depois uma nova revisão com um motor maior foi introduzido, para um ano mais tarde mudar para a revisão *Turbo*. A figura mostra também o percentual de mudanças realizadas em cada revisão, podendo ser verificada a grande diferença de percentual existente entre as mudanças.

Assim, um problema crucial é não permitir redundância quando se armazenam versões iguais de elementos pertencentes a revisões diferentes. Em [CEL90] é apresentada uma solução para esse problema: *versão de banco de dados* (*DBV – Version Database*). Seções anteriores (outros modelos) já trataram dessa abordagem, porém, algumas características são detalhadas, por se tratar da base para o modelo *MHD*. *DBV* possui dois tipos de versões de objetos: *lógico*, visto por um usuário e *físico*, visto pelo sistema de gerenciamento do banco de dados e usados pelo sistema para armazenar versões lógicas de mesmo valor. Para manter consistência, o banco de dados inteiro é logicamente dividido em *versões de banco de dados* independentes, cada uma representando um estado do mundo real. Uma

[1] *CIM – Computer Integrated Manufacturing*

versão de banco de dados é derivada de outra, formando assim uma *árvore de derivação de versões de banco de dados*.

No nível físico, um objeto multiversionado é um conjunto de versões físicas, que são identificadas por uma estrutura de dados especial chamada *tabela de associação*, responsável por mapear os identificadores de versões de objetos lógicos com os identificadores físicos de versões apropriadas. Duas regras existem para evitar redundância no armazenamento de informações de mapeamento nas tabelas de associação: (1) depois da derivação de uma nova versão de banco de dados, são compartilhadas fisicamente todas as versões de objeto com sua versão de banco de dados pai e; (2) se não há associação de uma versão de objeto físico com uma versão de banco de dados em uma tabela de associação, assume-se que essa versão de banco de dados fica compartilhada com seu pai, o que é chamado de *regra de derivação*.

A representação de variantes de um carro é abordada como um problema por [RYK96], pois cada revisão pode ter variantes, como *pick-up*, *standard*, *turbo*, etc. As variantes das revisões precisam ser modeladas, o que gera alguns novos problemas de consistência de configurações, além da necessidade de realizar operações em um conjunto de variantes em paralelo. O sistema deve decidir quais variantes pertencentes a este conjunto são manipuladas, baseado em critérios externos, como listas de atributos. Outro problema apresentado é a representação de diferentes formas de um carro: descrição de produção, manual do usuário e serviço, lista de acessórios, informação para distribuidores, etc.

O modelo *MHD* pode ser utilizado para representar configurações consistentes de objetos multi-revisão, multi-variante e multi-representação. Tal modelo consiste em capturar informação nos nodos multiversionados, conectado por *links* multiversionados, descritos por conjuntos de atributos multiversionados. São três os tipos de versões existentes: *revisões*, *variantes* e *representações*, usados para modelar modificações ocorridas, maneiras paralelas de desenvolvimento e diferentes formas de objetos. A arquitetura do sistema é composta por dois níveis: *objetos encapsulados* e *descritores*.

As *revisões* são usadas para modelar mudanças de um objeto, como já foi abordado no exemplo da *figura 3.7*. As *variantes* são usadas para modelar maneiras paralelas de desenvolvimento. Todas as variantes de um objeto formam um conjunto, e podem ser representadas como um conjunto de níveis de uma árvore, cujos nodos são descritos por atributos, que são representados por partes de descritores.

A *figura 3.8* mostra um exemplo de árvore de derivação de variantes. Cada nodo representa uma configuração. Ao acessar um nodo-folha é possível descrever a configuração da variante pelo conjunto de atributos alcançados ao percorrer a árvore. As *representações* são usadas para modelar diferentes formas de um objeto, também representadas por uma árvore de derivação. Há a possibilidade de realizar uma operação, chamada *derivação de representação*, em vários nodos dessa árvore, para criar uma representação baseada em outra.

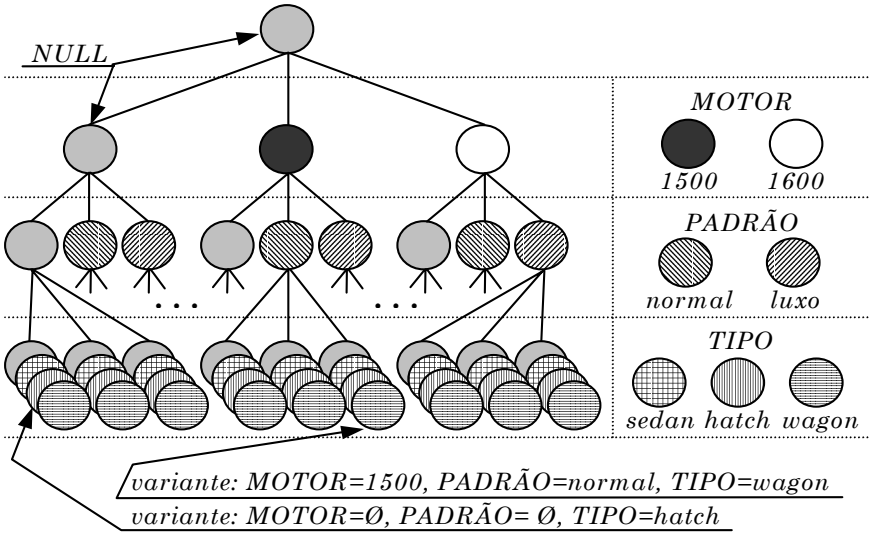


FIGURA 3.8 - Representação de variantes e seus atributos em uma árvore

3.2.6 O Modelo de Versões de Lia Goledziner

Para que se entenda o modelo apresentado em [GOL95a, GOL95b, GOL95c, GOL95d] é necessário compreender os conceitos relativos a versões no contexto de bancos de dados orientados a objetos. Tais conceitos estão descritos em [GOL93] e relatados nos demais artigos que tratam do modelo citado.

Alguns desses conceitos referem-se a versão e objeto versionado. Segundo [GOL93], em um sistema orientado a objetos, uma *versão* é a descrição de um objeto em um determinado momento de tempo, ou sob um determinado ponto de vista, cujo registro é importante para a aplicação.Um *objeto versionado* é constituído por versões de qualquer entidade do mundo real (objeto), devendo ficar agrupadas. Os objetos versionados mantêm informações sobre as versões a eles associados, podendo apresentar propriedades que devem ser comuns a todas as suas versões.

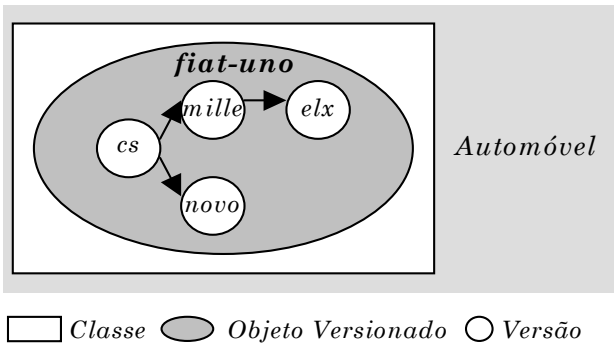


FIGURA 3.9 - Objeto versionado e suas versões

Os objetos, que podem ser versionados ou não, são agrupados em classes se possuírem as mesmas propriedades e comportamento. No exemplo apresentado em [GOL95a] e mostrado na *figura 3.9*, o objeto versionado **fiat-uno** apresenta quatro versões associadas, identificadas por *cs*, *mille*, *elx* e *novo*. Cada objeto versionado possui uma versão que é considerada a *versão corrente*, a qual é mantida automaticamente como a mais recentemente criada, podendo o usuário especificar

uma versão diferente para ser a corrente, tornando-a fixa, não mudando com o surgimento de novas versões.

As versões de um objeto versionado estão relacionadas através de um relacionamento de derivação, formando um *grafo acíclico dirigido*. No exemplo da *figura 3.9*, a versão *cs* em relação à versão *mille* é dita sua antecessora, e a versão *elx*, sua sucessora. Uma versão pode ter várias sucessoras e várias antecessoras. Cada versão tem um *status* que reflete seu estágio de desenvolvimento e/ou consistência, podendo ser: (1) *em trabalho*, que é uma versão temporária e que sofrerá alterações; (2) *estável*, que é uma versão que já atingiu um estágio mais estável e não pode sofrer alterações, mas pode ser removida, ou; (3) *consolidada*, que é uma versão que chegou ao seu estágio final e não pode ser alterada nem removida [GOL95c].

Objetos compostos podem ser construídos utilizando os construtores de objetos de forma recursiva, formando uma *hierarquia de agregação*. As referências de um objeto que apresenta versões e é usado como componente de outro podem ser: *estática*, quando se refere a uma versão específica do objeto; e *dinâmica*, quando se refere ao objeto versionado, podendo indicar qualquer uma das versões do objeto.

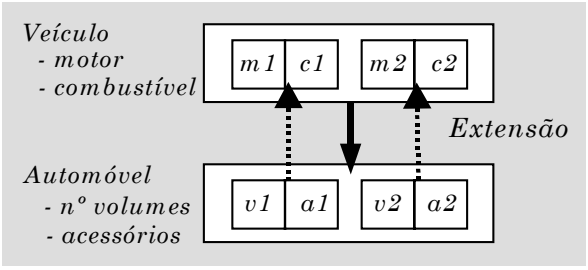


FIGURA 3.10 - Herança por Extensão

O modelo é baseado na herança por extensão[2], onde as propriedades são admitidas nos vários níveis da hierarquia de herança [ATK89]. Extensão está relacionada com a idéia de protótipos. Cada objeto em uma subclasse C2 possui um objeto associado na superclasse C1, aqui denominado de ascendente. O objeto em C2 é dito descendente do objeto em C1, conforme mostra a *figura 3.10*.

No mecanismo de herança por extensão um objeto pode ser desenvolvido em um nível de abstração e posteriormente detalhado nos níveis inferiores da hierarquia de herança. Essa representação de versões nos diversos níveis de hierarquia permite a existência de múltiplos ascendentes para um objeto (versionado, não versionado ou versão) em uma subclasse, devido à possibilidade de o objeto ascendente possuir múltiplas versões.

O modelo permite ao usuário determinar a cardinalidade das *correspondências* ou *mapeamentos* entre versões de um objeto em uma classe e versões de seu ascendente na superclasse. *A tabela 3.2* mostra a classificação das correspondências. Quando é necessário buscar um objeto e seus atributos herdados, a busca inicia na classe mais especializada, sendo escolhido um ascendente para cada superclasse relacionada. O ascendente pode ser especificado pelo seu OID,

[2] A herança por extensão aparece como principal característica do Modelo de Entidades (Modelo E), proposto por [SAN86, SAN89].

cuja estrutura proposta é OID = <id entidade, classe, número de versão>, ou através de um dos critérios pré-definidos: *recent*, *first* ou *current*.

TABELA 3.2 - Classificação das correspondências

1:1	*Cada versão na subclasse corresponde a uma versão na superclasse.*
n:1	*Uma versão na subclasse pode corresponder a somente uma versão na superclasse, mas uma versão na superclasse pode corresponder a várias versões na subclasse.*
1:n	*Cada versão na subclasse pode corresponder a várias versões na superclasse e várias versões na superclasse podem corresponder somente a uma versão na subclasse.*
n:m	*Várias versões na subclasse podem estar relacionadas com uma versão na superclasse e cada versão na subclasse pode corresponder a várias versões na superclasse.*

A *figura 3.11* mostra um exemplo de versões representadas em mais de um nível de hierarquia de herança e suas correspondências, sendo que a classificação é n:m, pois a classe *Automóvel* é definida como subclasse de *Veículo*. A entidade do mundo real **fiat-uno** é representada em dois níveis de abstração: no nível de *Veículo*, onde apresenta as propriedades *motor* e *combustível*, e no nível de *Automóvel*, onde apresenta as propriedades *número de volumes* e *acessórios*. Em cada um dos níveis existem versões associadas a objetos versionados. Cada versão deve possuir pelo menos um ascendente que lhe corresponda, isto é, quando uma versão é criada, obrigatoriamente deve ser ligada a um ascendente.

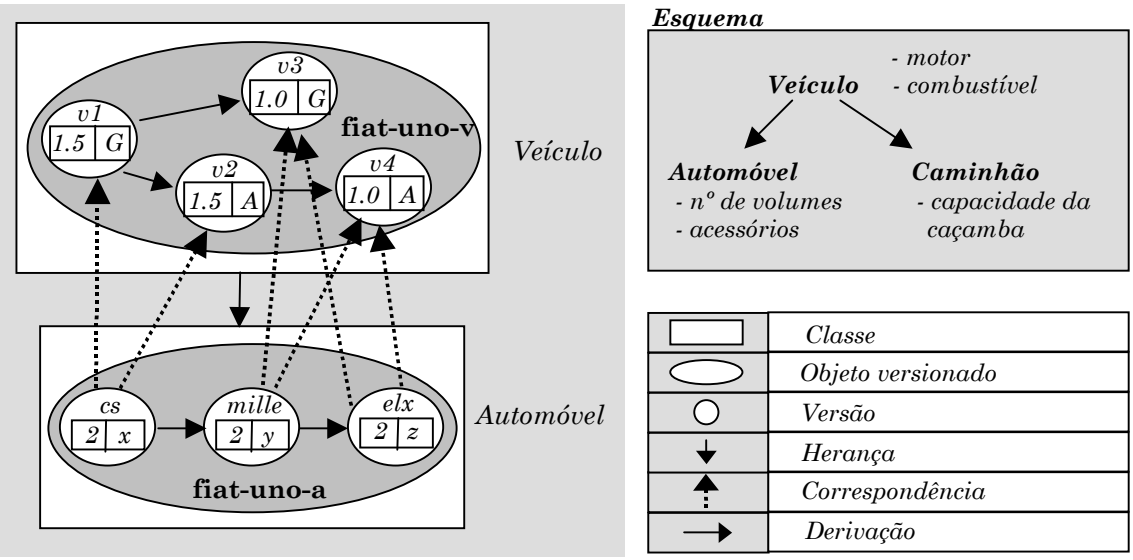


FIGURA 3.11 - Versões representadas em mais de um nível da hierarquia de herança e suas correspondências segundo o esquema

As mesmas características definidas para um **fiat-uno cs**, no nível *Automóvel*, podem ser ligadas a diferentes versões de **fiat-uno** no nível de *Veículo*, resultando em um **fiat-uno cs** a gasolina ou a álcool. Por outro lado, uma mesma versão em uma superclasse pode corresponder a mais de uma versão na subclasse. É o que acontece quando uma versão de **fiat-uno** no nível de *Veículo* corresponde a duas versões no nível de *Automóvel*. Desta forma, o projeto de um novo automóvel pode ser desenvolvido em um nível de abstração e depois detalhado nos níveis inferiores da hierarquia de herança.

3.3 Outras Particularidades Importantes Referentes à Gerência de Dados

Um *framework*, ou sistemas que tenham suas características, ou ainda sistemas que implementem suas técnicas, como é o caso de um *sistema gerenciador de documentos* (principal objeto desse trabalho) necessitam respeitar alguns aspectos importantes. Dois deles são vistos nessa seção: (1) *mecanismos de controle de acesso*, fundamentais para a segurança dos dados e; (2) *mecanismos de navegação e consulta*, fundamentais para visualização, por parte do usuário, dos objetos e representações da base de dados.

3.3.1 Mecanismos de Controle de Acesso

A segurança de dados é aspecto importante em qualquer sistema que possua bases de dados compartilhadas por vários usuários. Em ambientes de engenharia, essa importância é evidenciada pelo fato de usuários trabalharem cooperativamente em projetos.

O mecanismo de *controle de acesso* aplicado sobre a base de dados vai definir o nível de granularidade dos direitos de acesso oferecidos aos perfis de usuários existentes. Perfis de usuários são criados em concordância com os tipos de usuários existentes no ambiente de projeto existente, como por exemplo: administradores, gerentes de equipe, engenheiros, projetistas.

Como exemplo, pode-se citar o ambiente AMPLO [WAG91], onde o controle de acesso está relacionado a uma hierarquia de bases de dados, como mostra a *tabela 3.3*. Em tal ambiente, o esquema de direitos de acesso é fixo.

TABELA 3.3 - Controle de acesso do ambiente AMPLO

Base de Dados	***Permissão de Leitura***	***Permissão de Escrita***
Privada	*Apenas ao projetista responsável pela base de dados*	*Apenas ao projetista responsável pela base de dados*
de Projeto	*Todos os projetistas envolvidos no projeto associado à base de dados*	*Administrador do projeto*
Pública	*Administrador do projeto*	*Administrador geral do sistema*

Através de uma espécie de *gerenciador de usuários e acessos* é possível oferecer recursos mais flexíveis, permitindo a definição de perfis de usuários variados.

3.3.2 Mecanismos de Navegação e Consulta

A manipulação de bases de dados com grandes volumes de objetos e representações se torna complicada se o ambiente não disponibilizar alguma forma de interface gráfica (ou visual) para a visualização adequada dos objetos disponíveis e a relação entre eles. Uma funcionalidade oferecida pelas interfaces gráficas é a *navegação*, genericamente chamadas de *browsers*, que mostram-se úteis na manipulação de informações de bancos de dados. É muito mais prático percorrer um grafo de versões de um objeto versionado para visualizar suas diferentes versões, do que realizar uma busca individual, por exemplo.

Para [WAG94], os *browsers* são ferramentas que devem interagir com o objeto que está sendo analisado naquele instante, oferecendo alguns recursos importantes, entre outros, para:

① Visualizar grupos de objetos ou representações destes segundo diferentes critérios, sempre a partir de um objeto *corrente*;

② Percorrer as diferentes relações, procurando os objetos relacionados ao objeto corrente;

③ Apontar um novo objeto corrente na tela, alterando assim os demais objetos e representações apresentados, de acordo com as relações nas quais o novo objeto corrente está envolvido.

Além da *navegação*, que se caracteriza por ser um processo de visualização seqüencial de informações de um determinado tipo, uma outra funcionalidade é oferecida em interfaces gráficas: a *consulta*. Um mecanismo de consulta se caracteriza por ser um processo de seleção e restrição das informações, de modo que apenas as informações explicitamente solicitadas sejam recuperadas.

É comum encontrar mecanismos de consulta que permitem ao usuário formular consultas gráficas interativamente, através de filtros informados diretamente nas caixas de edição dos atributos, restringindo a navegação para um subconjunto de objetos ou versões. A utilização de *predicados* é muito comum nesses mecanismos. Em uma consulta por predicados, normalmente, o usuário utiliza três elementos: (1) um *atributo* do objeto; (2) um *operador* de comparação ou de conjunto e; (3) um valor *referencial*. Para exemplificar, toma-se o predicado *valor* < *950*, onde *valor* é o atributo do objeto, < é o operador e *950* é o valor referencial.

Tanto os mecanismos de *navegação* quanto os de *consulta* devem ser orientados pelos objetos e relações existentes no modelo de dados implementado no ambiente.

3.4 Relação da Gerência de Dados com o Gerenciamento da Documentação dos Passos de Projetos

A gerência de dados de projetos complexos é apresentada nesse capítulo como forma de manipular de forma eficiente volumes de dados gerados em função da complexidade de projetos de engenharia. A importância desse capítulo encontra-se na relação direta existente entre os conceitos e modelos apresentados com o *gerenciamento da documentação dos passos de projetos*.

A implementação de um sistema que gerencie essa documentação deve garantir a presença de recursos relativos à gerência de dados, como:

① Uma interface visual, com mecanismos de navegação e consulta, objetivando apresentar ao usuário formas interativas de acesso aos objetos complexos que compõem os projetos;

② Um rigoroso controle de usuários e seus acessos, mantendo a segurança no acesso aos dados e não deixando que ações indesejadas sejam realizadas por projetistas sem permissões, além de gerenciar as equipes de usuários que interagem em projetos distintos;

③ Um mecanismo de *controle de versões*, que garanta aos projetistas uma melhor organização nas mudanças ocorridas em projetos, permitindo

retornar a pontos anteriores para testes em busca de melhores alternativas.

4 A Gerência do Processo de Projetos Complexos

Nos diversos tópicos abordados até esse capítulo, comentou-se em várias oportunidades a necessidade da existência de ambientes cooperativos na busca de melhores alternativas de sincronização de atividades e controle de projetos de engenharia. Isso devido à complexidade existente nos projetos, o que exige uma separação em diversos módulos, fazendo com que projetistas consigam manipular individualmente cada um deles até que se atinja um nível de consistência satisfatório em termos do projeto como um todo.

A fim de entender as características que compõem a *gerência do processo de projeto complexos* e os mecanismos utilizados para satisfazê-las, esse capítulo busca apresentar dois itens fundamentais nesse processo:

① O *controle do fluxo de tarefas*, onde, dependendo do estado de um projeto, tarefas podem ser delegadas a projetistas específicos, objetivando automatizar seqüências tediosas de tarefas, forçar disciplina de projeto, bem como ajudar projetistas novatos;

② A *gerência de cooperação* entre projetistas, para que o processo de projeto possa ser modularizado e os resultados compartilhados.

4.1 O Controle do Fluxo de Tarefas e Modelos Existentes

Em termos de tarefas, projetistas podem ter a responsabilidade de executar tarefas meramente simples, onde é clara a ação a ser realizada, mas não o objetivo de tal ato. Tarefas mais complexas já são voltadas ao problema em questão, onde é necessário vislumbrar os objetivos da ação a ser realizada pelo projetista.

Existem, portanto, controles puramente *sintáticos*, onde é seguida uma seqüência pré-definida de passos, independente da situação do projeto, e controles com determinado nível de *semântica*, onde há mecanismos de apoio ao projetista na escolha do caminho a seguir.

Conforme visto na *seção 2.3.2*, o termo *metodologias de projeto* é aplicado ao conjunto de regras que guiam as atividades realizadas pelos projetistas. Para gerenciar essas metodologias de projeto, normalmente são utilizadas regras para controle da execução de fluxos de tarefa, objetivando apresentar a tarefa a ser executada, dependendo do estado atual do projeto. Conforme já foi comentado, dependendo do tipo de controle seguido, a tarefa a ser executada pode fazer parte de uma seqüência pré-definida ou partir de condições calculadas pelo próprio sistema, dependendo dos valores dos atributos dos objetos.

Nas próximas seções são apresentados alguns modelos existentes na literatura, com a finalidade principal de mostrar as diferenças existentes entre modelos sintáticos e semânticos.

4.1.1 Modelo Histórico de Chiueh e Katz

O modelo histórico de Chiueh e Katz apresenta um gerenciamento de tarefas composto de três partes: (1) um compilador que traduz as especificações para *modelos de tarefas*, que são armazenadas na base de dados; (2) um

gerenciador que interage com os projetistas, e; (3) um *visualizador de tarefas*, que é uma ferramenta de depuração para a especificação das tarefas.

A interação *projetista-execução de tarefas* é realizada por um gerenciador, que guiará o usuário, mostrando o ponto onde este se encontra dentro de uma *atividade de projeto*, que são seqüências explícitas de tarefas, na forma de uma árvore, conforme *figura 4.1*, definidas pelo usuário através de uma ferramenta gráfico-interativa.

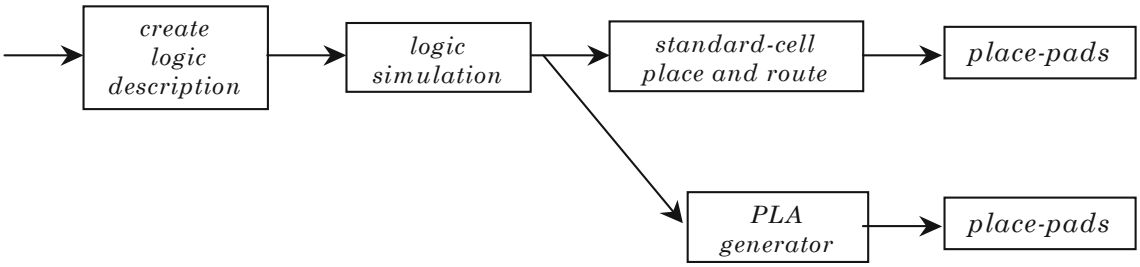


FIGURA 4.1 - Exemplo de atividade de projeto

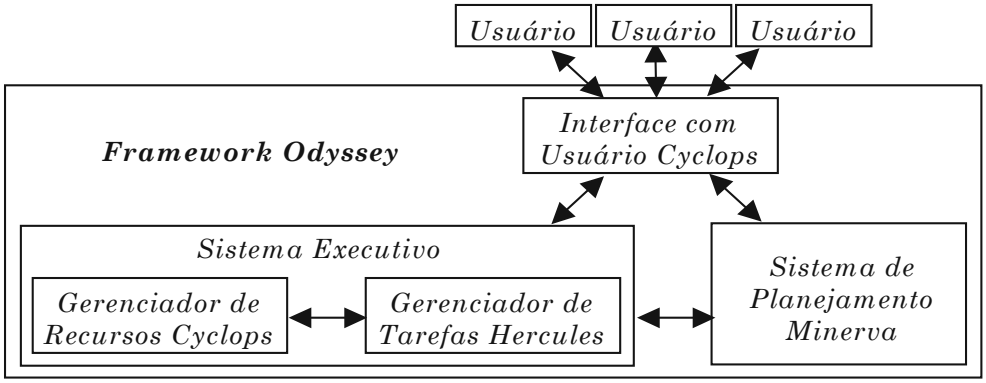
Dependendo da situação, pode haver mais de uma tarefa como alternativa a seguir. Cabe ao usuário decidir a tarefa que deve ser executada. No momento da seleção da tarefa, são solicitadas suas entradas, saídas e parâmetros. O gerenciador verifica todas as operações efetuadas, retornando erros, se for o caso.

Cada atividade possui uma organização que permite retornos a pontos anteriores. Isso é feito através de um *cursor atual*, que aponta para o ponto em que será inserido o registro de história da tarefa subseqüente. Após a inserção deste registro, o cursor é avançado. Essa possibilidade de reposicionamento do cursor em pontos anteriores torna possível a exploração de alternativas pela aplicação de diferentes seqüências de operações a um mesmo estado da atividade.

O modelo apresentado em [CHI90] aborda uma *modelagem explícita de controle sintático do fluxo de tarefas*, característica facilmente verificada através das seqüências de tarefas criadas pelos próprios usuários.

4.1.2 O Esquema de Tarefas do *Framework Odyssey*

O *framework* Odyssey apresenta três componentes principais, como mostra a *figura 4.2*: (1) O Gerenciador de Recursos *Cyclops*; (2) o Gerenciador de Tarefas *Hercules*, e; (3) o Sistema de Planejamento *Minerva*. O Gerenciador de Tarefas *Hercules* é o item abordado nessa seção.



S FIGURA 4.2 - Arquitetura do Framework Odissey

egundo [BRO92a], *tarefas* são funções abstratas de projeto necessárias para alcançar objetivos específicos de projeto, independente das ferramentas empregadas. O *Hercules* baseia seu gerenciamento em um modelo chamado *esquema de tarefas*, que é uma dependência gráfica de todas as tarefas e objetos previstos no ambiente. A *figura 4.3* mostra uma árvore de tarefas abstraída do esquema de tarefas pelo próprio sistema. A árvore é uma *template* que mostra os dados e feramentas dos quais este objeto depende. A execução do *template* cria uma instância deste, que pode ser re-executada se for necessário.

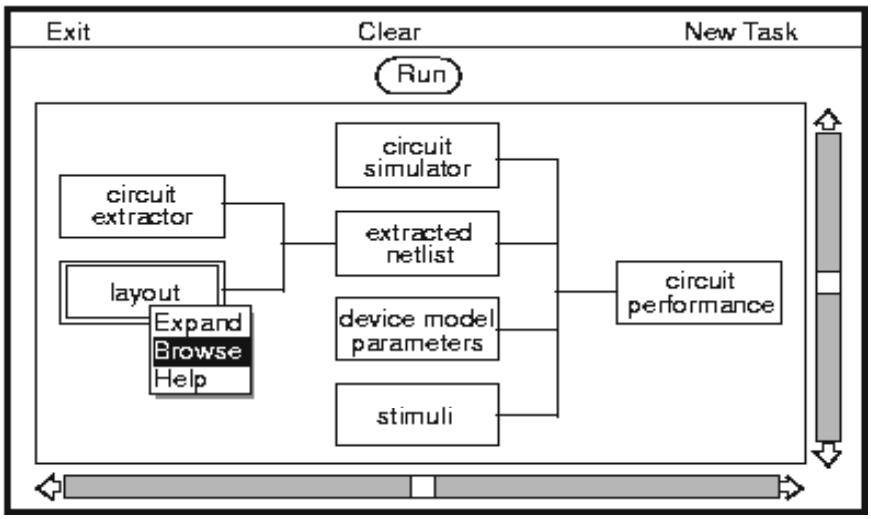


FIGURA 4.3 - Árvore de Tarefas

O esquema de [BRO92a, BRO92b, SUT93] apresenta uma abordagem conhecida como *modelagem do fluxo de dados*, onde existem grafos de dependências de dados, sendo a seqüência de tarefas derivada da existência de dados de entrada para as tarefas. Também é baseado em um controle puramente sintático do fluxo de tarefas.

4.1.3 O Modelo Baseado em Orientação a Objetos *Cadweld*

Com a finalidade de apresentar um exemplo de modelo com controle *semântico* do fluxo de tarefas, será apresentado nos próximos parágrafos o sistema *Cadweld* [DAN89]. O controle semântico objetiva oferecer apoio ao usuário na escolha do caminho a seguir, auxiliando na seleção da melhor tarefa subseqüente, em função dos objetivos a serem alcançados e das restrições a serem atendidas.

O sistema *Cadweld* apresenta uma visão orientada a objeto das ferramentas de CAD que, juntamente com seu modelo representativo constituem um CTKO (*CAD Tool Knowledge Object*), controlando a interação entre a ferramenta e o projetista. Para a comunicação entre as ferramentas, o projetista e outros módulos dentro do *framework*, é proposto um modelo *blackboard*, que responde a informações nele colocadas de forma distribuída.

Tarefas são descritas hierarquicamente. Uma tarefa que necessita executar uma sub-tarefa coloca um pedido no *blackboard*, com informações a respeito desta sub-tarefa. Os CTKO's monitoram constantemente o *blackboard*, e aqueles que sejam capazes de atender à solicitação se voluntariam. O CTKO requisitante interroga os CTKO's candidatos a respeito de suas funcionalidades e capacidades, escolhendo o mais conveniente e disparando sua execução.

Após isso, pode ser realizado um *backtracking*, escolhendo outra alternativa. Isso ocorre até que os resultados desejados são obtidos, quando o CTKO os armazena de forma permanente no *blackboard*. O sistema é capaz de manter a história das decisões e dados para possíveis *backtracks*. Para isso, usa um mecanismo de *log* para salvar dados do contexto, bem como um servidor de versões de arquivo, para guardar arquivos que foram modificados.

4.1.4 O Sistema Especialista *ADAM*

Os chamados planos de projeto são construídos e executados pelo DPE (*Design Planning Engine*) do ambiente de projeto *ADAM* [KNA86], que é um sistema especialista projetado para construir seqüências de atividades de projeto que não estão explicitamente codificadas no sistema, mas que são construídas em resposta a necessidades de um conjunto particular de especificações e restrições. Através de um conjunto próprio de regras, é construído um plano de projeto concatenando membros de um conjunto de possíveis ferramentas e tarefas.

Pré-condições determinam o conhecimento do DPE acerca das tarefas que podem ser aplicadas para a construção de um plano possível, partindo de um estado de projeto inicial. Avaliações são realizadas para cada caminho alternativo que surge no plano, dependendo do estado atual do projeto e as pré-condições estabelecidas. Depois de selecionado o melhor caminho, o plano atinge um novo estado de projeto, sendo repetido a partir deste o processo de construção.

Bactrackings são realizados caso o estado final desejado não seja alcançado, pesquisando novas alternativas e criando um novo plano alternativo. Um problema desse sistema é que ele tende a ser lento, uma vez que as regras de conhecimento são avaliadas e disparadas em tempo de execução.

O sistema *ADAM* não se caracteriza apenas como um modelo de controle semântico de fluxo de tarefas, mas como um *planejador de projeto*, que objetiva não apenas controlar a execução do fluxo de tarefas, mas também construir planos de projeto a partir de informações fornecidas pelo próprio usuário.

4.2 A Gerência de Transações em Ambientes Cooperativos de Projeto

Segundo [LIM97], *transações* têm por objetivo manter de forma transparente a consistência dos dados. Tradicionalmente isso têm sido feito através de mecanismos que preservam propriedades como atomicidade e seriabilidade. As aplicações tradicionais caracterizam-se por possuir curta duração e não necessitar de cooperação. O gerenciamento de transações baseado em seriabilidade agrupa um conjunto de operações em unidades, as quais possuem as seguintes propriedades, seguindo o princípio ACID, conforme *tabela 4.1*.

No entanto, em ambientes de projeto na área de engenharia há o requisito de suporte à cooperação entre projetistas que não é atendido adequadamente pelos sistemas de banco de dados convencionais. Nesses ambientes onde existem aplicações avançadas, o gerenciamento de transações deve ser mais flexível, permitindo a cooperação entre usuários, fornecendo suporte à longa duração e a objetos complexos, para os quais é necessário manter várias versões.

TABELA 4.1 - Propriedades ACID

Atomicidade	*Todas as operações pertencentes a uma transação são executadas com sucesso ou nenhuma delas é executada.*
Consistência	*O conjunto de itens de dados do sistema obedece a algumas assertivas que permanecem válidas após a execução da transação.*
Isolamento	*Estados intermediários de uma transação não devem ser observados por outras transações.*
Durabilidade	*Uma vez confirmadas as atualizações de uma transação, estas devem ser mantidas, mesmo se falhas ocorrerem.*

Baseado nesse contexto, modelos de cooperação foram desenvolvidos para atender o requisito citado e, essa seção objetiva dar uma visão geral de alguns deles, apresentando suas importâncias em relação à gerência do processo de projeto, já que procuram controlar o seqüenciamento de tarefas, visando garantir a consistência dos dados.

4.2.1 Características de Modelos de Cooperação Existentes

Os modelos de cooperação existentes na literatura revisaram os mecanismos de controle de concorrência e recuperação tradicionais para um ambiente em que as aplicações têm longa duração e usuários desenvolvem tarefas cooperativamente. Os novos mecanismos propostos consideram os seguintes aspectos, segundo [LIM97]:

① A longa duração aumenta o número de conflitos entre transações. Resolver conflitos através de bloqueios ou cancelamentos de transações, que são métodos convencionais, degrada o desempenho do sistema;

② O critério de correção baseado em seriabilidade impossibilita a interação entre usuários trabalhando cooperativamente;

③ Desfazer totalmente uma transação interrompida por uma falha pode ser inaceitável quando essa transação possui longa duração.

A partir desses aspectos, surgem novos requisitos a serem observados pelo modelo de transações, que torna-se um modelo de cooperação:

① **Suporte à longa duração:** quando transações têm longa duração é inaceitável usar mecanismos baseados em bloqueio ou desfazer totalmente uma transação que já dura semanas;

② **Suporte ao trabalho de grupo:** tarefas de uma aplicação cooperativa são atribuídas a grupos de projetistas, que interagem entre si. Não devem existir barreiras de visibildade;

③ **Suporte a critérios flexíveis de correção:** aplicações cooperativas requerem critérios de correção mais flexíveis que levam em conta sua duração e a necessidade de interações entre membros do mesmo grupo;

④ **Suporte a dados complexos:** algumas aplicações avançadas necessitam modelar os objetos do mundo real que são compostos de outros objetos, sendo necessário um suporte de representação de dados complexos [KAT90];

⑤ **Suporte à evolução dos dados:** novas versões de objetos de projetos surgem a partir de refinamentos. Tais versões devem ser controladas a fim de representar a evolução desses objetos.

A literatura apresenta modelos de cooperação que enfocam a problemática vivida por um grupo de projetistas em ambiente de projeto auxiliado por computador. Alguns deles são vistos nas seções seguintes. Para [IOC91a], esses modelos de cooperação possuem características diferentes e, por isso, critérios de classificação foram definidos para que se possa avaliar as potencialidades de cada um. Dos cinco critérios apresentados, dois já foram vistos na gerência de dados: (1) *integração a um mecanismo de versões*, e; (2) *controle do acesso compartilhado aos objetos*. Os outros três são: (3) *gerência de projeto*: referenciando basicamente o controle de tarefas, já abordado nesse capítulo; (4) *controle do empréstimo de objetos não estabilizados*: referenciando o trabalho em cooperação, definindo o empréstimo de objetos entre projetistas, e; (5) *suporte a critérios de correção da aplicação*, referenciando a seriabilidade das transações e a inadequação dela quanto ao trabalho cooperativo.

4.2.2 Modelo de Cooperação Proposto por Käfer

No modelo de cooperação proposto por Käfer [KAF91], um grafo acíclico representa relações de dependência entre tarefas. Cada nodo representa uma tarefa a ser executada e os arcos especificam as relações de dependência do tipo tarefa/subtarefa. No exemplo da *figura 4.4*, o projeto de um microprocessador é dividido em um conjunto de subprojetos (memória, ucp, barramento e unidade de ponto flutuante). Esta divisão de projetos em conjuntos de subprojetos é repetida de forma recursiva até que os subprojetos alcancem o nível de complexidade desejado.

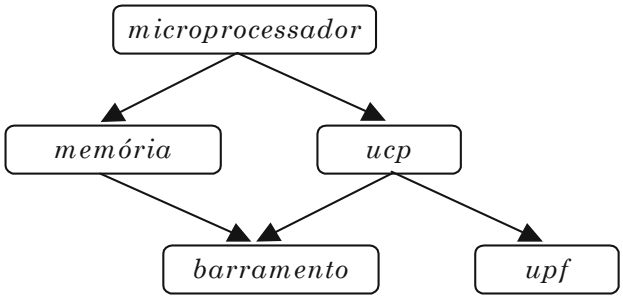


FIGURA 4.4 - Exemplo de grafo de tarefas/objetos

Cada projetista pode possuir dois papéis: o de criador e o de usuário. Objetos possuem um conjunto de características (base para o critério de correção da base de dados), e somente estará terminado quando atingir as características desejadas.

As versões dos objetos são classificadas em:

① **Privativas:** de uso exclusivo de seu criador;

② **Experimentais:** apresentam um subconjunto de características desejáveis, estão disponíveis aos projetistas em geral e estão sujeitas a modificações freqüentes;

③ **Confiáveis:** apresentam o conjunto de características, estão disponíveis aos projetistas em geral, porém, ainda estão sujeitas a modificações;

④ **Liberadas:** não podem mais ser modificadas.

O *empréstimo de objetos* é a forma utilizada para interação de projetistas. Um usuário pode solicitar uma versão emprestada de um objeto e o gerente de cooperação irá localizar uma versão que atenda aos requisitos, concedendo o

empréstimo. Essa versão não pode ser modificada, mas pode ser um subobjeto de um novo objeto. O gerente de cooperação possui papel fundamental nesse processo.

4.2.3 Modelo de Cooperação Proposto por Iochpe

O modelo de Iochpe [IOC91b] é uma extensão daquele apresentado na seção anterior (Käfer [KAF91]), pois utiliza conjuntos de características de objetos para avaliar condições de empréstimo e o processo evolucionário da construção de objetos é materializado por grafos de versões que podem ser percebidos e utilizados pelos projetistas.

No modelo de Käfer, os projetistas não têm uma visão global do projeto, nem de sua evolução. O modelo de Iochpe tenta contornar essa e outras deficiências existentes no modelo de cooperação de Käfer, descrevendo objetos através de:

① Um conjunto de características (especificação de sua versão ao final do projeto);

② Uma ordem parcial de passos para sua implementação;

③ Um grafo de versões, onde cada versão representa uma implementação parcial do objeto, a qual realiza um subconjunto do total de características especificadas para o objeto;

O modelo está centrado nas ordens parciais de passos de projeto que geram os objetos de projeto e nas características destes objetos. Os projetistas dividem o projeto global em subprojetos e associam a cada um destes, um grupo de projetistas. Cada grupo fica responsável pela execução de, pelo menos, um subprojeto. O grupo executa o projeto através do desenvolvimento dos objetos nele previstos.

4.3 A Gerência do Processo de Projeto e a Relação com o Gerenciamento da Documentação dos Passos de Projetos

A gerência do processo de projeto se torna indispensável em ambientes onde projetistas trabalham cooperativamente e necessitam de uma perfeita sincronização de atividades na busca de bons níveis de consistência. Alguns recursos importantes foram abordados nesse capítulo, além de alguns modelos existentes na literatura. Como no capítulo anterior, os conceitos e modelos vistos possuem uma relação direta com o *gerenciamento da documentação dos passos de projetos*.

Basicamente, dois mecanismos são fundamentais em sistemas que gerenciam essa documentação:

① O controle do fluxo de tarefas de projetistas inseridos em equipes de projetos específicos, visando, além de ajudar o próprio projetista em suas ações, auxiliar o projetista-chefe a manter um controle sobre todas as atividades realizadas acerca do projeto;

② A gerência de cooperação entre projetistas, incluindo os suportes ao trabalho em grupo, a dados complexos e a evolução dos dados.

5 O Modelo Conceitual do *GerDoc Ábacus*

O sistema gerenciador de documentação técnica para ambientes de engenharia/CAD *GerDoc Ábacus* possui a estrutura mostrada na *figura 5.1*, onde é possível verificar os recursos utilizados a fim de manipular os objetos complexos de projeto na tentativa de resolver problemas relativos às gerências de dados e do processo de projeto já apresentados no *capítulo 2*.

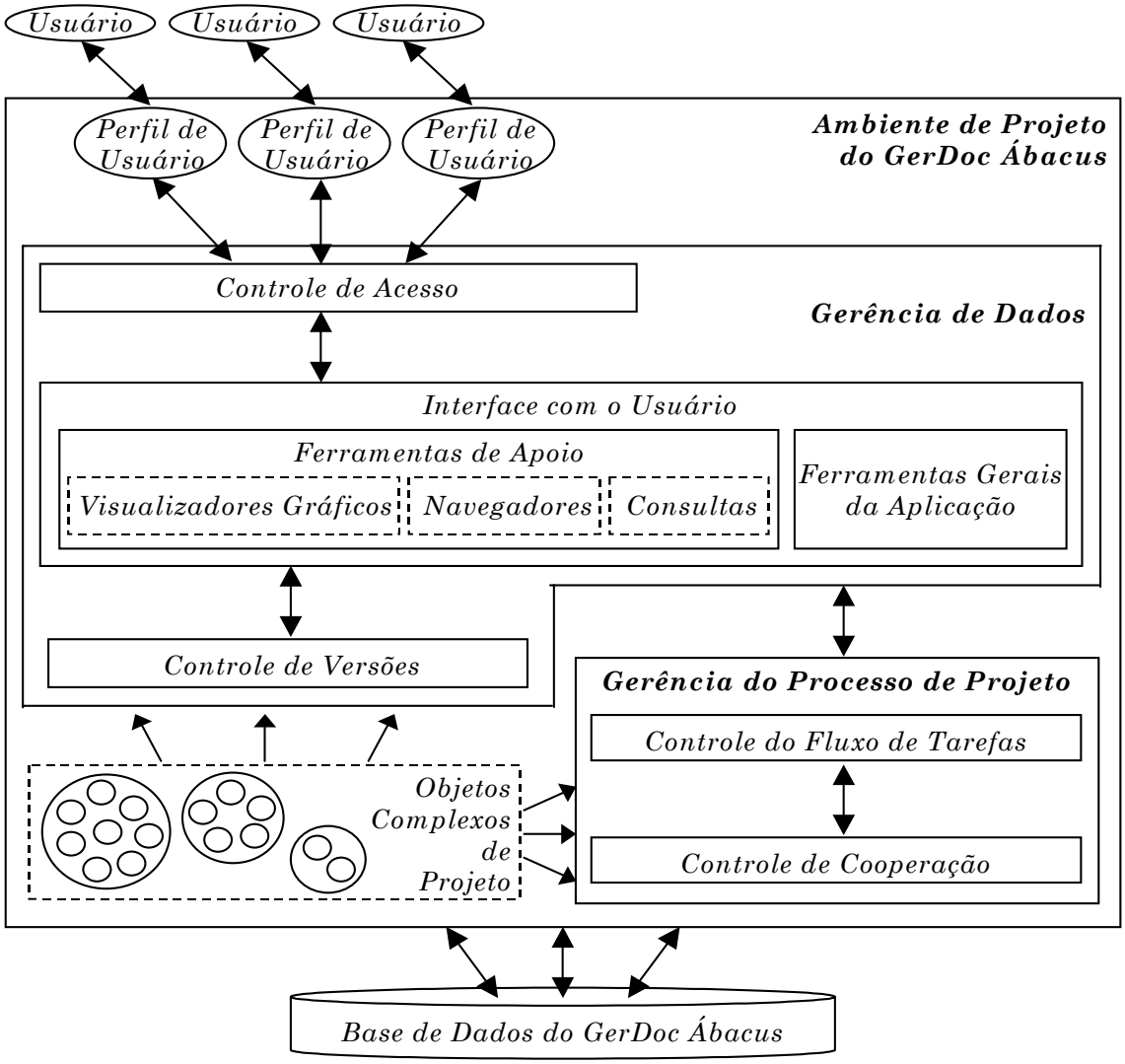


FIGURA 5.1 - Modelo Conceitual GerDoc Ábacus

A documentação de projetos é um recurso fundamental no processo de manutenção de históricos sobre o que acontece no desenrolar do processo de projeto, e a própria exploração do espaço de projeto, possibilitando o retorno a estados anteriores e, a partir daí, seguir novos caminhos na busca de um estado consistente de projeto.

No modelo, é possível visualizar, claramente, que o ambiente de trabalho do *GerDoc Ábacus* possui recursos suficientes para garantir uma certa integridade em sua base de dados, através da *gerência de dados*, com mecanismos de controle de acesso, interação com o usuário e controle de versões e da *gerência do processo de projeto*, com mecanismos de controle do fluxo de tarefas e de cooperação entre projetistas.

Tais recursos, apresentados na *figura 5.1*, e seus modelos individuais, que são modelos elaborados especificamente para o *GerDoc Ábacus*, além de outras funcionalidades do modelo conceitual são mostradas nas próximas seções.

5.1 Funcionalidades previstas para o *GerDoc Ábacus*

As funcionalidades listadas a seguir concentram os resultados buscados em cada modelo que irá ser apresentado posteriormente. É nelas que a construção dos diversos modelos está baseada. Estas funcionalidades são consideradas fundamentais para o próprio sistema *GerDoc Ábacus*, e é apoiado nelas que ele foi construído:

① Usuários devem ter *perfis* diferentes, com *áreas de acesso* distintas, uma vez que projetistas são classificados segundo suas experiências;

② O espaço de trabalho deve ser gráfico e disponibilizar ferramentas diversas de *navegação*, seja na forma de árvore, grafo ou pela simples listagem ordenada segundo critérios de classificação, e *consulta*, dando preferência à utilização de predicados;

③ A designação *objeto de projeto* refere-se a cada objeto específico a ser projetado. Cada objeto de projeto pode possuir *versões* e deve estar associado a um *projeto* específico. Cada versão referencia um *documento*, que é o próprio projeto CAD, desenvolvido em programas específicos de engenharia;

④ A base de dados deve armazenar todas as informações a respeito de cada projeto, disponibilizando-as na forma <*projeto, objeto de projeto, versão do objeto de projeto*>;

⑤ As *versões* de cada objeto de projeto devem ser gerenciadas a fim de permitir retornos a pontos anteriores para testes de melhores alternativas. Deve existir, também, garantia da não ocorrência de alterações indevidas de algumas versões por projetistas sem autorização;

⑥ Cada versão de objeto de projeto pode fazer parte da constituição de outra versão de objeto de projeto, caracterizando assim *objetos complexos*. É considerado objeto complexo, portanto, a versão de um objeto de projeto que possui subobjetos como componentes;

⑦ As tarefas executadas por projetistas devem ser controladas a partir de um *fluxo possível de tarefas* a ser criado por projetistas com autorização, cabendo ao projetista responsável tomar a decisão no caso de existir mais que um caminho a seguir;

⑧ Todas transações realizadas nos objetos de projeto e em suas versões devem ser gerenciadas de uma forma *cooperativa*, já que vários projetistas podem *interagir* nesse contexto.

Nas próximas seções são apresentados os submodelos que compõem o modelo conceitual geral do *GerDoc Ábacus*.

5.2 Controle de Acesso ao *GerDoc Ábacus*

Ambientes cooperativos de engenharia possuem uma forte característica, que acaba por resultar em uma grande preocupação por parte dos engenheiros-chefe de

projetos: o controle das restrições cabíveis aos projetistas que são seus subordinados. Sem esse controle não existe maneira de garantir um bom nível de segurança na manipulação dos objetos de projeto. É necessário haver uma forma de *gerenciamento de usuários e acessos* permitidos, além da criação de *grupos de engenheiros e projetistas*, para que o nível de acesso possa ser diferenciado na relação do projeto com os engenheiros e projetistas que participam e os que não participam dele.

O *GerDoc Ábacus* possui um mecanismo de controle de acesso baseado nessa preocupação. Existe uma classificação fixa dos *perfis de usuários* com as *áreas de acesso* admissíveis a cada um, para após haver uma distribuição destes perfis aos usuários através de *grupos de usuários*, segundo o envolvimento de cada um nos projetos de engenharia existentes.

O mecanismo de acesso aqui proposto possui um nível largo de granularidade, pois define direitos de acesso para grupos de usuários dentro de projetos como um todo, não permitindo delegar acessos, por exemplo, a um único objeto de projeto.

5.2.1 Perfis de Usuários e Áreas de Acesso

O *GerDoc Ábacus* possui atividades de configuração, administrativas e de operacionalização do sistema, o que exige um controle de *perfis de usuários*, conforme mostra a *tabela 5.1*, que recebem permissões de acesso conforme suas responsabilidades dentro de projetos.

As áreas de acesso são referentes às atividades possíveis de serem realizadas dentro do ambiente do sistema, sejam elas atividades no nível de configuração do próprio ambiente, administração de projetos ou a própria manipulação dos objetos de projeto e suas versões.

Uma questão importante é a flexibilidade existente na determinação das áreas de acesso. Apesar do *gerente de ambiente* ser o único a ter permissão de realizar essa distribuição de acessos, o sistema permite que, através dele, novas áreas de acesso sejam incluídas em um perfil específico ou que áreas de acesso já existentes em um perfil específico sejam retirados de sua lista de áreas permitidas.

TABELA 5.1 - Perfis de usuários no GerDoc Ábacus

Perfil	***Características e Responsabilidades***
Gerente de Ambiente	*É o responsável pela configuração geral do ambiente do sistema e sua personalização. É também o único que pode alterar as áreas de acesso disponíveis aos demais perfis. O Gerente de Ambiente possui acesso irrestrito ao sistema, motivo pela qual suas áreas de acesso não são configuráveis.*

Gerente de Engenharia	*Dentro do ambiente do sistema, é responsável pela criação de novos projetos e definição dos grupos de usuários participantes e o perfil de cada um. É também responsável pela construção do fluxo de tarefas a ser executado em projetos. Suas áreas de acesso podem ser alteradas pelo Gerente de Ambiente.*
Engenheiro de Projeto	*É o próprio projetista, responsável pela execução das tarefas determinadas pelo Gerente de Engenharia. Suas áreas de acesso podem ser alteradas pelo Gerente de Ambiente.*
Consultor	*Não possui acesso à execução de tarefas dentro do ambiente do sistema. Pode apenas visualizar a situação de projetos. Suas áreas de acesso podem ser alteradas pelo Gerente de Ambiente.*

Tendo definido completamente os perfis de usuários, é possível criar grupos de usuários para projetos específicos.

5.2.2 Usuários e Grupos de Usuários

Levando em consideração o fato de que um projeto de engenharia exige trabalho cooperativo entre vários projetistas e de que nem todos exercem função de alteração dentro dele, surge o conceito de *grupo de usuários*, que são montados e mantidos pelo *gerente de engenharia* responsável por cada projeto, no momento de sua criação.

Em primeira instância, os usuários são cadastrados pelo(s) *gerente(s) de ambiente* como *consultores* ou *gerentes de engenharia* (apenas os responsáveis pela criação de projetos). Os projetos são criados por esses *gerentes de engenharia*, que montam *grupos de usuários* para cada projeto, indicando o perfil de *engenheiro de projeto* aos projetistas e de *gerente de engenharia* aos usuários que podem ser úteis na execução de tarefas que o gerente de engenharia responsável pelo projeto não pode realizar. Estes últimos serviriam como substitutos do engenheiro responsável pelo projeto.

Os perfis permitidos em cada grupo de usuários são: (1) *gerentes de engenharia*, caso exista mais do que um no projeto, ou caso seja necessário substituição na execução de uma tarefa específica, e; (2) *engenheiros de projeto*, que são os próprios projetistas do projeto. Os que não estiverem cadastrados no grupo de um projeto específico são considerados *consultores* e, como tais, podem apenas visualizar a situação daquele projeto, como de qualquer outro.

Os *gerentes de ambiente* podem ter direitos de alteração sobre um projeto apenas se forem incluídos em um grupo, tendo assim um outro perfil para aquele projeto. Isso significa que ele vai atuar como gerente de ambiente para alterações no ambiente do sistema, mas somente pode alterar projetos se tiver permissão para isso. O único que pode dar essa permissão é o gerente de engenharia responsável pelo projeto.

O objetivo de controlar usuários desta forma é a busca da integridade em cada projeto existente, calcada na responsabilidade que recai sobre o engenheiro-chefe

do projeto. É possível, através desse mecanismo, evitar acessos indesejados de projetistas que não estão ligados ao projeto, tornando mais fácil a tarefa de controlar as ações dos projetistas nos objetos de projeto.

5.3 Mecanismos de Navegação e Consulta Disponíveis no *GerDoc Ábacus*

O *GerDoc Ábacus* possui, como forte característica de sua gerência de dados, a disponibilização de uma *interface gráfica* provida de vários mecanismos de navegação e consulta. Tal interface apresenta facilidades que garantem uma perfeita interação do sistema com o usuário, sendo muito intuitiva e flexível no que diz respeito à forma de acesso aos dados dos projetos de engenharia.

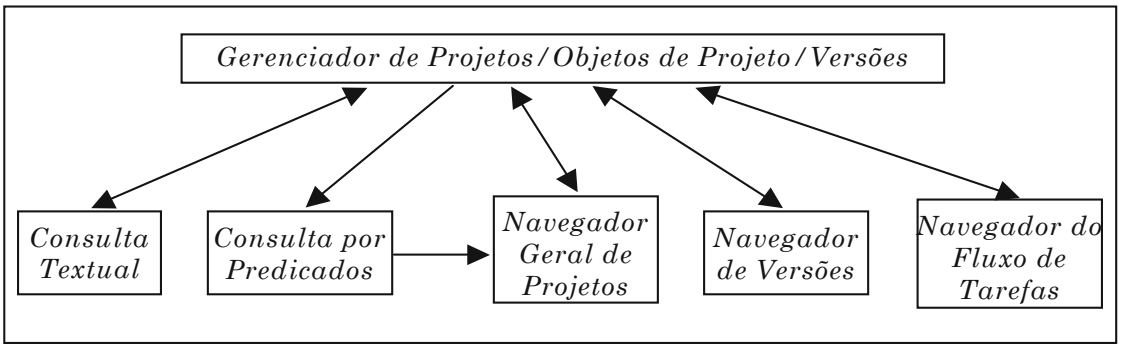


FIGURA 5.2 - Interface Gráfica do GerDoc Ábacus

A *figura 5.2* mostra a estrutura da interface do *GerDoc Ábacus* referente aos mecanismos de navegação e consulta de dados existentes. As próximas seções detalham um pouco mais cada mecanismo.

5.3.1 Gerenciador de Projetos/Objetos de Projeto/Versões

Seguindo as funcionalidades apresentadas na *seção 5.1*, a base de dados do *GerDoc Ábacus* armazena todas as informações a respeito de cada projeto, disponibilizando-as na forma <*projeto, objeto de projeto, versão do objeto de projeto*>. Desta forma, cada projeto criado pode possuir objetos de projeto com versões que conterão documentos, que são os próprios projetos de engenharia/CAD.

Assim, a estrutura da interface do *GerDoc Ábacus* é gerenciada a partir dessa hierarquização. O módulo de manutenção é dividido em quatro partes: (1) *cadastro de projetos*, onde os novos projetos são criados pelos engenheiros de projeto; (2) *cadastro de objetos de projeto*, onde os objetos do projeto selecionado são cadastrados; (3) *cadastro das versões dos objetos de projeto*, onde as versões (documentos) decorrentes de cada objeto de projeto são criadas, e (4) *cadastro das versões de objeto de projeto componentes*, onde ligações com outras versões de outros objetos de projeto são identificadas.

A partir desse módulo de manutenção surgem os mecanismos para consulta e navegação dos dados cadastrados. O grande volume de dados existente faz com que se torne indispensável possuir formas diversas de busca a um dado específico, seja ele um projeto, um objeto de projeto, ou uma versão.

5.3.2 Navegador Geral

O principal navegador existente no *GerDoc Ábacus* funciona seguindo as características encontradas em vários gerenciadores de arquivos conhecidos, onde os dados de cada versão dos objetos de projeto são listados seqüencialmente segundo diversas formas de classificação, que são apresentadas na *tabela 5.2*.

TABELA 5.2 - Formas de Classificação do Navegador Geral do GerDoc Ábacus

Formas de Classificação	***Funcionalidade***
Árvore de Diretórios	*Os dados de cada versão de objeto de projeto são armazenados fisicamente em algum dispositivo de armazenamento. A árvore de diretórios mostra a localização de cada uma dessas versões.*
Lista de Tarefas	*A forma mais simples de verificar as tarefas a serem executadas por um projetista em um objeto de projeto é através da lista de tarefas. O usuário que estiver ativo pode visualizar as suas tarefas pendentes em ordem de prioridade (data e hora de execução limite).*
Por Data	*As versões dos objetos de projeto podem ser visualizadas por ordem de data de criação.*
Tipo de Arquivo	*As versões dos objetos de projeto podem ser listadas conforme o tipo de arquivo em que os respectivos documentos CAD estão armazenados.*
Autor/Versão	*O usuário ativo pode visualizar as versões dos objetos de projeto onde ele foi responsável pela criação, ordenadas por ordem alfabética do autor.*
Projeto/Autor	*As versões de cada objeto de projeto são listadas apresentando os autores responsáveis pela criação. Aparecem ordenadas por ordem alfabética do projeto.*

Ao contrário das consultas, navegadores não permitem filtrar conjuntos de dados. O navegador geral do *GerDoc Ábacus* não foge a essa regra, porém, por ter uma estrutura definida, permite abrir e fechar itens título de conjuntos de dados, conforme a classificação selecionada.

5.3.3 Consultas Textual e por Predicados

O *GerDoc Ábacus* apresenta duas formas distintas de consulta aos dados de projeto. A primeira é a *consulta textual*, caracterizada por uma filtragem de dados baseada em ordenações selecionadas e intervalos digitados pelo usuário. As ordenações e intervalos são pré-definidos pelo próprio sistema, apresentados na tabela 5.3. O resultado da consulta é expresso em uma forma tabular, contendo as versões dos objetos de projeto selecionadas pelo filtro.

A segunda forma é a *consulta por predicados*. Na *seção 3.3.2* foi realizada uma introdução de predicados. Os três elementos então apresentados foram absorvidos pelo sistema da seguinte forma:

① O *atributo* é selecionado através de uma tela interativa, onde o usuário pode clicar no campo selecionado e, automaticamente, é atualizada a caixa de texto que irá construir a expressão SQL;

TABELA 5.3 - Resultados das ordenações e intervalos da consulta textual

Ordenação	***Intervalo***	***Resultado***
Código da Versão	*código inicial ⇨código final*	*Filtro contendo todas as versões contidas no intervalo de código estipulado.*
Projeto	*seleção de um projeto específico*	*Filtro contendo todas as versões do projeto selecionado.*
Data de Criação	*data inicial⇨data final*	*Filtro contendo todas as versões contidas no intervalo de data estipulado.*
Autor	*seleção de um usuário específico*	*Filtro contendo todas as versões do usuário selecionado.*

② O *operador* de comparação ou de conjunto é selecionado a partir de um conjunto de botões. Os operadores são mostrados na *tabela 5.4*;

③ O valor *referencial* é digitado ou selecionado diretamente pelo usuário em uma caixa de texto.

Após alimentar os três elementos, o usuário pode inserir na expressão SQL que está sendo montada. Os botões *incluir* e *excluir* permitem realizar essas ações na expressão e os botões *E* e *OU* permitem realizar ligações de duas expressões.

TABELA 5.4 - Operadores de comparação/conjunto da consulta por predicados

Operador	***Descrição***
=	*Igual*
≠	*Diferente*
>	*Maior*
>=	*Maior ou igual*
<	*Menor*
<=	*Menor ou igual*
⊂	*Contém*
⊄	*Não contém*
{}	*Vazio*
{x}	*Não vazio*
⊢	*Começa com*
[]	*Intervalo*

Por exemplo, se o usuário quiser procurar todas as versões de objeto de projeto onde ele foi responsável pela criação em um projeto específico, terão que ser realizados os passos abaixo:

① Selecionar o atributo *projeto*, o operador = e o referencial *nome do projeto* desejado;

② Inserir na expressão através do botão *incluir*;

③ Inserir o operador lógico *E*;

④ Selecionar o atributo *autor*, o operador = e o referencial *nome do usuário* desejado;

⑤ Inserir na expressão através do botão *incluir*;

A cada inserção, o sistema vai mostrando a expressão SQL que está sendo criada. Quando o usuário terminar, basta executar a expressão para ver o resultado. O sistema mostrará o número de registros encontrados e permitirá ao usuário selecionar a forma de classificação (*tabela 5.2*) do navegador geral. Feito isso, o sistema transfere o controle para o navegador geral, mostrando apenas as versões selecionadas. A qualquer momento pode ser desfeita a seleção.

5.3.4 Navegador de Versões

Comumente, projetistas possuem necessidade de verificar a evolução de um objeto de projeto, já que ele começa por um estado inicial e pode sofrer diversas alterações até atingir um estado satisfatório. A cada alteração realizada sobre um objeto de projeto é criada uma nova versão para ele (o modelo de versões será apresentado posteriormente). Porém, diversas situações podem surgir nesse processo, como a necessidade de retorno a versões anteriores para testes de outras alternativas, e até mesmo a criação de uma versão baseada em outra que não é a versão corrente, objetivando criar uma variação distinta do mesmo objeto.

O *GerDoc Ábacus*, baseado nessas características usualmente encontradas em ambientes de engenharia, oferece uma forma de navegação pelo grafo de versões de objetos de projeto específicos. Através de um ambiente gráfico, é possível selecionar um objeto específico e visualizar toda a sua evolução.

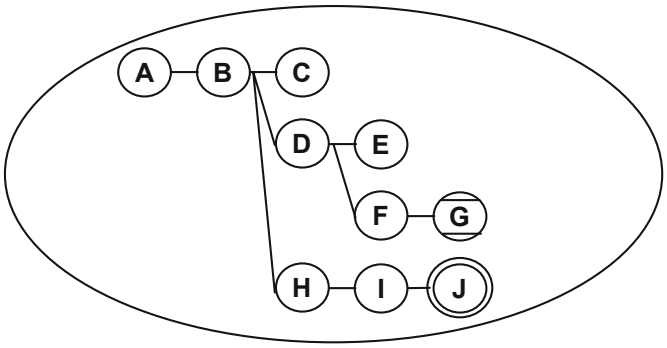


FIGURA 5.3 - Exemplo de grafo de versões de um objeto de projeto

O grafo apresenta, além do *status* de cada versão, identificado através de cores diferentes, uma interatividade com o usuário, pois é possível selecionar cada elemento do grafo e visualizar os dados da versão específica, o projeto CAD

associado a ela, e permite, se for o caso, passar o controle para o módulo de manutenção, onde alterações podem ser realizadas.

A *figura 5.3* identifica a evolução de um objeto de projeto específico, mostrando as diversas tentativas realizadas até se atingir um estado satisfatório para o objeto. A versão considerada corrente é aquela com traços na parte superior e inferior do círculo. A versão que possui dois círculos indica aquela que foi selecionada pelo usuário com fins de consulta. Nesse momento, o navegador irá mostrar dados sobre essa versão e o projeto CAD associado a ela. Adicionalmente, os círculos que representam as versões possuem cores diferentes para identificar o *status* de cada uma.

5.3.5 Navegador do Fluxo de Tarefas

Cada novo projeto pode ser criado por um engenheiro de projeto responsável, que irá criar um grupo de usuários e associará a ele um fluxo de tarefas pré-estabelecido, que deve ser seguido rigorosamente durante o ciclo de vida de um objeto de projeto.

O *GerDoc Ábacus* contempla essa preocupação, apresentando uma forma gráfica e interativa de construção e execução de tarefas por parte dos usuários. A criação é realizada através de um editor de tarefas, que é uma ferramenta gráfica e prática. A execução é de total responsabilidade do *navegador do fluxo de tarefas*, baseada em dois elementos: (1) o *visualizador de tarefas subseqüentes,* e; (2) a *tabela de tarefas executadas*.

O *visualizador de tarefas subseqüentes* permite visualizar, no fluxo de tarefas destinado àquele projeto, os próximos passos a serem seguidos. Na prática, o usuário que realiza determinada tarefa possui a responsabilidade de destinar outra tarefa a outro usuário. A ferramenta permite, então, que possam ser verificadas as opções possíveis a partir da tarefa que foi realizada.

A *tabela de tarefas executadas* armazena o conjunto de tarefas já realizadas durante o ciclo de vida do objeto de projeto. A tabela é formada a partir da tupla <*data, hora, usuário responsável*>, guardando o histórico para consultas futuras.

5.4 Controle de Versões de Objetos de Projeto

Conceitos e modelos de versões foram descritos na *seção 3.2*, tendo como direcionamento único os ambientes de engenharia, compostos de projetos complexos que necessitam realizar testes diversos até alcançar os objetivos propostos.

Tais testes provocam modificações nos objetos de projeto, e é incoerente realizar alterações em objetos que podem se tornar os mais adequados ao final dos experimentos. Modificações realizadas podem não se adaptar ao resultado previsto, sendo necessário retornar à origem das modificações. Essa situação provoca a necessidade de criar vários documentos (projetos CAD) para um mesmo objeto de projeto. Para cada um desses documentos é dado o nome de *versão de objeto de projeto*. A *figura 5.4* mostra a representação das versões de objeto de projeto seguida pelo *GerDoc Ábacus*.

O ponto crítico desse processo é o *gerenciamento das versões*, já que grupos de projetistas interagem no ambiente de construção de um projeto, todos em busca do

objetivo proposto, mas correndo riscos de realizar alterações indevidas que podem afetar a integridade mantida nas versões criadas.

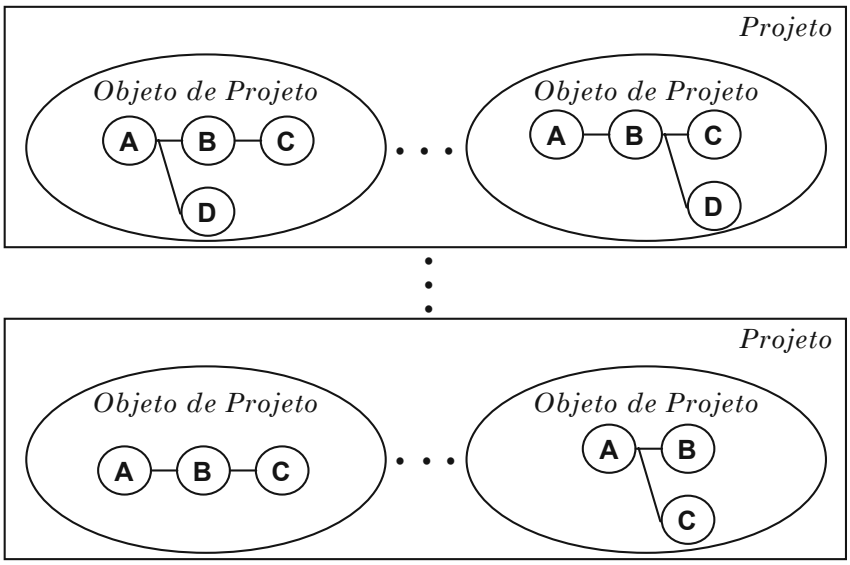


FIGURA 5.4 - Representação das Versões de Objeto de Projeto

Objetivando facilitar o versionamento de objetos de projeto e apresentar formas de manter a integridade entre as versões, o modelo de versões do *GerDoc Ábacus* engloba as características a seguir descritas, tendo a maioria delas baseada no modelo de [GOL95a]:

① Cada projeto é composto por vários objetos de projeto que, por necessidade de modificações durante seu ciclo de vida, podem ser versionados até atingirem um nível satisfatório;

② Baseado na primeira característica, um objeto de projeto é considerado um *objeto versionado*, mantendo um grafo de derivação acessível ao usuário, para que seja possível visualizar sua situação nos diversos estágios de desenvolvimento. No momento da criação de um objeto de projeto, dados referentes a ele são armazenados e, automaticamente, é criada a sua primeira versão (A), que se torna o ponto inicial do desenvolvimento do objeto, tendo anexado um documento inicial, que poderá ser versionado posteriormente;

③ Cada objeto versionado possui uma versão que é considerada *versão corrente*. O usuário pode especificar a versão corrente que, por *default*, é mantida automaticamente como a mais recentemente criada. No processo de concepção de um objeto de projeto, várias versões podem surgir e a última versão criada se tornará a versão corrente se o usuário não alterar essa propriedade. Isso significa que a versão corrente, na maioria dos casos, não é a versão final (considerada consistente) do objeto de projeto. Quando for atingido um estado satisfatório, o usuário pode indicar qual é a versão que será considerada corrente. Isso deverá ser uma tarefa usual aos engenheiros de projeto. A versão corrente será utilizada sempre que o usuário não se referir a uma versão específica de um objeto de projeto. Por exemplo, quando for criado um objeto complexo de projeto (visto posteriormente), o usuário pode compô-lo a partir de

vários outros objetos de projeto e, se não for identificada uma versão específica, o sistema irá relacionar automaticamente a versão corrente.

④ Uma versão pode ter várias antecessoras e várias sucessoras. No momento em que uma versão é criada como sucessora de uma já existente, ela passa a possuir seus mesmos dados e, automaticamente, uma cópia do projeto CAD é feita e gravada em um arquivo com nome diferente.

⑤ Cada versão de um documento possui um *status*, que reflete seu estágio de desenvolvimento, podendo ser: (1) *em trabalho*, que é uma versão temporária que deve sofrer diversas alterações; (2) *estável*, que é uma versão que já atingiu um estado mais consistente, podendo ser compartilhada e não podendo sofrer alterações ou; (3) *consolidada*, que é uma versão no seu estágio final, não podendo mais ser alterada ou removida. No momento de sua criação, uma versão recebe o *status em trabalho*. As versões utilizadas como antecessoras de novas versões passam automaticamente para *estáveis*, impedindo que alterações sejam feitas em versões-base de outras. Essas promoções automáticas não significam que o usuário não possa realizar as suas próprias promoções, explicitamente, podendo passar uma versão *em trabalho* para *estável* e uma versão *estável* para *consolidada*.

5.5 Controle do Fluxo de Tarefas do *GerDoc Ábacus*

A *seção 4.1* destacou alguns modelos de controle de fluxo de tarefas existentes na literatura. Alguns desses mecanismos apresentam formas de apoiar o projetista na seleção da alternativa a ser seguida, dependendo da situação atual do projeto. Tais mecanismos, conhecidos como *semânticos*, diferem dos *sintáticos* exatamente por essa característica.

Apesar de não possuírem apoio à tomada de decisões, os mecanismos sintáticos possuem relevante importância, pois indicam o caminho a ser seguido pelo projetista segundo as pretensões do gerente de engenharia responsável pelo projeto. É possível, portanto, controlar as ações dos engenheiros de projeto, além de indicar qual ou quais caminhos a serem seguidos.

O *GerDoc Ábacus* possui um gerenciamento de tarefas meramente *sintático*, onde é possível traçar um fluxo de tarefas específico para cada projeto, individualmente. Possivelmente, esse seja o ponto forte desse modelo: a possibilidade da criação de um fluxo de tarefas para cada projeto. O motivo dessa característica está embasado em dois fatos: (1) gerentes de engenharia podem querer controlar seus subordinados de formas diferentes, e; (2) projetos diferentes podem precisar receber tratamentos diferentes.

Conforme visto anteriormente, apenas um gerente de engenharia pode criar um novo projeto, tendo que, posteriormente, criar o grupo de usuários que irá trabalhar nele, traçando o perfil de cada um. Juntamente com isso, o gerente de engenharia responsável precisa indicar o fluxo de tarefas que será utilizado para o ciclo de vida de cada objeto de projeto. Caso não exista um fluxo de tarefas adequado, existe a possibilidade de criação de um novo.

Portanto, cada projeto terá um modelo de fluxo de tarefas associado, que será seguido pelos projetistas em cada objeto de projeto criado. Isso não significa que

seja necessário criar um modelo de fluxo de tarefas para cada projeto. É possível utilizar o mesmo para vários projetos.

A criação dos modelos de fluxos de tarefas se dá em um ambiente gráfico, denominado *editor de tarefas*, que permite criar um *modelo de tarefas*, conforme o exemplo da *figura 5.5*, indicando a tarefa inicial, sendo que é permitido ter mais que uma tarefa inicial, as tarefas intermediárias e as tarefas finais. As tarefas são representadas por um retângulo e a direção do fluxo por uma seta. Uma tarefa pode ter mais que uma sucessora no fluxo e duas ou mais tarefas podem convergir para uma única tarefa. Também é possível retornar o fluxo para pontos anteriores.

Além disso, para cada tarefa é possível determinar qual o perfil de usuário mínimo para a execução da tarefa, quais tarefas se caracterizam por serem finais e quais tarefas permitem a realização de versionamento.

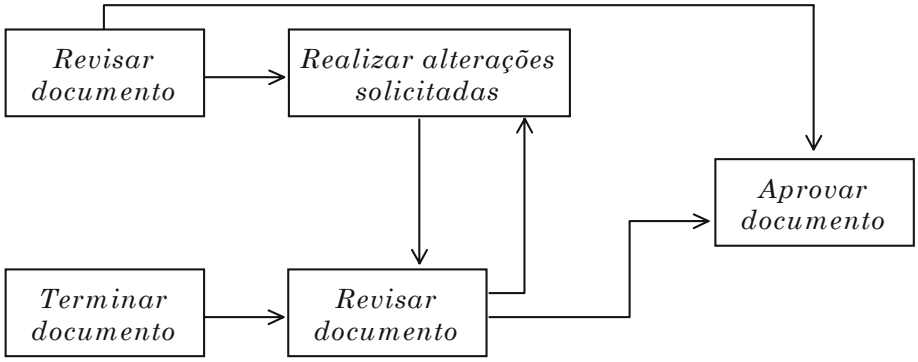


FIGURA 5.5 - Exemplo de Modelo de Fluxo de Tarefas

O exemplo da *figura 5.5* mostra um modelo com duas tarefas iniciais possíveis. O engenheiro de projeto responsável pela criação do objeto de projeto pode solicitar a revisão do documento ao gerente de engenharia responsável ou pode solicitar que outro engenheiro de projeto termine o documento. Percebe-se que, se a opção for a primeira, o responsável pela tarefa pode, após a revisão, solicitar a aprovação do documento, que se trata de uma tarefa final de aprovação, ou a realização de alterações, que, depois de executada, deve voltar a sofrer uma revisão. Se a opção for a segunda, o responsável pela tarefa deve, após terminar o documento, mandar revisar o documento e, após isso pode passar para a aprovação ou para a realização de alterações, sendo que essa tarefa permite versionamento. O fato da criação de um fluxo cíclico entre as tarefas *revisar documento* e *realizar alterações solicitadas* é comum, já que o objeto de projeto pode sofrer várias alterações e várias versões podem ser criadas até se atingir um estado satisfatório.

Todo objeto de projeto criado possuirá o seu próprio fluxo de tarefas, que parte do modelo indicado na criação do projeto. O navegador do fluxo de tarefas, apresentado na *seção 5.3.5*, é responsável por mostrar graficamente os próximos passos a serem seguidos, cabendo ao usuário definir o caminho a ser seguido no caso de várias alternativas. No momento em que uma tarefa é executada, cabe ao usuário indicar a próxima tarefa a ser executada e quem deve executá-la. A *tabela de tarefas executadas* mantém um histórico das tarefas já executadas.

5.6 Tratamento dos Objetos Complexos de Projeto

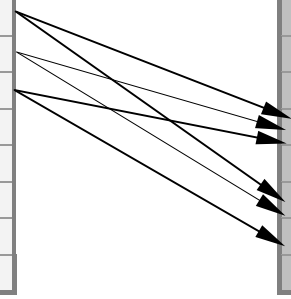
A complexidade de projetos já foi citada várias vezes como principal característica de ambientes integrados de engenharia. No caso do *GerDoc Ábacus*,

um único projeto pode ser composto por inúmeros objetos de projeto que, por sua vez, podem possuir várias versões cada. Tal fato não seria tão relevante se não fosse o fato de que grupos de projetistas interagem nesse contexto. Já foram descritas maneiras de dominar essa complexidade, como gerenciamento eficiente de usuários e fluxo de tarefas.

Existe, porém, uma outra característica de suma importância no tratamento de objetos de projeto: a existência de *objetos complexos de projeto*. As ferramentas próprias de engenharia permitem que um objeto criado possa ser utilizado na composição de outro objeto. O que se tem, então, é um objeto composto por vários outros objetos, sendo que ele pode fazer parte da composição de um terceiro objeto, e assim sucessivamente.

O que complica a situação é o fato de que um objeto de projeto pode, apesar de não ser usual, possuir objetos componentes diferentes ou adicionais, ou ainda, ser composto pelos mesmos objetos, mas de versões diferentes. Adicionalmente, um objeto pode possuir, na sua composição, objetos de outros projetos, como mostra o exemplo da *figura 5.6*. Há, portanto, a necessidade de controlar a composição de cada versão de objeto de projeto. Por isso, uma versão de objeto de projeto é considerada pelo *GerDoc Ábacus* um *objeto complexo*.

Projeto	*Objeto*	*Versão*
X	*1*	*A*
X	*1*	*B*
X	*1*	*C*
X	*2*	*A*
X	*2*	*B*
Y	*1*	*A*
Y	*1*	*B*
Y	*2*	*A*



Projeto	*Objeto*	*Versão*
X	*1*	*A*
X	*1*	*B*
X	*1*	*C*
X	*2*	*A*
X	*2*	*B*
Y	*1*	*A*
Y	*1*	*B*
Y	*2*	*A*

FIGURA 5.6 - Exemplo de composição de objetos complexos no GerDoc

A *figura 5.6* apresenta dois objetos de projeto para o projeto X e dois para o projeto Y, cada um com suas respectivas versões. As versões A, B e C do objeto 1 do projeto X são objetos complexos compostos por versões de outros objetos. Percebe-se que as versões A e B possuem a mesma composição, diferindo da versão C, que é composta pelos mesmos objetos, mas por versões diferentes destes.

O *GerDoc Ábacus* gerencia os objetos complexos de projeto de uma maneira muito simples, permitindo, para cada versão, relacionar as referências a outras versões de objetos, ou a outros objetos (nesse caso são relacionadas as versões correntes desses objetos). É possível, por exemplo, acessar os dados de um objeto componente facilmente, além de obter relações de quais os outros objetos que ele faz parte.

5.7 Controle de Cooperação no *GerDoc Ábacus*

A *seção 4.2* explicitou o requisito de suporte à cooperação entre projetistas componentes de projetos de engenharia. O gerenciamento de transações nesses ambientes deve permitir que projetistas compartilhem informações através de determinados mecanismos.

O *GerDoc Ábacus* apresenta apenas alguns dos requisitos necessários à gerência de cooperação vistos na *seção 4.2*. São eles:

① **Suporte ao trabalho em grupo:** conforme *seção 5.2*, grupos de projetistas são criados para trabalhar em cada projeto. Perfis diferentes de usuários garantem a hierarquia do trabalho cooperativo. Adicionalmente, o controle do fluxo de tarefas (*seção 5.5*) permite gerenciar de forma segura as ações realizadas pelos projetistas;

② **Suporte a dados complexos:** Os objetos complexos, apresentados na *seção 5.6*, garantem uma boa manipulação da complexidade existente em projetos de engenharia;

③ **Suporte a evolução dos dados:** O versionamento de dados (*seção 5.4)* garante a perfeita representação dos sucessivos refinamentos necessários na concepção de um projeto de engenharia.

Apesar de não se tratar de um modelo de cooperação completo, ele permite o controle da troca de dados e/ou informações entre projetistas durante a execução de suas tarefas e garante um controle uniforme do ciclo de vida de cada objeto de projeto.

6 Aplicação do Modelo Conceitual na Interface do *GerDoc Ábacus*

A interface do sistema gerenciador de documentação técnica para ambientes de engenharia/CAD *GerDoc Ábacus* é composta por dois fatores imprescindíveis para qualquer bom sistema de informática: (1) *a facilidade de operacionalização*, fazendo com que o usuário interaja rapidamente com todos os módulos do sistema, sem complicação, e; (2) *um visual agradável*, fator que complementa o primeiro, garantindo que o usuário se sinta bem na sua relação diária com o sistema.

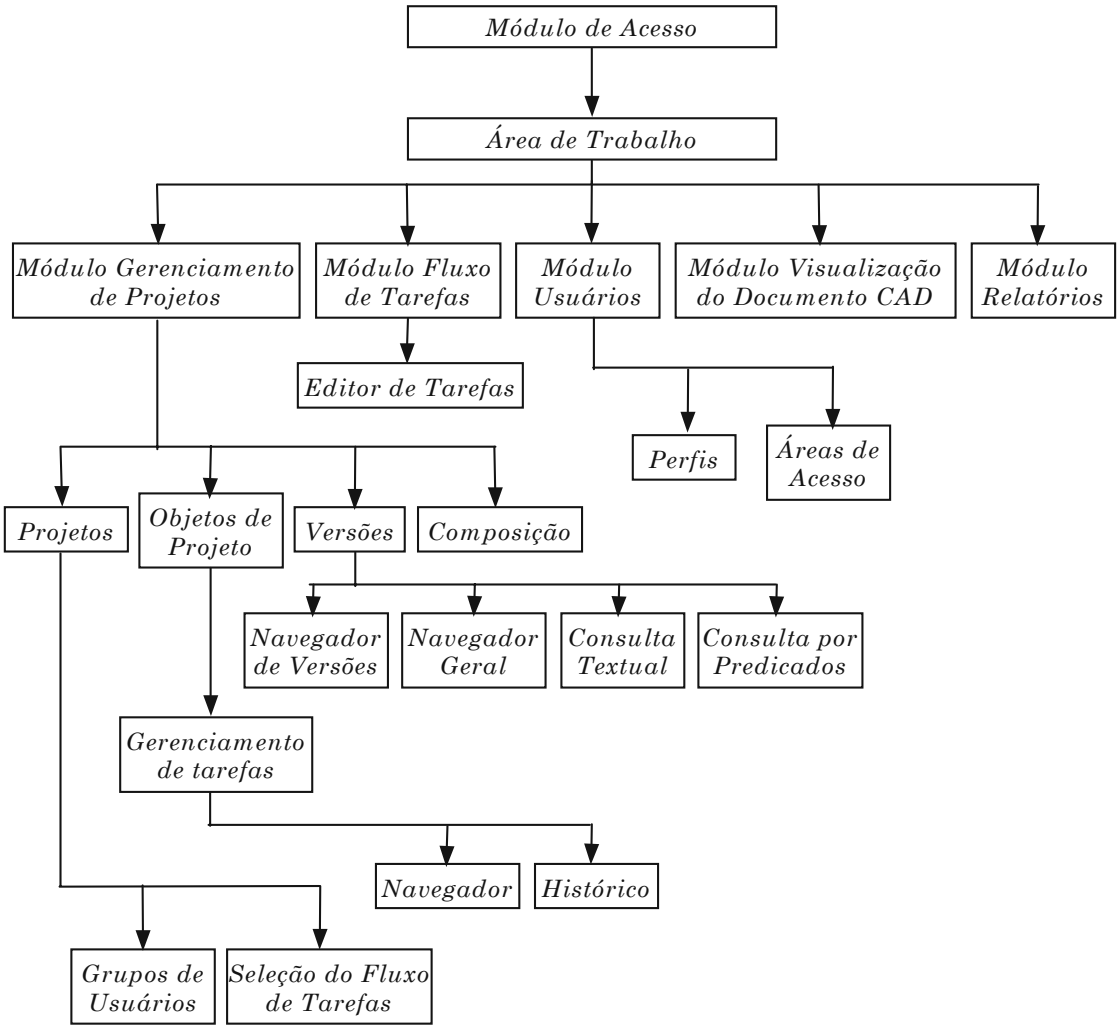


FIGURA 6.1 - Módulos Componentes do GerDoc Ábacus

A parte operacional do *GerDoc Ábacus* segue explicitamente o modelo conceitual apresentado no *capítulo 5*. Cada módulo componente possui seu funcionamento calcado nos conceitos do modelo específico que justifica sua existência. Os módulos existentes e a interação entre eles são mostrados na *figura 6.1*, e cada um deles será especificado nas seções subseqüentes.

6.1 Módulo de Apresentação e Acesso ao *GerDoc Ábacus*

Após a invocação do *GerDoc Ábacus*, é mostrada imediatamente ao usuário a tela de entrada do sistema, havendo uma iniciação de componentes fundamentais para seu funcionamento, acompanhados por uma barra de progresso, sendo disponibilizado a seguir campos para digitação do nome do usuário e sua respectiva senha, conforme mostra a *figura 6.2*.

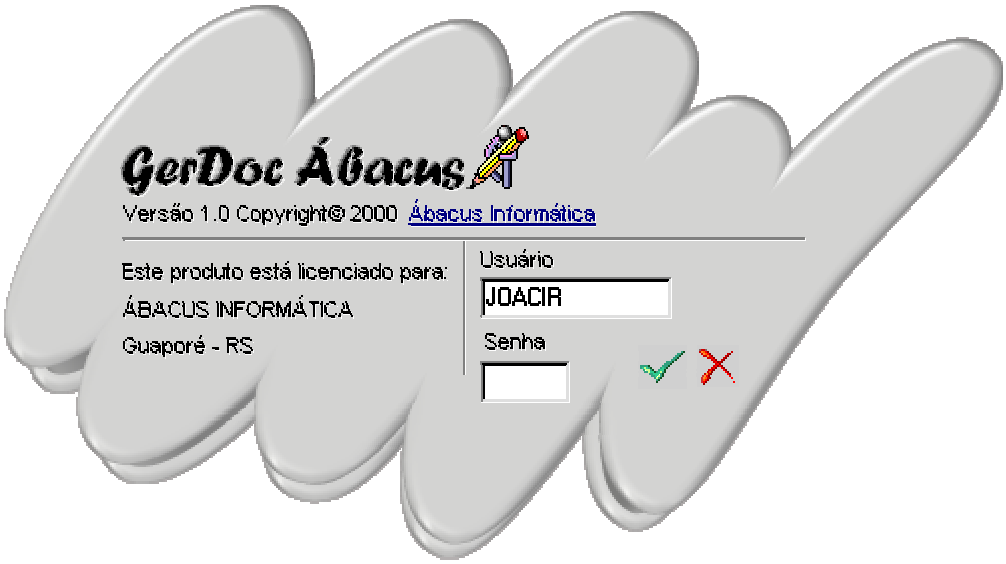


FIGURA 6.2 - Módulo de Acesso ao GerDoc Ábacus

Esse módulo controla todos os acessos ao sistema, garantindo que usuários não autorizados visualizem as informações nele contidas. Os usuários, senhas e áreas de acesso permitidas são cadastrados no módulo de usuários, descrito mais adiante. No momento que um usuário não quiser mais trabalhar no sistema, ou desejar continuar seu trabalho posteriormente, não é necessário finalizar o programa, bastando encerrar a sua sessão de trabalho, através do acesso a essa mesma tela de abertura. Assim, ele mesmo, ou qualquer outro usuário, pode abrir uma nova sessão de trabalho e continuar a operacionalização do sistema.

6.2 Área de Trabalho do *GerDoc Ábacus*

A área de trabalho do *GerDoc Ábacus* é mostrada após a validação do usuário que irá operar o sistema. A interface segue o padrão existente na maioria das aplicações construídas para o ambiente *Windows*®. Todos os módulos apresentados nesse capítulo funcionam sob essa área de trabalho, cujo *layout* é mostrado na *figura 6.3*, tendo como componentes:

① **Ícone do Sistema:** localizado no canto superior esquerdo. Ao clicar nele, surge um menu de controle da janela;

② **Identificação do Sistema:** localizada na barra de título da janela, identifica o sistema;

③ **Barra de Menu:** localizada abaixo da barra de título, possui o acesso a todas as funções do sistema. Seu funcionamento será detalhado na *seção 6.2.1*;

④ **Botões de Controle da Janela:** localizados no canto direito da barra de título, são atalhos para os controles da janela;

⑤ **Barra de Ferramentas:** localizado abaixo da barra de menu, contém botões de acesso rápido às funções do sistema. Basicamente, representam uma forma rápida de acessar os itens do menu. Seu funcionamento será detalhado na *seção 6.2.2*;

⑥ **Área de Trabalho:** espaço destinado à execução de todas os módulos do sistema;

⑦ **Barra de *Status*:** localizado na parte inferior do sistema, mostra informações gerais do sistema. Seu funcionamento será detalhado na *seção 6.2.3*;

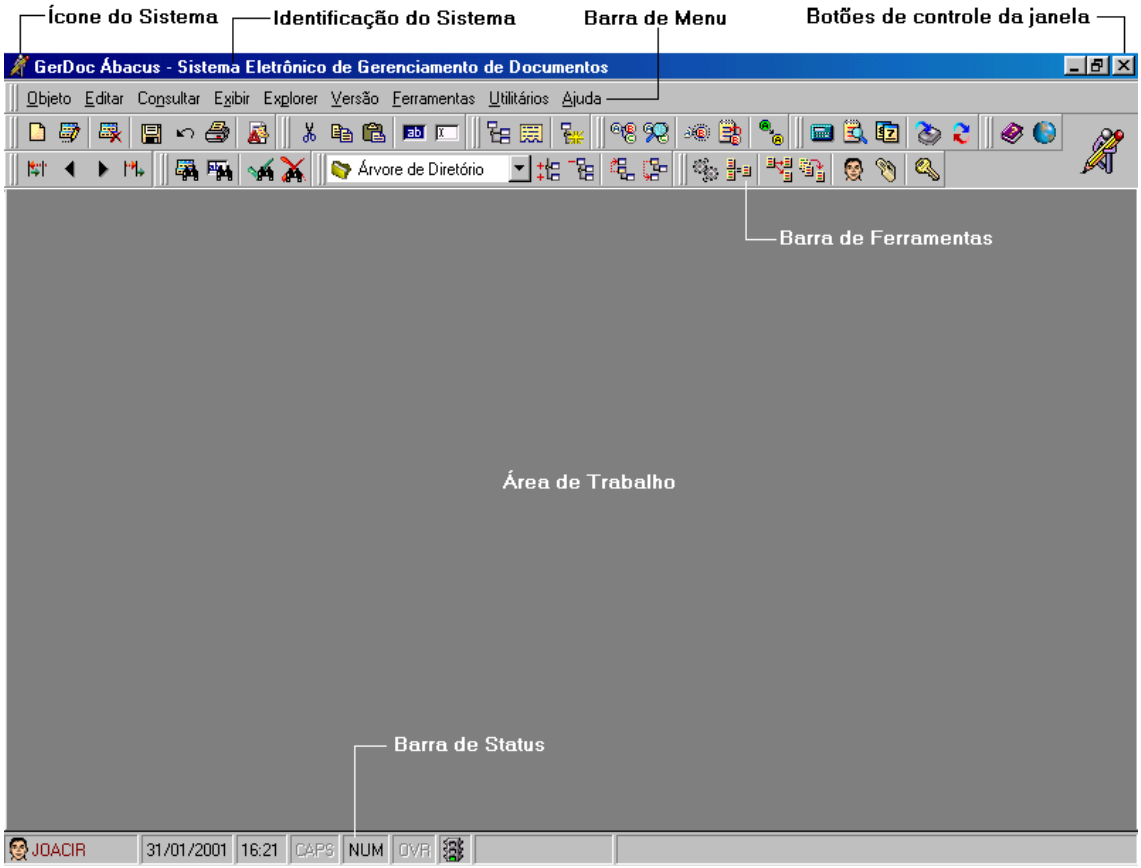


FIGURA 6.3 - Layout da Área de Trabalho do GerDoc Ábacus

6.2.1 Barra de Menu

A barra de menu do *GerDoc Ábacus* objetiva o acesso às funções do sistema e é dividida em nove itens, conforme mostra a *figura 6.4*: (1) *objeto*, contendo funções de manipulação de objetos de projeto; (2) *editar*, contendo funções de edição; (3) *consultar*, contendo funções para realização das consultas existentes no sistema; (4) *exibir*, contendo funções de manipulação do ambiente do sistema, além de alternar entre o explorer e o gerenciador de projetos; (5) *explorer*, contendo funções de navegação do explorer; (6) *versão*, contendo funções de tratamento das versões dos objetos de projeto; (7) *ferramentas*, contendo as ferramentas de suporte do sistema; (8) *utilitários*, contendo alguns acessórios úteis ao sistema ; (9) *ajuda*, contendo a ajuda do sistema.

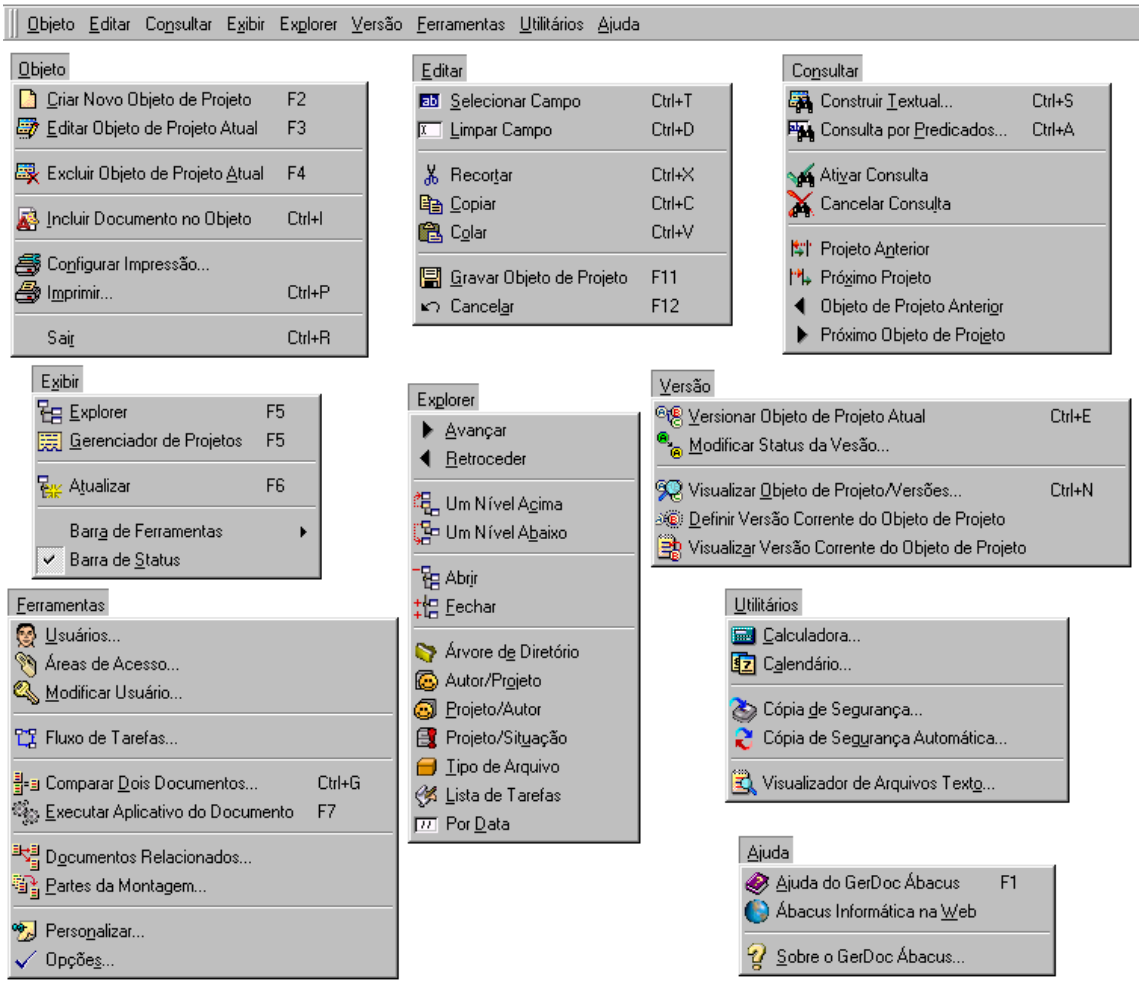


FIGURA 6.4 - Barra de Menu e seus Itens

6.2.2 Barra de Ferramentas

A barra de ferramentas do *GerDoc Ábacus* também objetiva o acesso às funções do sistema, da mesma forma que a barra de menu, porém de maneira mais rápida, bastando clicar o botão da função correspondente. A barra de ferramentas no layout da área de trabalho pode ser visualizada na *figura 6.3*.

FIGURA 6.5 - Divisões da Barra de Ferramentas

Uma característica interessante desta barra de ferramentas é a possibilidade de movimentar e/ou remover partes dela, ou ainda transforma-las em pequenas janelas, permitindo um posicionamento em qualquer parte da área de trabalho do *GerDoc Ábacus*. Dez sub-barras a compõem, conforme mostra a *figura 6.5*.

6.2.3 Barra de *Status*

A *figura 6.6* mostra a divisão existente da barra de *status* do *GerDoc Ábacus*, composta por nove painéis: (1) *usuário corrente*, identificando o usuário que estaá operando o sistema; (2) *data*, mostrando a data atual; (3) *hora*, mostrando a hora atual; (3) *Caps Lock*, identificando se a caixa alta está ativa ou não; (4) *Num Lock*, identificando se o teclado numérico está ativo ou não; (5) *Overwrite*, identificando se o modo de inserção está ativo ou não; (7) *status da versão*, mostrando graficamente o *status* da versão que está na área de trabalho, onde a cor verde é *em trabalho*, a cor amarela é *estável* e a cor vermelha é *consolidada*; (8) *barra de progresso*, apresentando uma barra com percentual para indicar o andamento de operações mais demoradas, e; (9) *mensagens*, apresentando a explicação correspondente ao local de foco do sistema.

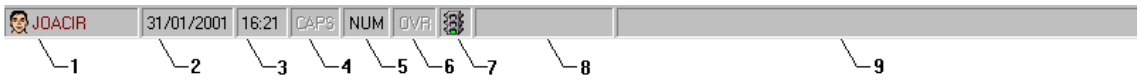


FIGURA 6.6 - Componentes da Barra de Status

6.3 Módulo de Criação dos Modelos de Fluxo de Tarefas

Essa seção objetiva apresentar a implementação do gerenciamento de tarefas sintático visto na *seção 5.5*, onde destacou-se a criação de *modelos de tarefas* específicos para diferentes projetos, criados pelos Gerentes de Engenharia, onde cada projeto criado terá um modelo de tarefas associado, que será seguido pelo grupo de usuários designado para o desenvolvimento do projeto.

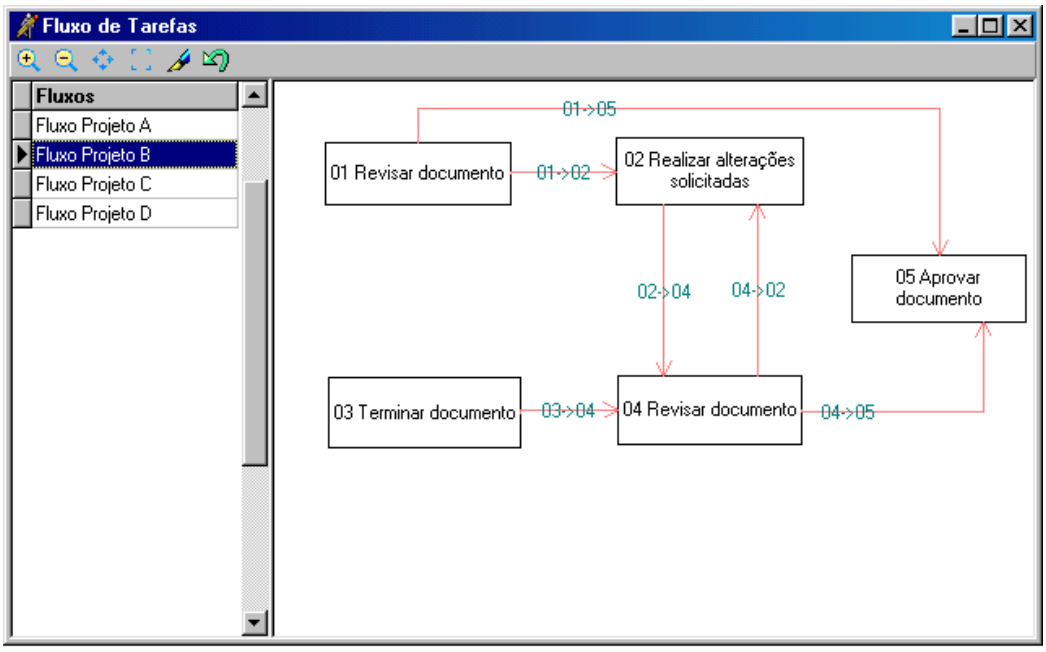


FIGURA 6.7 - Gerenciador de Modelos de Fluxos de Tarefas

Os modelos podem ser visualizados a partir do *gerenciador de modelos de fluxos de tarefas*, mostrado na *figura 6.7*, que lista os nomes atribuídos aos modelos existentes no lado esquerdo e o próprio modelo no lado direito. Uma pequena barra

de ferramentas contém algumas funções de visualização e a chamada ao *editor de tarefas*, que pode ser alcançado também a partir do duplo clique sobre o modelo.

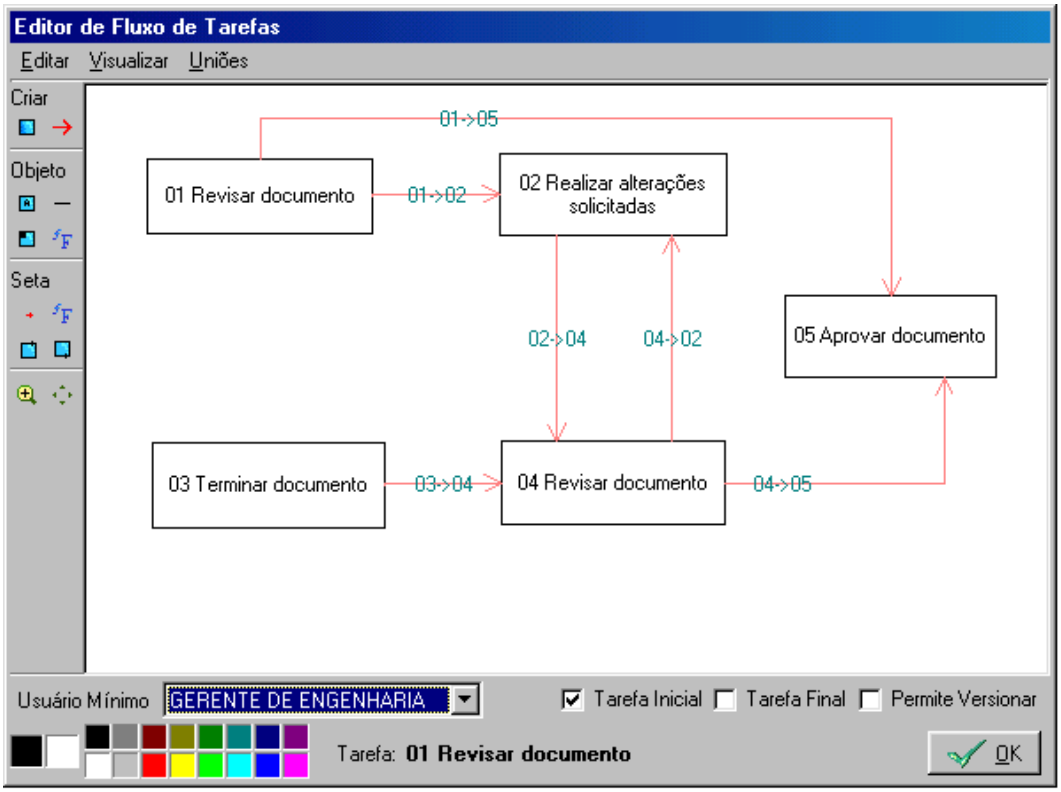


FIGURA 6.8 - Editor de Tarefas

O *editor de tarefas*, mostrado na *figura 6.8* contém diversas funções para a criação dos modelos, seguindo os conceitos da *seção 5.5*. A partir da divisão *criar* da barra de ferramentas, é possível inserir novas tarefas e novas conexões entre tarefas. No momento em que uma tarefa é criada, as propriedades da tarefa são apresentadas, conforme *figura 6.9*, onde é possível definir a aparência do retângulo que representará a tarefa, além de permitir a inserção de imagens.

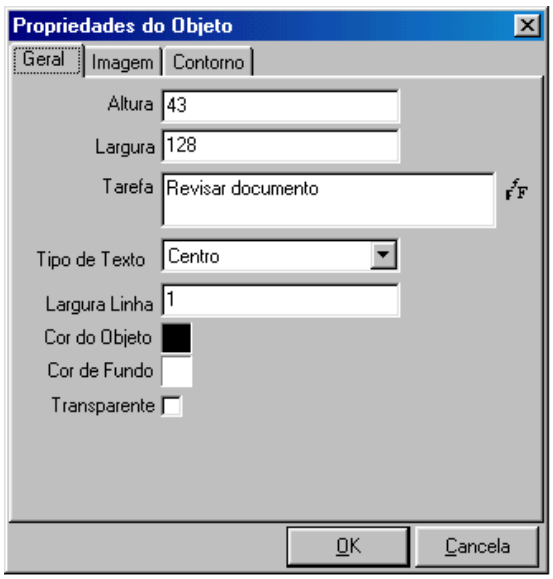


FIGURA 6.9 - Propriedades do Objeto

Cada tarefa criada deve apresentar o usuário mínimo para execução da tarefa, garantindo que determinadas tarefas não sejam executadas por usuários de hierarquia inferior. Além disso, é necessário ter, pelo menos, uma tarefa inicial e uma tarefa final, podendo existir mais do que uma de cada tipo. Se uma tarefa for designada para a criação de uma nova versão do objeto, então deve ser marcado o item *Permite Versionar*. O modelo apresentado na *figura 6.8* é o mesmo do exemplo apresentado na *seção 5.5*.

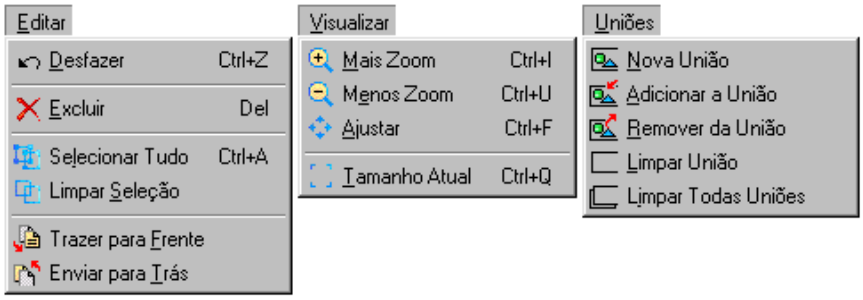


FIGURA 6.10 - Menu do Editor de Tarefas

Adicionalmente, o editor apresenta uma barra de menu (*figura 6.10*) com funções de edição e visualização, além de permitir a criação de uniões entre vários elementos criados. A barra de ferramentas também apresenta funções de manipulação dos objetos e das setas, referente a tamanho, posicionamento e cores possíveis (*figura 6.11)*.

FIGURA 6.11 - Barra de Ferramentas do Editor de Tarefas

6.4 Módulo de Usuários, seus Perfis e Áreas de Acesso

Através da criação de *usuários*, da classificação destes em *perfis de usuários* e da distribuição de *áreas de acesso* a cada perfil, é constituído o gerenciamento de usuários do *GerDoc Ábacus*, baseado na preocupação da interação necessária entre projetistas que trabalham cooperativamente em projetos distintos. A *seção 5.2* apresentou detalhes sobre todo modelo de controle de acesso existente e essa seção visa apresentar a implementação de tal modelo.

Conforme o modelo, são quatro os perfis existentes, cada um com áreas de acesso próprias: (1) *Gerente de Ambiente*; (2) *Gerente de Engenharia*; (3) *Engenheiro de Projeto*, e; (4) *Consultor*. O gerente de ambiente, responsável pela manutenção do ambiente do sistema, é o único que pode cadastrar usuários e modificar as áreas

de acesso possíveis a cada perfil de usuário. A *figura 6.12* mostra a manutenção de usuários.

FIGURA 6.12 - Manutenção de Usuários

Um usuário pode ter, inicialmente, mais que um perfil, e todos usuários cadastrados são considerados *consultores*, com direitos apenas de visualização dos dados dos projetos. Um *gerente de ambiente* pode ser também um *gerente de engenharia*, tendo, além de direitos de personalização do sistema, direitos de criação de projetos. Os usuários que são *engenheiros de projeto* de projetos específicos são cadastrados aqui apenas como *consultores*, para depois serem distribuídos em grupos de usuários próprios para cada projeto. As áreas de acesso são atribuídas aos perfis na tela mostrada pela *figura 6.13*.

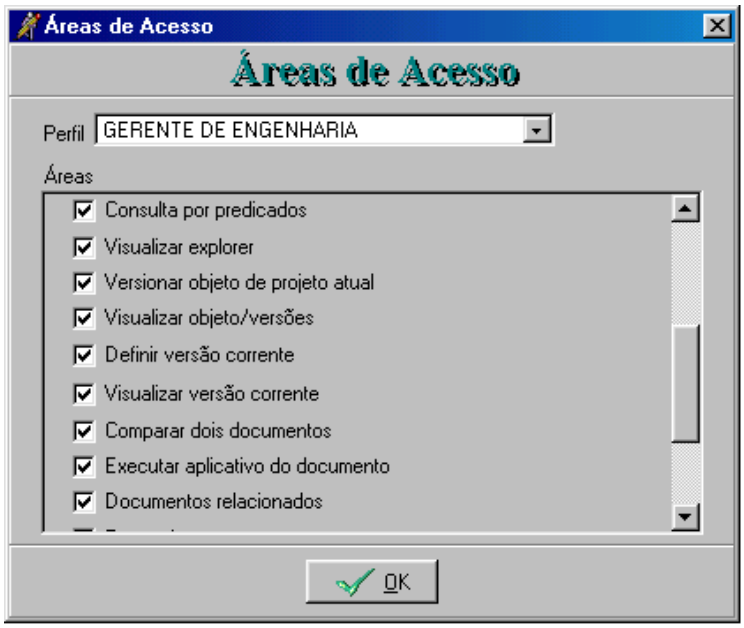


FIGURA 6.13 - Áreas de Acesso por Perfil de Usuário

No momento em que um projeto é criado, o *gerente de engenharia* responsável determinará o grupo de usuários que irá trabalhar no projeto. Só então são determinados os *engenheiros de projeto* que interagem nele. Os usuários que não

estiverem incluídos no grupo de um projeto são considerados *consultores*. A criação dos grupos de usuários será melhor especificada em seções posteriores.

6.5 Visualizador de Documentos CAD

No terceiro item da apresentação das funcionalidades previstas para o *GerDoc Ábacus* (*seção 5.1*), destacou-se a referência existente entre as versões de um objeto de projeto específico e seus documentos, que são os próprios projetos CAD desenvolvidos em programas específicos de engenharia. Existe uma importância muito grande na visualização do documento CAD, pois é a partir dele que são feitas as avaliações do seu desenvolvimento para que se possa determinar as tarefas subseqüentes a serem realizadas até se atingir um estado satisfatório.

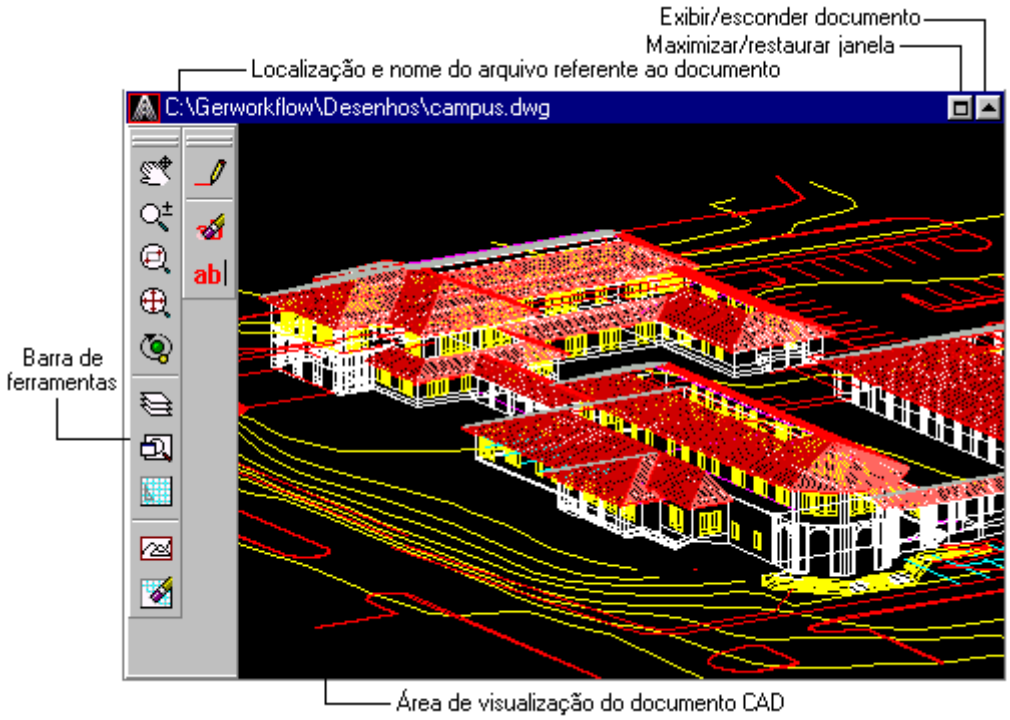


FIGURA 6.14 - Visualizador de Documentos CAD

Nesse contexto, o *GerDoc Ábacus* apresenta o *visualizador de documentos CAD*, mostrado na *figura 6.14*, que é uma ferramenta que possui várias opções de visualização e de *markup*, importantes para anotações e comentários sobre alterações a serem realizadas em versões futuras do objeto de projeto.

O *visualizador* está disposto ao lado direito do módulo de gerenciamento de projetos, explicitado em seções posteriores. O documento CAD visualizado é referente à versão do objeto de projeto que está sendo consultado. Como a área de visualização é restrita, existe a opção de maximização/restauração da janela no canto superior direito desta. A janela maximizada ocupa todo o espaço da área de trabalho do *GerDoc Ábacus*.

Como já foi explicado anteriormente, documentos CAD são os próprios projetos de engenharia, desenvolvidos em programas específicos e, portanto, armazenado em arquivos físicos. O que existe aqui é uma relação entre a descrição da versão de um objeto de projeto com um destes arquivos. Por se tratar de projetos de engenharia, o *visualizador* pode demorar para mostrar o documento,

dependendo de sua complexidade. Para tornar mais rápida a navegação entre versões e entre objetos de projeto, existe a opção de exibir/esconder o documento, garantindo que as informações referidas possam ser vistas sem ter que aguardar o carregamento do documento CAD.

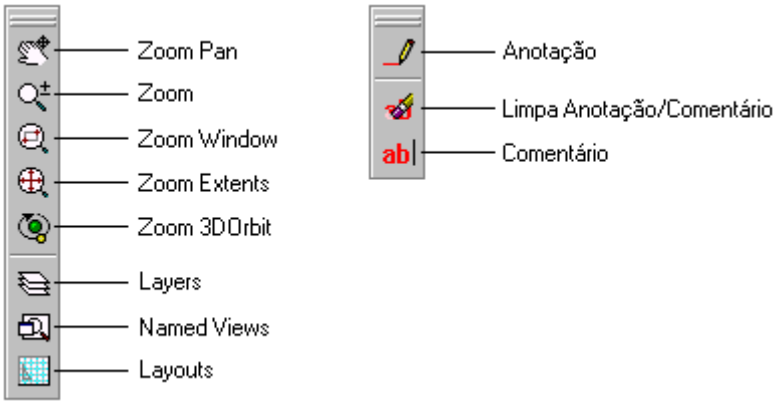


FIGURA 6.15 - Barra de Ferramentas do Visualizador de Documentos CAD

A barra de ferramentas (*figura 6.15*) apresenta as opções de visualização e *markup* já comentadas. Nas opções de visualização, é possível dar *zoom* de diversas formas, além de permitir movimentar o documento e rotacioná-lo. É possível também visualizar os *layers*, as *views* e os *layouts* do documento, opções particulares de documentos de engenharia. Nas opções de *markup*, é possível realizar anotações e comentários em qualquer parte do documento, objetivando o envio deste para alteração, provavelmente com o surgimento de uma nova versão.

6.6 Módulo de Gerenciamento de Projetos

Os conceitos vistos na *seção 5.3.1* referentes ao gerenciamento de projetos do *GerDoc Ábacus* são identificados nessa seção, objetivando visualizar a forma de operacionalização real do sistema em termos da organização de sua base de dados na forma <*projeto, objeto de projeto, versão do objeto de projeto*>, da interação dos usuários quanto aos seus direitos de acesso e da execução do fluxo de tarefas.

A *figura 6.16* mostra como o sistema se apresentará ao usuário. Observa-se na parte superior a *barra de menu* e a *barra de ferramentas*, na parte direita o *visualizador de documentos CAD* e na parte esquerda o gerenciador de projetos, dividido nas quatro partes já vistas na *seção 5.3.1*: (1) *gerenciamento de projetos*; (2) *gerenciamento de objetos de projeto*; (3) *gerenciamento das versões dos objetos de projeto*, e (4) *gerenciamento das versões de objeto de projeto componentes*.

Em primeira instância, *projetos* devem ser criados para receberem *objetos de projetos* com suas respectivas *versões*. É importante ressaltar que algumas operações possuem exclusividade de execução, como, por exemplo, a criação de novos projetos, que é tarefa exclusiva dos gerentes de engenharia. No caso de usuários que, apesar de não possuírem o perfil necessário para a realização de uma destas operações, ou qualquer outra que exija determinado perfil, tentarem realiza-las, surgirá uma mensagem alertando a irregularidade e informando o perfil que pode executar a operação.

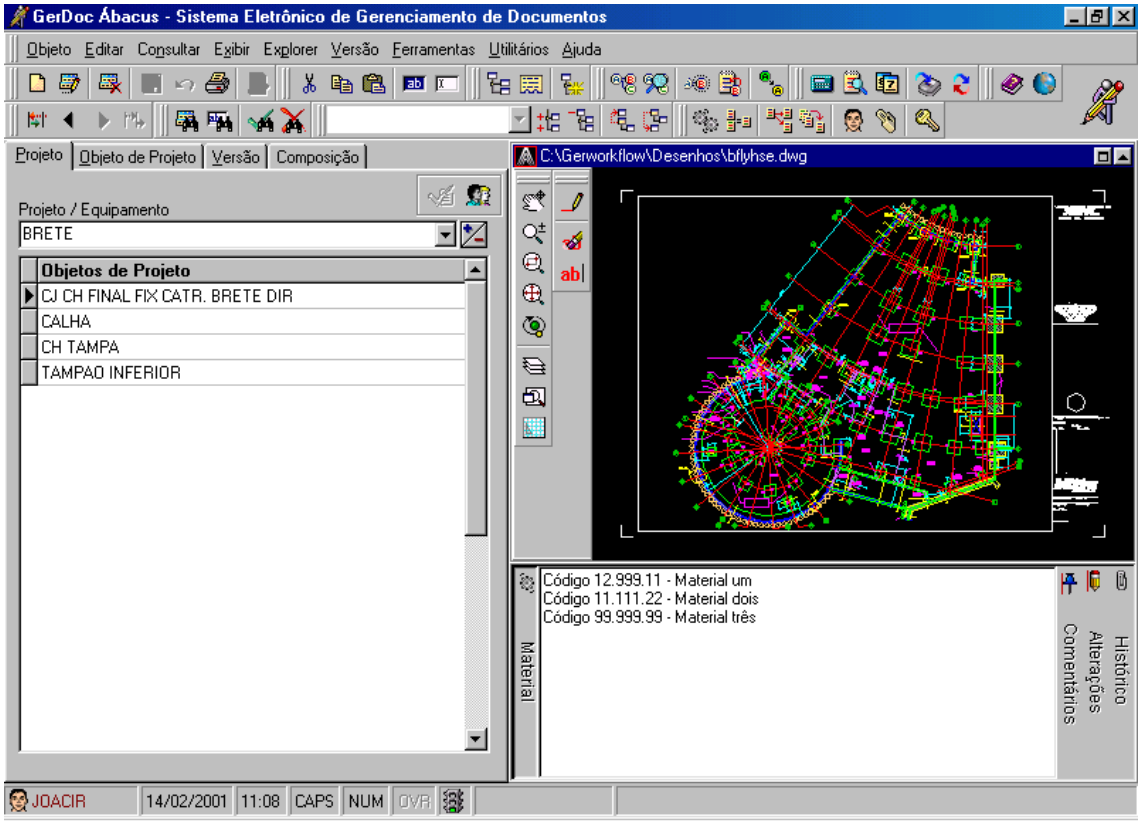


FIGURA 6.16 - Visão geral do GerDoc Ábacus

A *figura 6.17* mostra o *gerenciador de projetos*, onde é possível, além de criar novos projetos, visualizar os existentes, juntamente com os respectivos objetos de projeto que o compõem. Ao selecionar o objeto de projeto desejado, o usuário pode visualizar seus dados e suas versões clicando nas abas correspondentes. A inclusão de novos objetos de projetos também passa pelo gerenciador de projetos, já que o primeiro passo é selecionar o projeto em que o novo objeto será inserido. Quando o projeto for selecionado, basta clicar no botão de confirmação de projeto. Nesse instante, um novo objeto de projeto é criado, e o controle da janela passa para a aba correspondente, mostrado na *figura 6.19*.

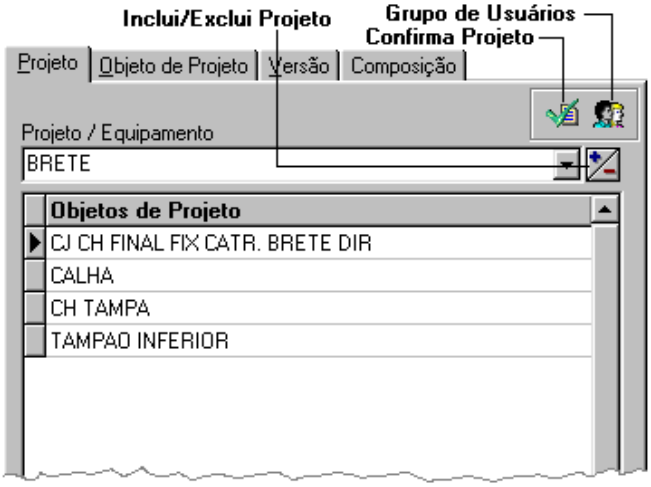


FIGURA 6.17 - Gerenciador de Projetos

A criação de novos projetos é uma tarefa simples, já que basta digitar o nome do projeto e clicar no botão de inclusão de projetos. Porém, uma operação importante está ligada à criação de um projeto: a montagem do grupo de usuários que interagirá com o projeto e a determinação do modelo de fluxo de tarefas que será seguido em cada objeto de projeto a ser criado posteriormente.

FIGURA 6.18 - Criação do grupo de usuários de um projeto específico

Observa-se na *figura 6.18* que o gerente de engenharia criador do projeto passa a ser o responsável pelo projeto, sendo o único que pode alterar a definição do grupo de usuários. Um usuário que, no seu cadastro inicial, fora definido como gerente de engenharia, por exemplo, pode ser redefinido em um projeto como engenheiro de projeto. Os usuários do sistema que não forem incluídos em um grupo de usuários assumem o perfil de consultores para aquele projeto.

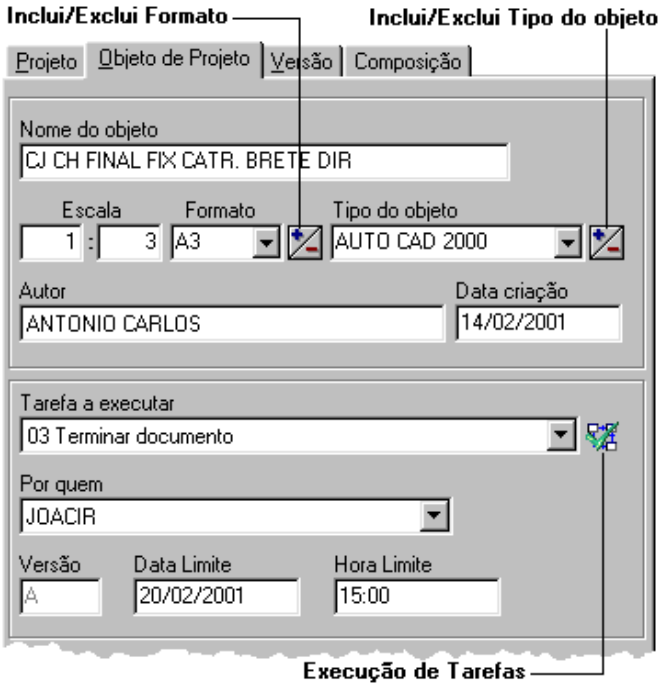


FIGURA 6.19 - Gerenciador de Objetos de Projeto

O modelo do fluxo de tarefas a ser seguido é determinado na criação do projeto, para não ser mais alterado. No momento que o primeiro objeto de projeto for criado, não é mais permitido trocar o modelo.

Os objetos de projeto são criados a partir do gerenciador de objetos de projeto, mostrado na *figura 6.19*. Conforme já visto na *seção 5.3.1*, no instante que um objeto de projeto é criado, sua versão inicial também é criada. É necessário, portanto, que o *objeto versionado* seja cadastrado juntamente com a sua primeira versão, que possuirá o documento CAD associado. Não é possível cadastrar um objeto de projeto sem ter uma versão associada. Caso isso aconteça, o sistema alertará sobre a irregularidade. A *figura 6.23* mostra o gerenciador de versões de objeto de projeto.

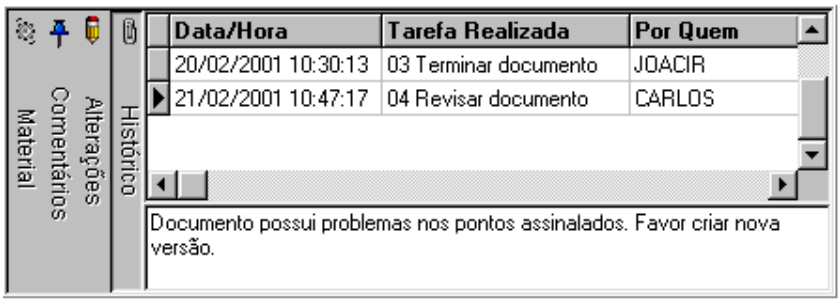


FIGURA 6.20 - Histórico das tarefas realizadas

Nos dois gerenciadores, percebe-se a existência de botões de inclusão e exclusão dos itens *formato* e *tipo de objeto* no gerenciador de objetos de projeto e *tipo do arquivo* e *extensão* no gerenciador de versões de objeto de projeto. Tais controles servem para incluir ou retirar da lista de valores possíveis para o item em questão, não interferindo no objeto ou na versão que estão sendo cadastrados.

O mecanismo sintático de gerenciamento de tarefas é mantido na mesma aba do gerenciador de objetos de projeto, pois cada objeto criado terá o seu próprio fluxo de tarefas particular, baseado no modelo de fluxo de tarefas determinado na criação do projeto (*figura 6.18*), não interessando o número de versões existentes.

FIGURA 6.21 - Execução de tarefas

A tarefa a ser executada é visualizada, indicando o usuário responsável pela execução, a versão-alvo, a data e a hora limite da operação. O histórico das tarefas

já realizadas é mantida na *tabela de tarefas executadas*, mostrada na *figura 6.20*, onde, a cada tarefa executada, ocorre uma atualização.

A realização das tarefas e a determinação da próxima tarefa a ser executada, são dadas a partir do botão existente ao lado do item *tarefa a executar* (*figura 6.19*). Nesse instante, a tela de execução de tarefas é mostrada, conforme *figura 6.21*, onde o usuário responsável pela tarefa pode incluir observações sobre a o que aconteceu e determinar a próxima tarefa a ser executada para algum outro usuário, traçando os limites de execução.

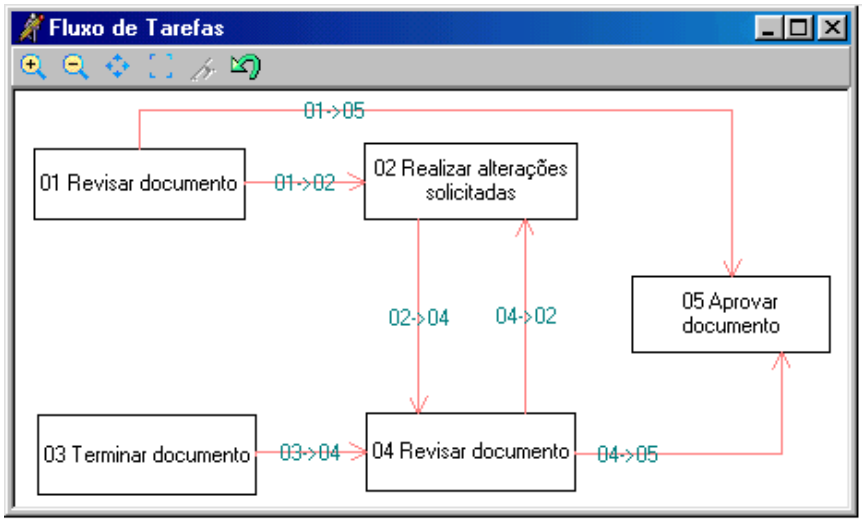


FIGURA 6.22 - Navegador do fluxo de tarefas

Caso o usuário queira visualizar todo o fluxo de tarefas, é possível acessar o *navegador do fluxo de tarefas*, mostrado na *figura 6.22*, que mostra graficamente os próximos passos possíveis, possibilitando a escolha do caminho a ser seguido, no caso de várias alternativas.

A *figura 6.23* mostra o *gerenciador das versões dos objetos de projeto*, responsável pelo controle das versões de cada objeto de projeto criado. Como cada objeto de projeto é considerado um objeto versionado, no instante de sua criação, automaticamente é criada a sua primeira versão, que terá associado o documento CAD correspondente.

As características do gerenciamento de versões apresentadas na *seção 5.4* são seguidas rigorosamente e todas as ações realizadas são encontradas em itens do menu ou na própria barra de ferramentas:

① A *versão corrente* é a mais recentemente criada, a não ser que o usuário altere essa propriedade. Através de opção própria na barra de ferramentas, é possível visualizar a versão corrente de maneira rápida;

② Uma cópia do documento CAD é feita da versão-pai para a que está sendo criada e um outro nome é dado a ele automaticamente, realizando um gerenciamento automático dos documentos CAD;

③ O usuário-autor, a data e a hora de criação da versão são preenchidos automaticamente;

④ O *status* da versão-pai é promovido automaticamente para *estável*, enquanto que a nova versão recebe o *status em trabalho*;

⑤ Apenas as versões *em trabalho* podem ser alteradas. As versões *estáveis* podem ser excluídas e as versões *consolidadas* não podem ser alteradas nem excluídas;

⑥ O usuário pode, a qualquer momento, modificar o *status* da versão, passando, por exemplo, uma versão estável para consolidada.

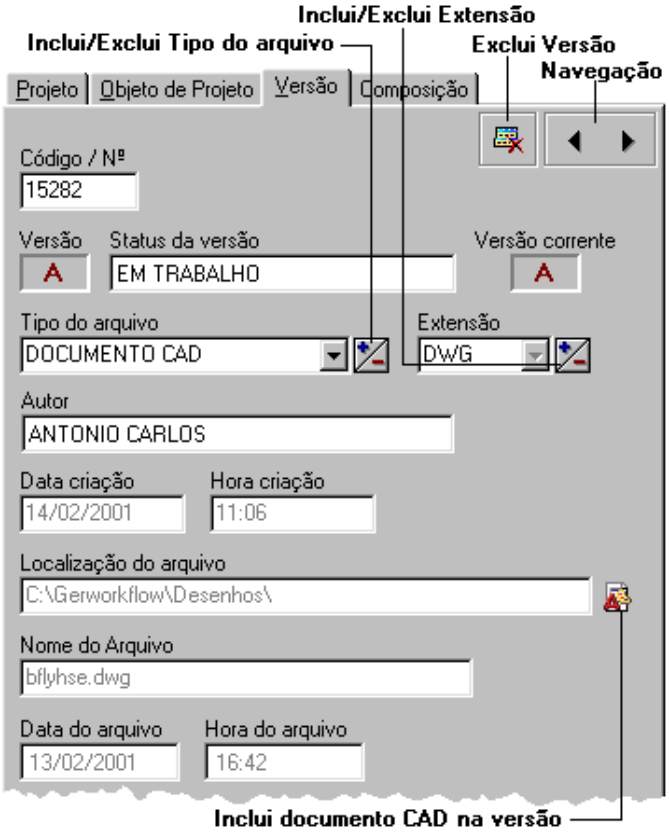


FIGURA 6.23 - Gerenciador de Versões de Objetos de Projeto

Botões dispostos na parte superior do gerenciador permitem realizar as tarefas de navegação entre as versões do objeto de projeto atual e exclusão da versão atual, dependendo de seu *status*.

O grafo de versões do objeto de projeto que está sendo trabalhado pode ser visualizado pelo *navegador de versões*, conforme *figura 6.24*. Graficamente, o *GerDoc Ábacus* permite percorrer o grafo e visualizar os principais dados das versões, além do próprio documento CAD.

Conforme já visto na *seção 5.3.4*, o navegador indica a versão corrente através de traços na parte inferior e superior do círculo e a versão que está sendo consultada através de dois círculos. O *status* das versões é identificado por cores, onde o verde representa as versões *em trabalho*, o amarelo representa as versões *estáveis* e o vermelho representa as versões *consolidadas*.

O exemplo da *figura 6.24* mostra a evolução de um objeto de projeto com quatro versões, sendo as versões *A* e *B* estáveis, a versão *C* em trabalho e a versão *D* consolidada. A versão que está sendo consultada é a versão *C*, enquanto que a versão corrente é a versão *D*.

Através de um duplo clique do mouse na versão desejada, é possível passar o controle para o gerenciador de versões de objetos de projeto, a fins de alterações ou consultas mais detalhadas.

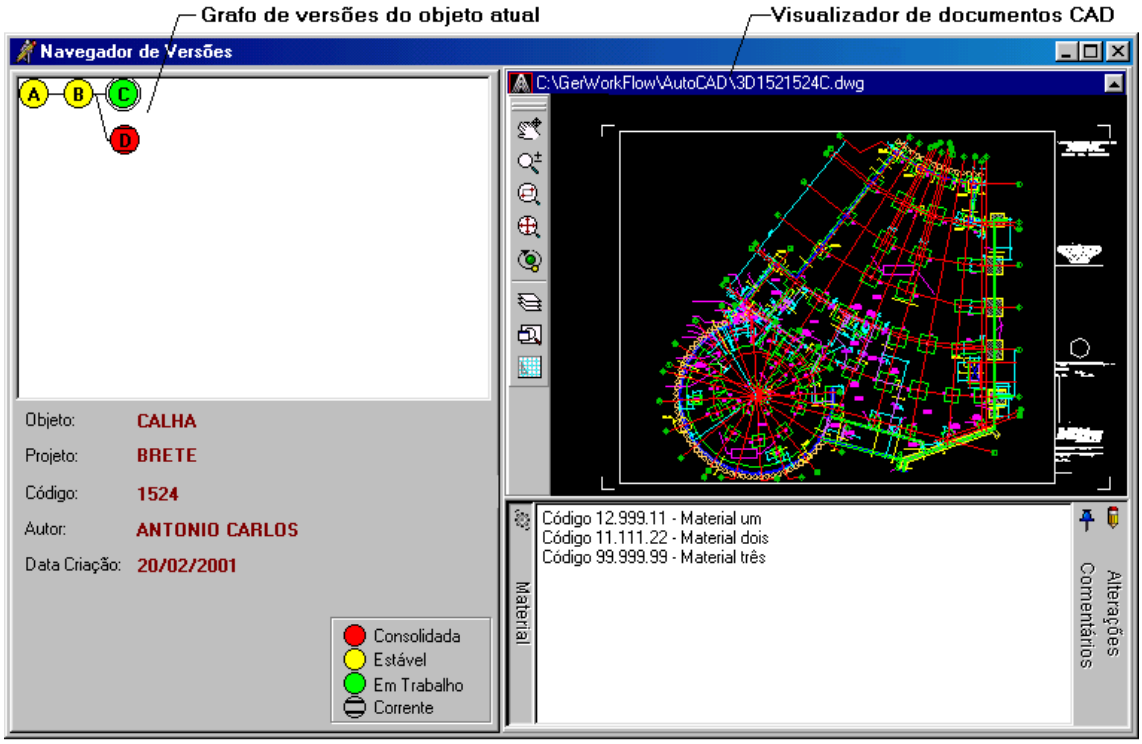


FIGURA 6.24 - Navegador de Versões

O tratamento de *objetos complexos de projeto* se dá, no *GerDoc Ábacus*, de maneira simples, conforme mostra a *figura 6.25*. O usuário cria a lista de composição das versões a partir do código da versão ou a partir de dois mecanismos de consulta (*figura 6.26*): (1) *consulta de objetos*, onde as versões correntes de cada objeto são listadas, permitindo que se faça uma consulta geral ou de apenas um projeto específico, e; (2) *consulta de versões*, onde todas as versões de cada objeto de projeto são listadas, permitindo a consulta geral ou de apenas um objeto de projeto específico.

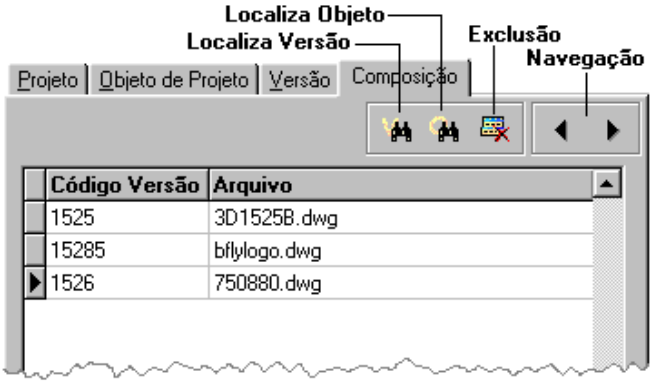


FIGURA 6.25 - Gerenciador das Versões de Objeto de Projeto Componentes

O fato de versões de objetos de projeto de um projeto específico poderem ser compostos por objetos de projeto de outros projetos, visto na *seção 5.6*, não figura

um problema, já que todos os projetos podem ser consultados e qualquer versão pode ser inserida na composição.

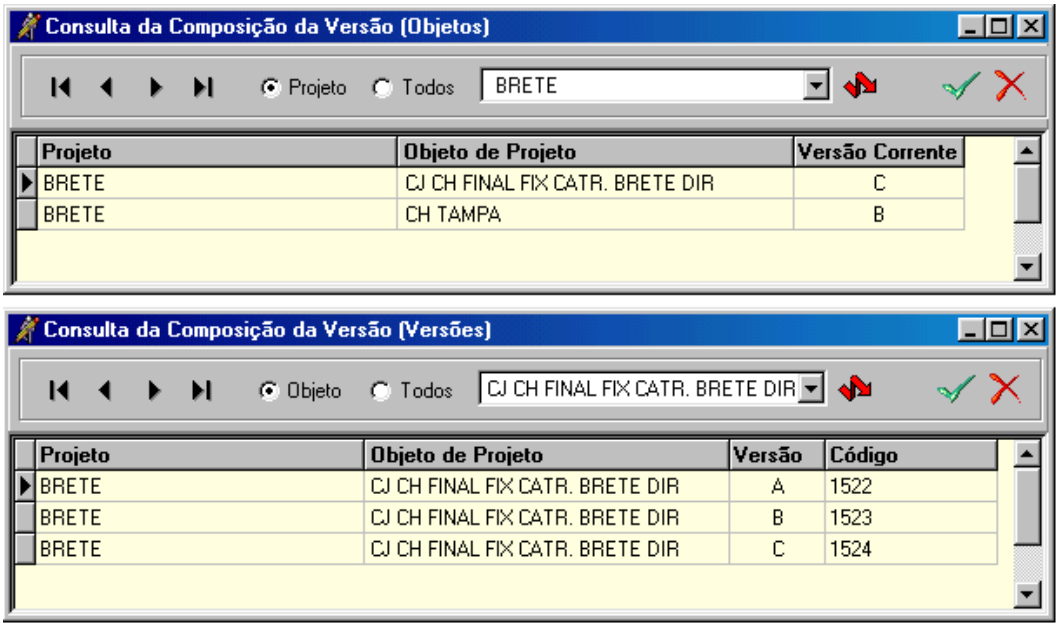


FIGURA 6.26 - Consultas da Composição da Versão

Como versões de um mesmo objeto de projeto podem possuir a mesma composição, a cada nova versão criada o *GerDoc Ábacus* questiona o usuário sobre a necessidade de armazenar a mesma composição da versão-pai. Em caso afirmativo, o sistema cria a mesma composição para a nova versão.

6.7 Módulos de Navegação e Consulta

A interface gráfica apresentada pelo *GerDoc Ábacus* permite que vários mecanismos de navegação e consulta estejam acessíveis aos usuários-projetistas durante a execução de seus trabalhos, objetivando facilitar a interação entre os grupos de projetistas e os diversos projetos existentes.

O *GerDoc Ábacus* possui uma preocupação especial nesse sentido, pois determinadas situações exigem tratamento especial dos dados de cada projeto. Por exemplo, usuários-projetistas precisam saber, diariamente e de forma rápida, quais tarefas são destinadas a ele. O *navegador geral* realiza essa operação, mostrando, de forma gráfica, a lista de tarefas correspondente.

Em outro exemplo, um gerente de engenharia responsável por um projeto pode querer saber quais foram os objetos de projeto que determinado usuário trabalhou em um determinado período de tempo. Através da *consulta por predicados* isso é possível.

Os navegadores gráficos de versões e de fluxo de tarefas já foram vistos na seção anterior, bastando explicitar o *navegador geral* e as consultas *textual* e *por predicados*. Os modelos de cada mecanismo foram descritos na *seção 5.3*. Suas implementações e interfaces são apresentadas nas seções que seguem.

6.7.1 Consulta Textual

A *consulta textual* representa a forma mais rápida e mais fácil de consulta do *GerDoc Ábacus*. A partir dela, o usuário pode realizar filtragens baseadas em ordenações fixas e intervalos pré-determinados. A *tabela 5.3* mostra as ordenações possíveis, o intervalo para cada ordenação e o resultado alcançado em cada consulta.

FIGURA 6.27 - Consulta Textual

A *figura 6.27* apresenta a interface da consulta textual, exemplificando uma consulta por data de criação em um intervalo de 14/02/2001 a 20/02/2001. O resultado, mostrado em um *grid*, contém as versões existentes nesse intervalo de código. O usuário pode selecionar uma versão e, através do duplo clique ou pelo botão OK, pode passar o controle para o gerenciador de versões.

6.7.2 Consulta por Predicados

Visando possibilitar consultas mais complexas, baseadas em expressões construídas pelo próprio usuário, a *consulta por predicados* segue os conceitos vistos na *seção 5.3.3*, e apresenta uma interface gráfica mostrada na *figura 6.28*.

Os três elementos já discutidos aparecem claramente na consulta por predicados: (1) os *atributos*, situados na parte esquerda; (2) os *operadores*, situados em botões, e; (3) os *valores referenciais*, situados no canto superior direito. O funcionamento da consulta segue alguns passos:

① O usuário seleciona o atributo através de um clique no campo desejado;

② Ao selecionar o *atributo*, automaticamente, o botão que contém o *operador* padrão fica ativo e o campo destinado ao *valor referencial* é alterado. Por exemplo, se o usuário selecionar o atributo *projeto*, o operador padrão será o *igual,* e o campo do valor referencial se transformará em uma lista de seleção dos projetos existentes. Em outro exemplo, se o usuário selecionar o atributo *data de criação*, o operador padrão será o *intervalo*, e dois campos de valores referenciais aparecem, o de data inicial e de data final;

③ Para cada atributo, há operadores que não podem ser aplicados, ficando desabilitados. O usuário pode alterar o operador a qualquer instante. Nesse caso, o campo relativo ao valor referencial é alterado, conforme o operador. Seguindo o mesmo exemplo anterior, se o usuário não quiser o operador *intervalo* para o atributo *data de criação*, pode mudar o

operador para *maior*, *menor*, *maior ou igual*, *menor ou igual*, *igual* ou *diferente*;

④ Tendo selecionado o atributo e digitado o valor referencial, o usuário deve inserir a expressão a partir do botão *incluir*;

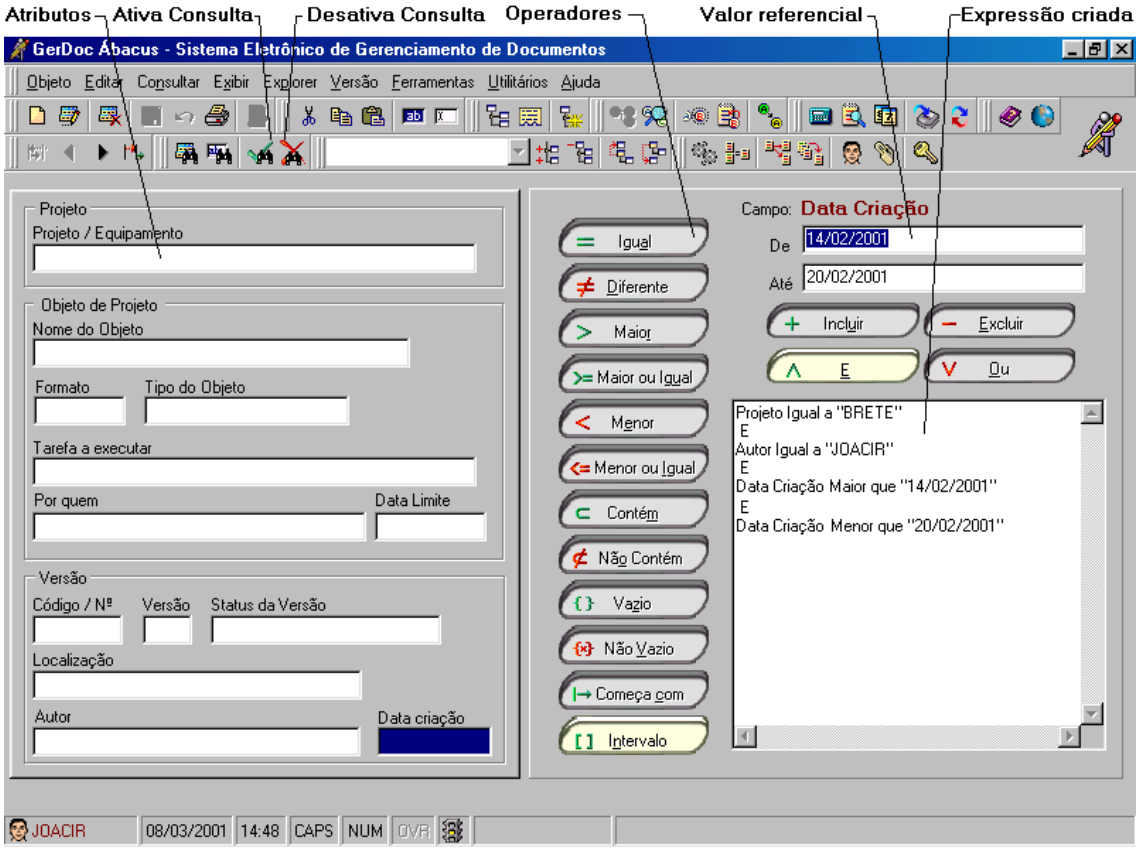


FIGURA 6.28 - Consulta por Predicados

⑤ Caso exista mais alguma expressão, o usuário deve retornar ao primeiro passo, sendo que, ao inserir a segunda expressão, poderá selecionar o operador lógico *E* ou *OU* para ligar as duas expressões;

⑥ Em qualquer momento, o usuário pode excluir expressões a partir do botão *excluir*.

⑦ Ao concluir a expressão, o usuário deve ativar a consulta a partir do botão de ativação. Nesse instante o resultado da consulta é mostrado em uma tela (*figura 6.29*), indicando o número de registros encontrados e permitindo selecionar o modo de exibição dos registros no *Navegador Geral*, que será explicitado na próxima seção.

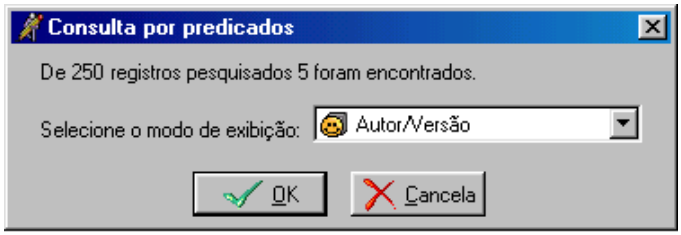


FIGURA 6.29 - Apresentação do resultado da consulta por predicados

O resultado da consulta mostrada no exemplo da *figura 6.28* mostrará, no navegador geral, todas as versões de objeto de projeto pertencentes ao projeto *BRETE*, cujo projetista-autor é o *JOACIR*, com data de criação no intervalo de *14/02/2001* a *20/02/2001*. Essa consulta pode ser feita, por exemplo, pelo gerente de engenharia responsável pelo projeto, para saber o andamento do trabalho do engenheiro de projeto em questão.

6.7.3 Navegador Geral

Através de *modos de exibição* diferentes, o *navegador geral* do *GerDoc Ábacus* apresenta os dados dos projetos no estilo *explorer*, permitindo abrir e fechar itens título de conjuntos de dados, de uma maneira gráfico-interativa. Conforme mostra a *figura 6.30*, o usuário pode selecionar o modo de exibição desejado através de lista própria.

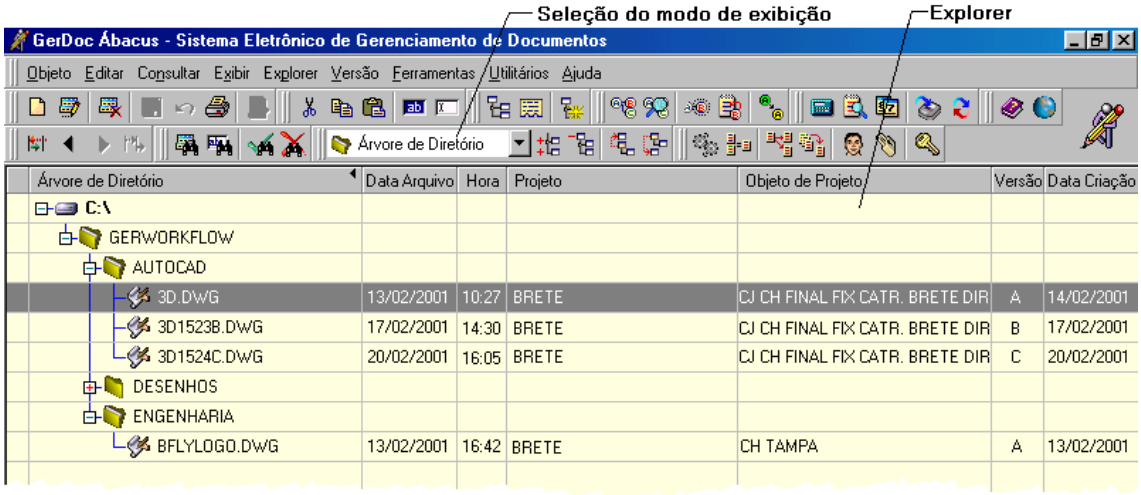


FIGURA 6.30 - Navegador Geral – Árvore de Diretórios

Não é possível realizar filtragem de dados no navegador geral, pois não se trata de uma consulta. O usuário pode localizar o item desejado e enviar o controle para o gerenciador de versões de objetos de projeto através do duplo clique sobre o item.

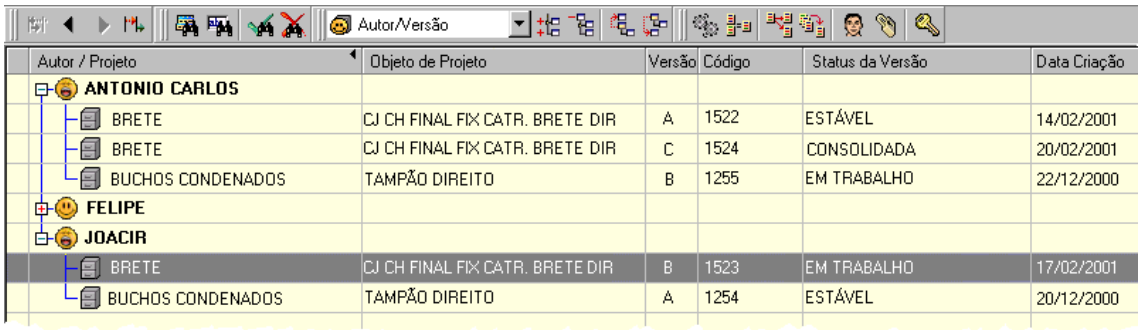
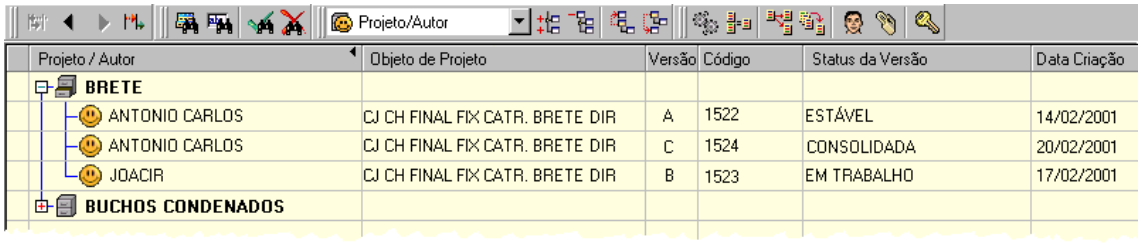


FIGURA 6.31 - Navegador Geral – Autor/Versão

A *tabela 5.2* apresenta os modos de exibição existentes e suas classificações particulares. São seis os modos passíveis de escolha:

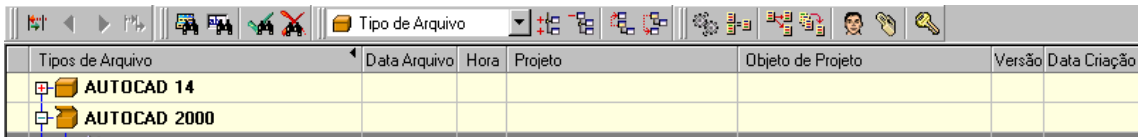
① **Árvore de Diretórios** (*figura 6.30*): Todos os diretórios que possuírem documentos CAD associados a versões de objetos de projetos são mostrados nesse modo, ordenando-os pela unidade de disco, pelo nome do diretório e pelo nome do arquivo. Para cada arquivo, são mostradas as informações: a data e hora do arquivo, o projeto, objeto de projeto e versão que ele está associado e a data de criação da versão que contém o documento CAD;



Projeto / Autor	Objeto de Projeto	Versão	Código	Status da Versão	Data Criação
BRETE					
ANTONIO CARLOS	CJ CH FINAL FIX CATR. BRETE DIR	A	1522	ESTÁVEL	14/02/2001
ANTONIO CARLOS	CJ CH FINAL FIX CATR. BRETE DIR	C	1524	CONSOLIDADA	20/02/2001
JOACIR	CJ CH FINAL FIX CATR. BRETE DIR	B	1523	EM TRABALHO	17/02/2001
BUCHOS CONDENADOS					

FIGURA 6.32 - Navegador Geral – Projeto/Autor

② **Autor/Versão** (*figura 6.31*): Engenheiros de projeto podem querer visualizar as versões em que foram responsáveis pela criação, ou ainda, gerentes de engenharia podem querer obter o desempenho dos engenheiros de projeto em um projeto específico. Através do modo *autor/versão*, as versões dos objetos de projeto aparecem ordenadas por autor e por projeto, mostrando as informações: objeto de projeto, versão, código da versão, *status* da versão e sua data de criação;



Tipos de Arquivo	Data Arquivo	Hora	Projeto	Objeto de Projeto	Versão	Data Criação
AUTOCAD 14						
AUTOCAD 2000						
3D.DWG	13/02/2001	10:27	BRETE	CJ CH FINAL FIX CATR. BRETE DIR	A	14/02/2001
3D1523B.DWG	17/02/2001	14:30	BRETE	CJ CH FINAL FIX CATR. BRETE DIR	B	17/02/2001
3D1524C.DWG	20/02/2001	16:05	BRETE	CJ CH FINAL FIX CATR. BRETE DIR	C	20/02/2001
MECHANICAL DESKTOP						
BFLYLOGO.DWG	13/02/2001	16:42	BRETE	CH TAMPA	A	13/02/2001

FIGURA 6.33 - Navegador Geral – Tipo de Arquivo

③ **Projeto/Autor** (*figura 6.32*): Através desse modo, um gerente de engenharia pode visualizar a situação do projeto que é responsável, em relação aos engenheiros de projeto que fazem parte do grupo de usuários do projeto e as versões criadas. A ordenação é feita por projeto e por autor do projeto, mostrando as informações: objeto de projeto, versão, código da versão, seu *status* e sua data de criação;

④ **Tipo de Arquivo** (*figura 6.33*): São diversas as ferramentas utilizadas para o projeto de documentos CAD. Além disso, cada ferramenta pode ter várias versões, e isso faz com que os documentos CAD tenham que ser divididos por tipo de arquivo. Através do modo *tipo de arquivo* é possível listar os documentos CAD associados a versões de objetos de projeto ordenados por tipo de arquivo e pelo nome do arquivo, seguindo as

informações: data e hora do arquivo, projeto, objeto de projeto e versão que o documento CAD está associado e a data de criação da versão;

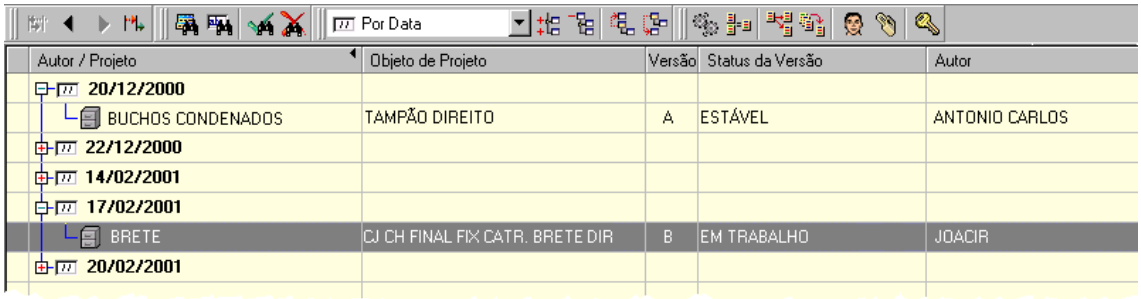


FIGURA 6.34 - Navegador Geral –Por Data

⑤ **Por Data** (*figura 6.34*): Através desse modo, o usuário pode visualizar todas as versões criadas em determinada data. Ordenada por data e por projeto, mostra as informações: objeto de projeto, versão, *status* e autor da versão;

FIGURA 6.35 - Navegador Geral – Lista de Tarefas

⑥ **Lista de Tarefas** (*figura 6.35*): Provavelmente esse será o modo mais utilizado pelos usuários, pois através dele será possível visualizar, de forma rápida, as tarefas a serem executadas em determinada data. Possuir ordenação pelo responsável pela tarefa e pela data e hora limite de execução, mostrando, além dessas informações, o projeto, objeto de projeto e versão que deve ser executada a tarefa.

6.8 Módulo de Relatórios

O *GerDoc Ábacus* não se caracteriza por ser um sistema que necessite de relatórios diversos, já que se trata de um gerenciador de documentos técnicos que objetiva organizar o trabalho dos projetistas de ambientes de engenharia. Os próprios mecanismos de navegação e consulta já são providos de vários modos de mostragem de dados de projetos, o que diminui significativamente a necessidade de imprimi-los.

Mesmo assim, o *GerDoc Ábacus* apresenta três tipos de relatórios distintos, com ordenações próprias e definição de intervalos de valores. A *figura 6.36* apresenta a tela do *filtro de relatórios*, onde aparece, além dos tipos e intervalos, a definição da saída do relatório e o número de cópias desejado.

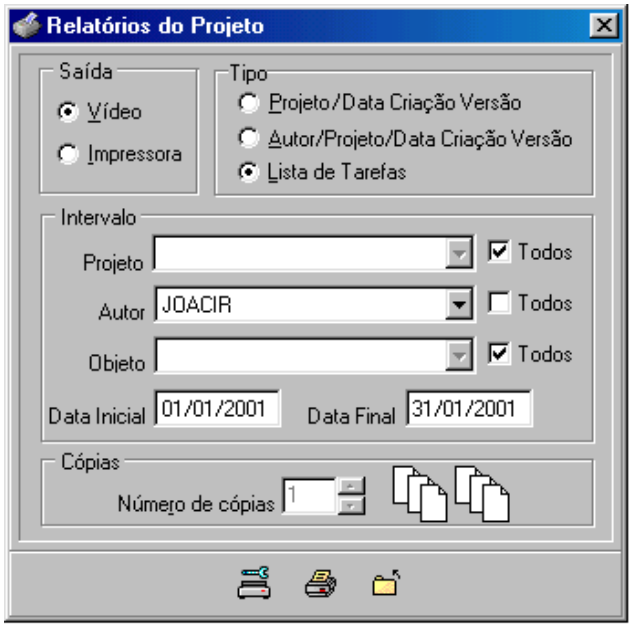
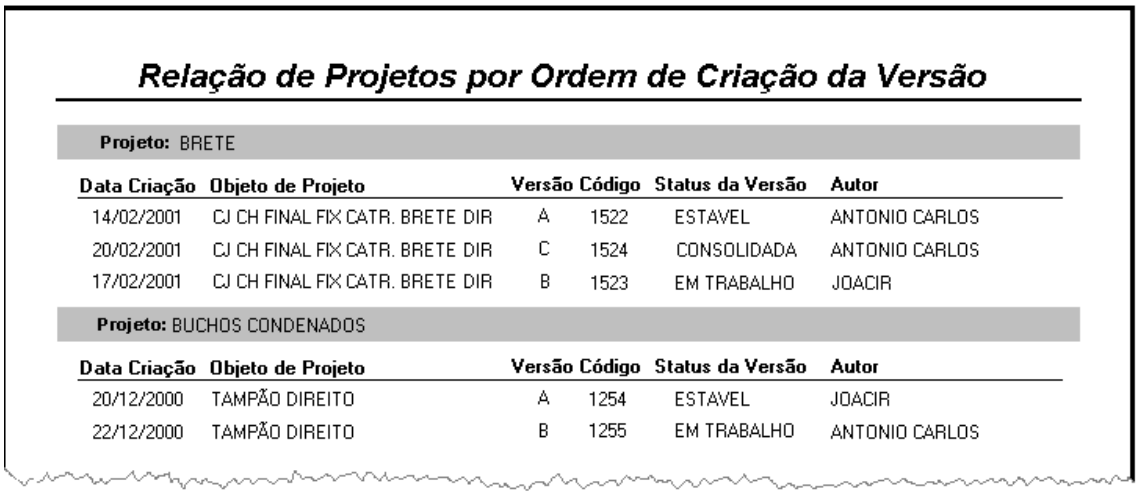


FIGURA 6.36 - Filtro de Relatórios do GerDoc Ábacus

No caso da *figura 6.36*, o usuário deseja visualizar a lista de tarefas do usuário JOACIR no mês de janeiro de 2001, para todos os projetos e objetos de projetos existentes. Poderia-se, por exemplo, querer visualizar a lista de tarefas de um projeto específico.

A não ser que o usuário deseje imprimir a lista, esse procedimento de visualização mostra os mesmos dados que o *navegador geral* mostraria no modo *lista de tarefas*.



Relação de Projetos por Ordem de Criação da Versão

Projeto: BRETE

Data Criação	Objeto de Projeto	Versão	Código	Status da Versão	Autor
14/02/2001	CJ CH FINAL FIX CATR. BRETE DIR	A	1522	ESTAVEL	ANTONIO CARLOS
20/02/2001	CJ CH FINAL FIX CATR. BRETE DIR	C	1524	CONSOLIDADA	ANTONIO CARLOS
17/02/2001	CJ CH FINAL FIX CATR. BRETE DIR	B	1523	EM TRABALHO	JOACIR

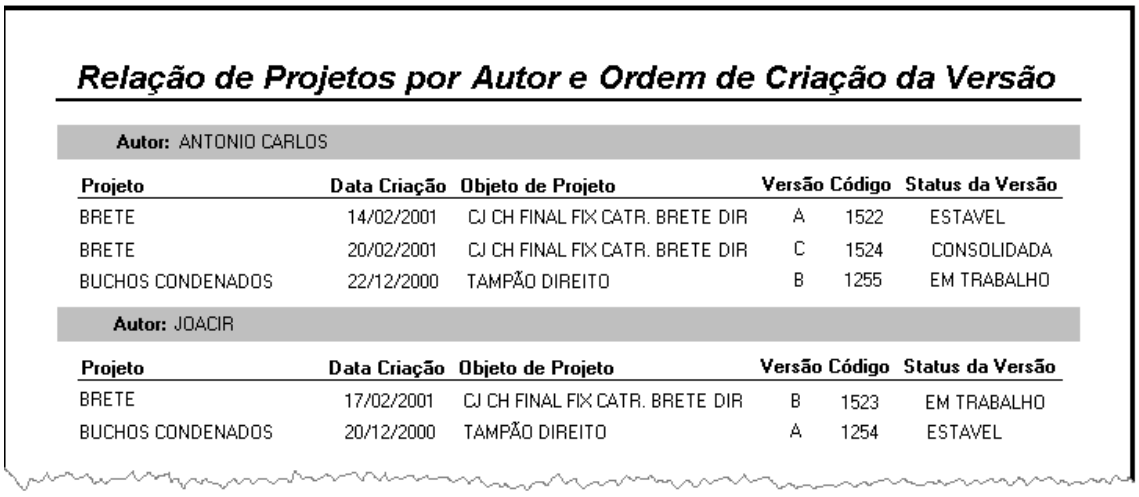
Projeto: BUCHOS CONDENADOS

Data Criação	Objeto de Projeto	Versão	Código	Status da Versão	Autor
20/12/2000	TAMPÃO DIREITO	A	1254	ESTAVEL	JOACIR
22/12/2000	TAMPÃO DIREITO	B	1255	EM TRABALHO	ANTONIO CARLOS

FIGURA 6.37 - Relatório de projetos por ordem de criação da versão

Os três tipos de relatórios existentes são:

① **Relação de projetos por ordem de criação da versão (*figura 6.37*):** Dependendo dos intervalos selecionados, as versões dos objetos de projeto são mostradas separadas por projeto e ordenadas por data de criação. São listados também, de forma tabular, o objeto de projeto, a versão, seu código, seu *status* e o usuário responsável pela criação;



Relação de Projetos por Autor e Ordem de Criação da Versão

Autor: ANTONIO CARLOS

Projeto	Data Criação	Objeto de Projeto	Versão	Código	Status da Versão
BRETE	14/02/2001	CJ CH FINAL FIX CATR. BRETE DIR	A	1522	ESTAVEL
BRETE	20/02/2001	CJ CH FINAL FIX CATR. BRETE DIR	C	1524	CONSOLIDADA
BUCHOS CONDENADOS	22/12/2000	TAMPÃO DIREITO	B	1255	EM TRABALHO

Autor: JOACIR

Projeto	Data Criação	Objeto de Projeto	Versão	Código	Status da Versão
BRETE	17/02/2001	CJ CH FINAL FIX CATR. BRETE DIR	B	1523	EM TRABALHO
BUCHOS CONDENADOS	20/12/2000	TAMPÃO DIREITO	A	1254	ESTAVEL

FIGURA 6.38 - Relatório de projetos por autor e ordem de criação da versão

② **Relação de projetos por autor e ordem de criação da versão (*figura 6.38*):** As versões dos objetos de projeto são mostradas separadas pelo autor responsável pela criação, ordenadas por projeto e data de criação. São listados também, de forma tabular, o objeto de projeto, a versão, seu código e seu *status*;

Lista de Tarefas

Responsável: ANTONIO CARLOS

Tarefa	Data Limite	Hora	Projeto	Objeto de Projeto	Versão
REALIZAR ALTERAÇÕES SOLICITADAS	22/01/2001	15:00	BRETE	CJ CH FINAL FIX CATR. BRETE DIR	A
TERMINAR DOCUMENTO	22/01/2001	16:30	BUCHOS CONDENADOS	TAMPÃO DIREITO	B
REALIZAR ALTERAÇÕES SOLICITADAS	23/01/2001	10:30	BRETE	CALHA	A

Responsável: JOACIR

Tarefa	Data Limite	Hora	Projeto	Objeto de Projeto	Versão
REVISAR DOCUMENTO	22/01/2001	11:00	BRETE	CH TAMPA	C
APROVAR DOCUMENTO	22/01/2001	17:00	BRETE	TAMPA INFERIOR	B

FIGURA 6.39 - Relatório da Lista de Tarefas

③ **Lista de Tarefas (*figura 6.39*):** Separado pelo responsável pela execução das tarefas, estas são mostradas de forma tabular, ordenadas por data e hora limite de execução. Além disso, é mostrado o projeto, objeto de projeto e a versão que deve ser realizada cada tarefa;

6.9 Outras Ferramentas Existentes e Trabalhos Futuros

Adicionalmente aos módulos apresentados nas seções anteriores, o *GerDoc Ábacus* possui outros, que dispensam descrição detalhada. Além de utilitários como *calculadora* e *calendário*, o sistema apresenta a ferramenta de *cópia/restauração* dos dados dos projetos, controlado pelas áreas de acesso, onde apenas usuários com direitos podem realizar a cópia diária em algum dispositivo de armazenamento.

Apesar de se tratar de um sistema que oferece praticamente todos os serviços fundamentais para garantir consistência de dados e manter um bom nível de cooperação entre projetistas, o *GerDoc Ábacus* poderia apresentar alguns recursos adicionais. O principal é o fornecimento automático da composição das versões de objeto de projeto, comentado na seção anterior. Duas ferramentas podem ser inseridas no momento que esse recurso se tornar automático. A primeira é a ferramenta *documentos relacionados*, que permite visualizar as versões de objeto de projeto na qual a que está sendo pesquisada está inserida. A segunda é a ferramenta *partes da montagem*, que permite visualizar a própria composição da versão que está sendo pesquisada.

Um mecanismo de *ajuda* deveria ser fornecido, contendo as informações básicas de operacionalização do sistema. Entende-se que esse mecanismo é fundamental para projetistas novatos ou até mesmo para quem já tem experiência na manipulação do sistema, mas que deseja realizar determinada tarefa e possui dúvidas. Esse mecanismo deve ser sensitivo ao contexto.

Outra ferramenta de auxílio que poderia ser inserida no ambiente do *GerDoc Ábacus* é a *comparação de dois documentos*, permitindo comparar versões diferentes de um mesmo objeto de projeto, para verificar inserções e exclusões de itens. Para isso, é necessário adquirir algum dos produtos existentes no mercado que permita realizar essa operação e inserir no *GerDoc Ábacus*.

7 Conclusões

O principal objeto deste trabalho está voltado ao assunto *documentação dos passos de projetos de engenharia*. A importância deste tópico está ligada diretamente com a maneira cooperativa que projetistas trabalham em ambientes de engenharia. Projetos CAD tendem a ser mantidos em atividade durante longos períodos de tempo, considerando o fato de que muitos objetos de projeto são criados e de que várias versões destes objetos podem existir. Documentar todo esse processo significa permitir uma fácil exploração do espaço de projeto, possibilitando retornos a pontos anteriores para o teste de novas alternativas na busca dos objetivos traçados.

Assim, o sistema gerenciador de documentação técnica para ambientes de engenharia/CAD *GerDoc Ábacus* une recursos considerados importantes para garantir a integridade de dados e o trabalho harmonioso entre projetistas que interagem em projetos específicos. Tais recursos são bem definidos, e fazem parte de duas categorias de gerência: a *de dados* e a *do processo de projeto*.

Objetivando garantir a integridade dos dados dos projetos cadastrados, os recursos de *gerência de dados* existentes no *GerDoc Ábacus* seguem os exigidos para alcançar bons níveis de consistência. Integrado aos recursos de gerência de dados, os recursos de *gerência do processo de projeto* objetivam garantir total organização no trabalho cooperativo existente entre projetistas. Os recursos existentes apresentam um nível satisfatório de controle alcançado, tornando o *GerDoc Ábacus* uma ferramenta de grande utilidade no controle da documentação dos passos de projetos de engenharia.

O estudo realizado nesse trabalho e o próprio sistema gerenciador de documentação técnica para ambientes de engenharia/CAD *GerDoc Ábacus* trazem algumas contribuições: (1) apresentar características, problemas e soluções encontrados em ambientes de engenharia em relação à cooperação necessária entre projetistas durante a concepção de projetos; (2) através de modelos existentes na literatura, apresentar recursos fundamentais para o bom funcionamento da documentação dos passos de projetos de engenharia; (3) introduzir o modelo do *GerDoc Ábacus*, cujas funcionalidades são baseadas no estudo realizado nos dois primeiros itens, e; (4) apresentar a interface do *GerDoc Ábacus*, fundamentado pelo modelo estabelecido, explicitando todos os mecanismos inseridos e a função de cada um dentro do contexto do sistema.

A constatação final acerca do *GerDoc Ábacus* é que ele é um sistema que apresenta um bom nível de integridade e consistência nos serviços prestados e, se bem aplicado em algum ambiente de engenharia, pode contribuir muito na organização e manutenção dos dados relativos aos projetos e no controle das tarefas realizadas pelas equipes de engenheiros inseridas nesse ambiente.

Bibliografia

[BAT85] BATORY, D. S.; KIM, W. Modeling Concepts for VLSI CAD Objects. **ACM Transactions on Database Systems**, New York, v.10, n.3, p.322-346, Sept. 1985.

[BOS91] BOSCH, K. O.; BINGLEY, P.; WOLF, P. Design Flow Management in the NELSIS CAD Framework. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 28., 1991, San Francisco. **Proceedings...** New York: ACM, 1991. v.11, p.711-716.

[BRO92a] BROCKMAN, J. B.; COBOURN, T. F.; JACOME, M. F.; DIRECTOR, S. W. **The Odyssey CAD Framework**. In: IEEE DATC NEWSLETTER ON DESIGN AUTOMATION, IEEE, Spring 1992.

[BRO92b] BROCKMAN, J. B. **A Schema-Based Approach to CAD Task Management**. Pittsburgh, Pennsylvania: Carnegie Mellon University, 1992.

[CEL91a] CELLARY, W.; VOSSEN, G.; JOMIER, G. **Multiversion Object Constellations for CAD Databases**. Germany: Division of Computer Science, University of Giessen, 1991. (Techn. Report 9105).

[CEL91b] CELLARY, W.; JOMIER, G.; KOSZLAJDA, T. Formal Model of an Object-Oriented Database with Versioned Objects and Schema. In: DEXA, 2., 1991. **Proceedings...** [S.l.:s.n.], 1991. p. 239-244.

[CHI90] CHIUEH, T.; KATZ, R. H. A History Model for Managing the VLSI Design Process. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1990, Santa Clara. **Digest of Technical Papers.** [S.l.:s.n.], 1990. p. 358-361.

[CHO86] CHOU, H. T.; KIM, W. A Unifying Framework for Version Control in a CAD Environment. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 12., 1986, Kyoto, Japão. **Proceedings...** [S.l.:s.n.], 1986. p. 336-344.

[DAN89] DANIELL, J.; DIRECTOR, S. W. An Object-oriented Approach to CAD Tool Control Within a Design Fremework. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 26., 1989, Las Vegas. **Proceedings...** [S.l.:s.n.], 1989. p. 197-202.

[DIT88] DITTRICH, K. R.; LORIE, R. A. Version Support for Engineering Database Systems. **IEEE Transactions on Software Engineering**, New York, v.14, n.4, p.429-437, April 1988.

[FRE88] FREITAS, C. M. D. S.; GOLENDZINER, L. G.; WAGNER, F. R. O Sistema AMPLO: um Ambiente Integrado para Projeto de Sistemas Digitais. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 15., 1988, Rio de Janeiro. **Anais...** Rio de Janeiro: [s.n.], 1988. p.272-284.

[GOL93] GOLEDZINER, L. G.; SANTOS, C. S. Versões em Banco de Dados Orientados a Objetos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 8., 1993, Campina Grande. **Anais...** Campina Grande: SBC/UFPb, 1993. p.7-20.

[GOL95a] GOLEDZINER, L. G. **Um modelo de versões para banco de dados orientados a objetos.** Porto Alegre: CPGCC da UFRGS, 1995. 147p. Tese de Doutorado.

[GOL95b] GOLEDZINER, L. Goldstein; SANTOS, C. S. Definição e manipulação de configurações em banco de dados orientados a objetos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 1995, Recife-PE. **Anais...** Recife: UFPE/DI, 1995. p.335-349.

[GOL95c] GOLEDZINER, L. G.; SANTOS, C. S. Uma abordagem multi-nível para suporte de versões em banco de dados orientados a objetos. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 15., 1995, Canela, RS. **Anais...** Porto Alegre: SBC, 1995. p.1127-1138.

[GOL95d] GOLEDZINER, L. G.; SANTOS, C. S. Versions and configurations in object-oriented database systems: a uniform treatment. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 7., 1995, Pune, Índia. **Proceedings...** New Delhi: Mcgraw-Hill, 1995. p.18-37.

[HEA92] HEATON, J. The JESSI-Common-Frame Project – SP4 Evaluation: Evaluating the Options. In: IFIP WG 10.2/WG 10.5 WORKSHOP ON ELECTRONIC DESIGN AUTOMATION FRAMEWORKS, 3., 1992. **Proceedings...** Amsterdam: North-Holland, 1992. p.271-288.

[IOC91a] IOCHPE, C.; LIVI, M. A. C. Suporte à Cooperação em Ambiente de Projeto Assistido por Computador. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 6., 1991, Manaus. **Anais...** Manaus: [s.n.], 1991. p.101-116.

[IOC91b] IOCHPE, C. **Um Modelo de Cooperação Baseado em Características de Objetos e em Ordens Parciais de Passos de Projeto**. Porto Alegre: CPGCC da UFRGS, 1991.

[KAF91] KÄFER, W. A Framework for Version-based Cooperation Control. In: INTERNATIONAL SYMPOSIUM ON DATABASE SYSTEMS FOR ADVANCED APPLICATION, 2., 1991, Tokyo, Japan. **Proceedings...** Tokyo: [s.n.], 1991. p.527-536.

[KAT86] KATZ, R.; CHANG, E.; BHATEJA, R.; PLOUFFE, W. Version Modeling Concepts for Computer-Aided Design Databases. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1986, Washington. **Proceedings...** New York: ACM, 1986.

[KAT90] KATZ, R. Toward a Unified Framework for Version Modeling in Engineering Databases. **ACM Computing Surveys**, New York, v.22, n.4, p.375-408, Dec. 1990.

[KAT92] KATHÖFER, T.; MILLER, J. The JESSI-Common-Frame Project – Sub-project Development. In: IFIP WG 10.2/WG 10.5 WORKSHOP ON ELECTRONIC DESIGN AUTOMATION FRAMEWORKS, 3., 1992. **Proceedings...** Amsterdam: North-Holland, 1992. p.253-269.

[KNA86] KNAPP, D. W.; PARKER, A. C. A Design Utility Manager: The ADAM Planning Engine. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 23., 1986, Las Vegas. **Proceedings...** Las Vegas: [s.n.], 1986. p.48-54.

[LIM97] LIMA, G. M. de A.; TOLEDO, M. B. F. de. Um Modelo de Transações Cooperativas Integrado a um Modelo de Versões. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 1997. **Anais...** [S.l.:s.n.], 1997. p.280-291.

[RYK96] RYKOWSKI, J.; CELLARY, W. Using Multiversion Object-Oriented Databases in CAD/CIM Systems. In: INTERNATIONAL DATABASES AND EXPERT SYSTEMS APPLICATIONS (DEXA), 7., 1996, Zurich, Switzerland. **Proceedings...** Zurich: [s.n.], 1996. p.183-190.

[SAN86] SANTOS, C. S.; OLIVEIRA, J. P. M. de; CASTILHO, J. M. V. de. O modelo E. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 13., 1986, Olinda. **Anais...** Recife: SBC/UFPE, 1986. p.342-350.

[SAN89] SANTOS, C. S. **O modelo E revisado**. Porto Alegre: CPGCC/UFRGS, 1989. (Relatório de pesquisa 119).

[SOA96] SOARES, S. N. **Sistema Gerenciador de Documentação de Projeto**. Porto Alegre: CPGCC da UFRGS, 1996. Dissertação de Mestrado.

[SUT93] SUTTON, P. R.; BROCKMAN, J. B.; DIRECTOR, S. W. Design Management Using Dynamically Defined Flows. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 30., 1993. **Proceedings...** [S.l.:s.n.], 1993. p.239-250.

[WAG91] WAGNER, F. R. Data Management Facilities in the AMPLO Design Framework**.** In: IFIP WORKSHOP ON ELECTRONIC DESIGN AUTOMATION FRAMEWORKS, 1991. **Proceedings...** Amsterdam: North-Holland, 1991. p.71-84.

[WAG92a] WAGNER, F. R. et al. STAR – Um ambiente para a Integração de Ferramentas de Projeto de Sistemas Digitais. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 6., 1991, Belo Horizonte. **Anais...** Belo Horizonte: [s.n.], 1991. p.176-185.

[WAG92b] WAGNER, F. R. et al. Design Version Management in the STAR framework. In: IFIP WORKSHOP ON ELECTRONIC DESIGN AUTOMATION FRAMEWORKS, 1992. **Proceedings...** Amsterdam: North-Holland, 1992. p.85-97.

[WAG94] WAGNER, Flávio Rech. Ambientes de Projeto de Sistemas Eletrônicos. In: ESCOLA DE COMPUTAÇÃO, 9., 1994, Recife, Brasil. **Anais...** Recife: UFPE, 1994. 155p.