On the Support of Activity Patterns in ProWAP: Case Studies, Formal Semantics, Tool Support

Lucinéia Heloisa Thom[1], Cirano Iochpe[2], Manfred Reichert[1], Barbara Weber[3], Droop Matthias[3], Gleison Samuel Nascimento[2], Carolina Ming Chiao[2]

[1] Institute of Databases and Information Systems, University of Ulm
Oberer Eselsberg, 89069, Ulm, Germany

[2] Institute of Informatics, Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500, 91501-970, Porto Alegre, Brazil

[3] Quality Engineering Research Group, University of Innsbruck, Austria

{lucineia.thom, manfred.reichert}@uni-ulm.de,
{ciochpe, gsnascimento, cchiao}@inf.ufrgs.br,
barbara.weber@uibk.ac.at, matthias.droop@student.uibk.ac.at

Abstract. *Recently, research on workflow activity patterns emerged in order to increase the reuse of recurring business functions (e.g., request for task execution, notification, approval and decision). While workflow patterns have been defined for several aspects related to process execution, recurrent business functions have been only partially addressed by existing work. Related to this challenge we proposed a set of seven workflow activity patterns in previous work. In this paper we report on the results of several case studies we performed in Brazilian and European companies in order to investigate how frequently the activity patterns occur in real-world process models. We further formalize the identified activity patterns using π–calculus. This formalization as well as our analysis results are applied in the development of a BPM tool fostering the reuse of business functions specifications.*

1. Introduction

In order to fulfill their business goals, companies have adopted several technologies related to the improvement of their business processes; e.g., workflow management systems and Business Process Management (BPM) tools [Dadam 2000] [Mutschler 2008]. In particular, such technologies enable the definition, execution and monitoring of the operational processes of an enterprise [Lenz, 2007], [Müller 2006], [Weske 2007], whereby different process variants may have to be managed along the process lifecycle [Hallerbach 2008].

A business process is a set of (structured) activities which jointly realize a particular business goal [Weske 2007]. Such activities are related to specific business functions or process fragments (e.g., notification, information request, approval) having a well defined semantics [Thom 2008]. In particular, a certain *process fragment* or *business function* (e.g., enabling document approval) can occur several times within one or different process models. That means multiple logical copies of the same process

fragment may be used with same or different parameterization (e.g. approval by a single actor or by multiple actors). As example consider Figure 1, which shows the procedure for carrying out a medical examination. This process includes the following activities (in the given order): a) First an order entry is created with a request for the medical examination; b) then an appointment with the radiologist is made; c) next, an authorization from the patient for beginning his or her treatment is needed; if the patient does not agree with the treatment all appointments with the radiologist will be cancelled; d) otherwise, the patient is prepared for the examination; e) at the day of the examination the patient is transported to the radiology unit; f) the physician performs the examination; g) based on the results of the examination the physician writes a report; h) as last step, the physician who requested the examination validates the report.

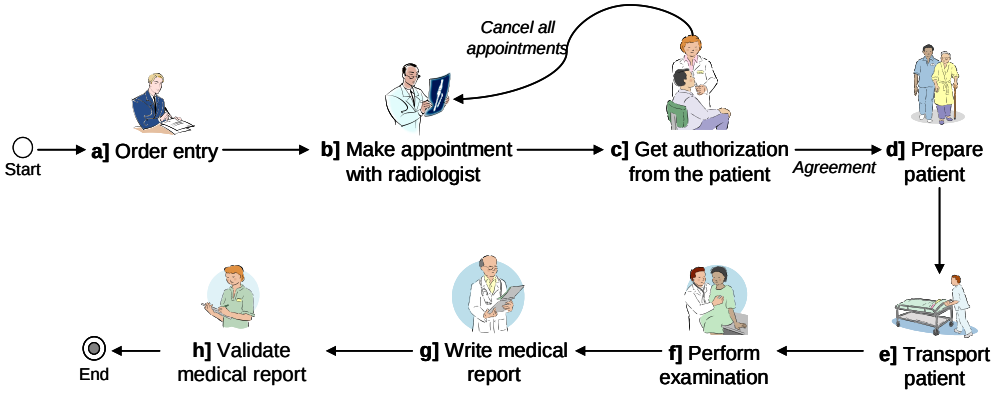


Figure 1. Process for accomplishing a medical examination of a patient

Interestingly, this simple process model comprises (atomic) fragments related to **activity patterns** like *Request for Activity Execution without Answer* (Activities a, b and h), *Request for Activity Execution with Answer* (Activities d, e, f and g), and *Approval* (Activity c). In this paper we use the term *Workflow Activity Pattern* (*WAP* or *activity pattern* for short) to refer to the description of such business functions as they frequently re-occur in business process models.

1.1. Problem Statement

Usually, the above mentioned process *fragments* are re-designed for each workflow application [Flores 1998], [Medina-Mora 1992], [Malone 2004], [zur Muehlen 2002]. This lack of reusing model fragments and process knowledge, in turn, has resulted in high costs and error rates regarding the modeling and maintenance of process-oriented applications.

While some research has been reported on how metadata can be organized to manage large-scale modeling projects [Thomas and Scheer 2006], there is not much work providing evidence for the existence of activity patterns that can be used to define recurrent business functions for real-world process models. Furthermore, there is a lack of investigations proofing the necessity and completeness of respective activity patterns with respect to process modeling. Finally, contemporary BPM tools like Intalio, ARIS

Toolset and WBI Modeler do not support process designers in defining, querying and reusing activity patterns as building blocks for business process modeling.

1.2. Background and Contributions

In this paper we report on the results from ProWAP[1] project, which addresses the aforementioned challenges. We first present an empirical study in which we analyze the relative frequency of the activity patterns presented in [Thom 2006b] in a collection of 239 real-world process models from application domains like quality management, software access control, textile manufacturing, and electronic change management. For selected process categories, we further discuss results of an additional analysis in which we investigate the frequency of co-occurring activity patterns.

The results of this second analysis are additionally utilized for developing a BPM tool, which shall foster the modeling of business processes based on the reuse of activity patterns [Thom 2008]. Given some additional information about the kind of process to be designed, the results of our analysis can be further used by this BPM tool to suggest a ranking of the activity patterns suited best to succeed the last pattern applied during process modeling.

Our BPM tool also provides support for analyzing and verifying model properties of composed processes like syntactical correctness or absence of deadlocks. Basic to this is a formalization of the activity patterns, which further contributes to reduce ambiguities. Different formalisms have been suggested for defining the formal semantics of process modeling languages, including Petri Nets [van der Aalst 2003a], Temporal Logic [Eshuis 2002], and π-calculus [Puhlmann 2005]. In this paper we are using π-calculus to define formal semantics of our activity patterns. This formalism has been successfully applied before in the context of software process formalization [Szturc 1999] as well as for formalizing workflow patterns [Puhlmann 2005] and block-structured process modeling languages (e.g., XLang [Thatte 2001]).

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 gives background information on the workflow activity patterns we identified in previous work. Section 4 evaluates the existence and completeness of these activity patterns with respect to real-world process models. In Section 5 we provide a formalization of the patterns using π-calculus. Section 6 introduces our BPM tool which offers activity patterns to the process designer. Section 7 concludes with a summary and outlook.

2. Survey on Workflow Patterns

A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts [Gamma 2005]. Patterns capture existing, well-proven experience in software development and help to promote good design practices [Buschmann 1996], [Eriksson 2001]. They have been used for many different domains. However, patterns for business process and workflow modeling are still subject of discussion and research. One of the first contributions in this respect was a set of process patterns to be used in connection with the software processes of an organization [Ambler 1998].

[1] http://www.uni-ulm.de/in/iui-dbis/forschung/projekte/prowap.html

Russell (2006a) proposes 43 workflow patterns for describing process. Each pattern represents a routing element (e.g., sequential, parallel and conditional routing) which can be used to process modeling. In the meantime these workflow patterns have been additionally used for evaluating process specification languages and process modeling tools [van der Aalst 2003b], [Wohed 2006]. Rinderle (2006) shows how selected control flow patterns contribute to automatically cope with certain exceptions in process-aware information systems.

A set of data patterns is proposed by [Russell 2005]. These patterns are based on data characteristics that occur repeatedly in different workflow modeling paradigms. Examples are *data visibility* and *data interaction.* In related work, Russell presents a set of resource patterns each of them describing a way through which resources can be represented and utilized in process models [Russell 2004]. A resource is considered as an entity being capable of doing work. It can be either human (e.g., a worker) or non-human (e.g., equipment). Examples of resource patterns include *Direct Allocation* and *Role-Based Allocation*.

Recently, Russell presented a pattern-based classification framework for characterizing *exception handling* in workflow systems [Russell 2006b]. This framework has been used to examine the exception handling capabilities of workflow and BPM tools and to evaluate process specification as well as process execution languages accordingly. As a result, limited support for exception handling in existing workflow management systems could be observed.

Mulyar (2005) proposes 34 implementation patterns to be used in the design of process models with Colored Petri Nets tools [Mulyar 2005]. An example is the *synchronous transfer pattern* which allows transportation of data from one location to another, ensuring that actors who post a request are blocked until they receive the requested information.

Barros (2005) proposes a set of *service interaction patterns*, which allow for service interactions, pertaining to choreography and orchestration, to be benchmarked against abstracted forms of representative scenarios. As example consider the *Send* and the *Send/Receive* patterns.

Altogether Russell and Barros provide a thorough examination of the various perspectives that need to be supported by a workflow language and tool respectively. However, none of these approaches investigate which are the most frequent patterns recurrently used during process modeling and in which way the introduction of such activity patterns eases process modeling. Furthermore, recent work has shown that consideration of the strong linkage existing between data and process allows for sophisticated IT support in all phases of the process lifecycle. For example consider the work done in COREPRO [Müller 2007], [Müller 2008] and ProCycle [Weber 2009]. This observation has not yet been fully covered in research on workflow patterns.

Obviously, broad support for workflow patterns allows for building flexible process-aware information systems. However, an evaluation of a PAIS regarding its ability to deal with process flexibility and change needs a broader view [Weber 2007], [Weber 2008a]. In addition to build-time flexibility (i.e., the ability to model flexible execution behavior based on advanced workflow patterns), run-time flexibility has to be considered as well [Reichert 1997], [Rinderle 2004]. The latter is to some degree addressed by the aforementioned exception handling patterns [Russell 2006b], which

describe different ways for coping with the exceptions occurring during process execution. To also cope with process adaptation, in addition, the aforementioned process change patterns have been introduced by [Weber 2008a], [Weber 2007]. A formalization of these change patterns is given in [Rinderle-Ma 2008]. Weber (2008b), in turn, shows how these change patterns can be used for refactoring process models in order to increase their quality.

PICTURE proposes a set of 37 domain-specific process building blocks. More precisely, these building blocks are used by end users in public administrations to capture and model the process landscape. The building blocks have been evaluated in practice [Becker 2007].

Concerning the formal representation of workflow patterns one of the first approaches used Petri Nets to formalize control flow patterns [van der Aalst 2003a]. Following this, with YAWL [van der Aalst 2005] presented a Petri-net based workflow specification language covering all control flow patterns on Petri nets.

In the context of process algebra, [Brogi 2004] used CCS (*Calculus of Communicating Systems*) to formalize web service flows. In another approach, CCS was used for defining a workflow modeling language called SMAWL (Small Workflow Language) [Stefansen 2005].

The π-calculus, in turn, was first used by Yang Dong and Zhang Shen-Sheng to formalize basic control flows and to define workflow activities [Dang 2003]. Based on the latter work, [Puhlmann 2005] formalize the control flow patterns described in [van der Aalst 2003a] based on π-calculus.

3. Workflow Activity Patterns

Starting with an extensive literature study (e.g., [Flores 1998], [Medina-Mora 2002], [zur Muehlen 2002]) we derived seven *activity patterns*. Each of them represents a frequently recurring business function as it can be found in real-world processes. The identified activity patterns are based on message exchanges where the *sender* and *receiver* of a particular message may belong to the same organization or to different ones the latter applies when dealing with inter-organizational processes [Dadam 2000].

Table 1 gives an overview of the seven activity patterns we identified. Generally, these patterns are close to the abstraction level or vocabulary used within an organization. This, in turn, fosters their reuse when modeling business processes, and therefore contributes to more standardized and better comparable business process models.

We have characterized each activity pattern by giving a description, an example illustrating the context in which the pattern can be applied, a problem motivating its use specific issues arising in this context, design choices (patterns variants) which allow for pattern parameterization keeping the number of distinct patterns manageable, related patterns, and implementation details in particular. The design choices were defined based on the process models we analyzed. For example, we define three variants of the approval pattern, namely single approval (i.e., approval is required from exactly one organizational role), iterative approval (i.e., sequential approval is required from a list of reviewers) and concurrent approval (i.e., approval is required from a list of reviewers simultaneously). Information about the variants we defined for the other six patterns can be found in [Thom 2009]. Note that Table 1 only shows selected pattern variants, but

does not contain all possible parameterizations. Reason is that most of these properties are out of the scope of this paper.

Table 1. Selected variants of activity patterns representing business functions

WAP - Name	Description
WAP 1: *Approval*	An object (e.g., a business document) has to be approved by one or more organizational roles.
WAP 2: *Question-Response*	A question which emerges during process enactment has to be answered. This pattern allows to formulate the question, to identify the organizational role(s) who shall answer it, to send the question to the respective role(s), and to wait for the response(s) (*single-question-response*).
WAP 3: *Unidirectional Performative*	A sender requests the execution of a particular task from another process participant. The sender continues process execution immediately after having sent the request for performing the activity.
WAP 4: *Bi-directional Performative*	A sender requests the execution of a particular task from another process actor. The sender waits until this actor notifies him that the requested task has been performed.
WAP 5: *Notification*	The status or result of an activity execution is communicated to one or more process participants.
WAP 6: *Information Request*	An actor requests certain information from a process participant. He continues process execution after having received the requested information.
WAP 7: *Decision*	This pattern can be used to represent a decision activity in the flow with different connectors to subsequent execution branches. Exactly those branches will be selected for execution whose transition conditions evaluate to true.

In the following sections we informally summarize pattern semantics using UML activity diagrams (cf. Fig. 2). As examples we discuss the *Uni-* and *Bi-directional Performative Pattern* and the *Approval Pattern.* The complete set of activity patterns is described in [Thom 2006a and Thom 2006b]. It is important to observe that the *send* and *receive signals* (cf. Fig. 2) do not configure process activities from the application domain perspective. They are used to implement the logic of the pattern.

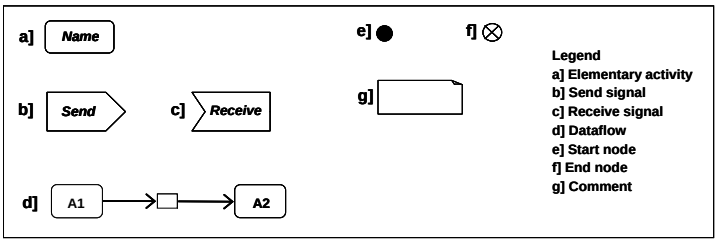


Figure 2. UML Notation used to informally summarize the activity patterns semantics

Approval Pattern (WAP 1)

An object (e.g. a document) has to be approved by one or more organizational roles. Depending on the respective context, the evaluation is executed only once (*single approval*) or multiple times. In the latter variant, approval either can be accomplished in sequence (*iterative* approval) or in parallel (*concurrent* approval).

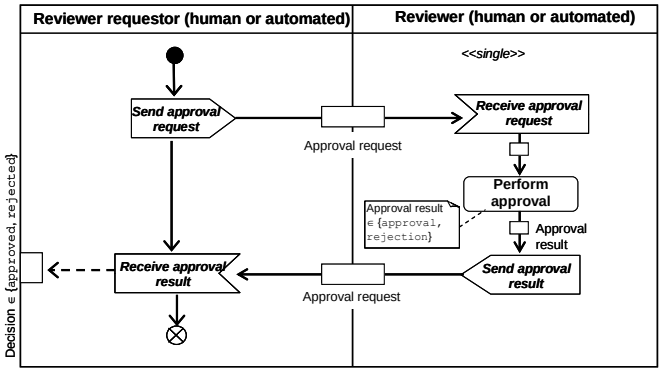


Figure 3. Approval pattern

Fig. 3 illustrates a single approval where an organizational role "reviewer" performs a document review either resulting in approval or disapproval. Generally, the *Approval Pattern* is related to the structural aspect "centralization on decision making"; i.e., the authority to make decisions in organizations can be more or less allocated to the highest positions within an organization (e.g., head of department, manager, director). The more centralized this authority is, the higher will be the number of organizational roles being involved in approval following the organizational scalar chain [Chiavenato, 2000].

Unidirectional Performative Pattern (WAP 3)

The unidirectional performative pattern represents a unidirectional message as described in [zur Muehlen 2002]. A sender uses unidirectional performative messages to request the execution of a particular activity from a receiver (e.g., human or software agent) involved in the process (cf. Fig. 4). The sender continues execution of his part of the process immediately after having sent the request. For example, in a procurement process, the execution of an activity to partially cancel an order can be requested from a manager if some irregularities occur. The flow continues execution after the cancel activity is requested.

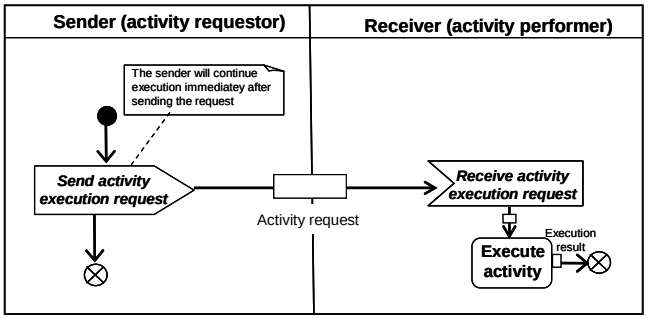


Figure 4. Unidirectional performative pattern

Bi-directional Performative Pattern (WAP 4)

A sender requests the execution of a particular activity from another role (e.g., a human or a software agent) involved in the process. The sender waits until the receiver notifies him that the requested activity has been performed (cf. Fig. 5). As example consider a customer requesting changes concerning the design of a particular product. This triggers a process at the manufacturer site where – first of all – a designer is requested to adapt the product design according to the specifications made by the customer. The

manufacturer process then has to wait until the designer finishes this task. Afterwards the process continues with a review of the new product design by another actor.

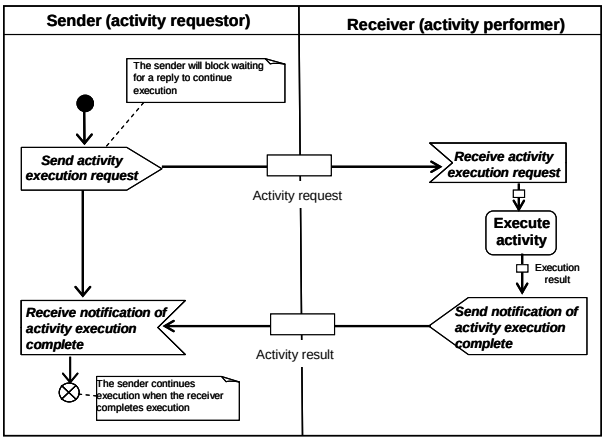


Figure 5. Bi-directional performative pattern

4. Evaluating the Existence and Completeness of Activity Patterns

To identify activity patterns in real-world processes we analyzed 239 process models. These process models have been modeled either with the Oracle Builder tool or an UML modeler. Our analysis has been based on process models instead of event logs, since we consider the semantics and the context of process activities as being fundamental for identifying activity patterns. Altogether, the analyzed process models stem from 14 different organizations – private as well as governmental ones – and are related to different applications like Total Quality Management (TQM), software access control, document management, help desk services, user feedback, document approval, textile manufacturing, and electronic change management. In most of organizations the respective process models are computerized, i.e. they are supported by a process-aware information system. Table 2 summarizes information about involved organizations and analyzed process models.

Table 2. Characteristics of the analyzed process models

Size and Number of Companies	Type of Decision-making	Examples of Analyzed Process Models	Number of Analyzed Models	Computerized
1 x small	Decentralized	Mgmt. of Internal Activities	17	yes
1 x large	Decentralized	TQM; Mgmt. of Activities	11	yes
6 x large	Centralized	TQM; Control of Software Access; Document Mgmt.	133	yes
4 x large	No information available	Help Desk Services, User Feedback; Document Approval	29	yes
1 x large	Rather Centralized	Electronic Change Management	24	yes
1 x medium	Rather Centralized	Facility Management	25	no

4.1. Applied Method

To our knowledge there exist no mining techniques for extracting activity patterns from real-world process models; i.e., contemporary process mining tools like ProM or MinAdept analyze the event logs (e.g., execution or change logs) related to process execution and do not extract information related to the semantics and the (internal) logic of process activities [van der Aalst 2003a], [Ellis 2006], [Günther 2008], [Li 2008a]. Therefore, we perform a manual analysis in order to identify relevant activity patterns as well as their co-occurrences within the 239 process models.

For each workflow activity pattern WAP* we calculate its support value S_{WAP*}, which represents the relative frequency of the respective activity pattern within the set of analyzed process models; i.e., S_{WAP*}:= Freq(WAP*)/239 where Freq(WAP*) denotes the absolute frequency of WAP* within the collection of the analyzed 239 models; for each process model we count at most one occurrence of a particular pattern. Reason is that in some process models, the activity patterns were identified not only in atomic activities but also in pairs (sequences) of activities.

4.2. Frequency of Activity Patterns in Real Process Models

Our analysis has shown that five out of the seven activity patterns (cf. Table 1) are not dependent on a specific application domain or a particular organizational structure (e.g., the degree of centralization in decision making or the standardization of work abilities). Any kind of process model might contain these patterns independent of the application domain (e.g., healthcare, automotive engineering) or the kind of organization (e.g., process oriented, functional and matrix). More precisely, this applies to the following five patterns: UNIDIRECTIONAL and BI-DIRECTIONAL PERFORMATIVE (WAP 3+4), NOTIFICATION (WAP 5), INFORMATIVE (WAP 6) and DECISION MAKING (WAP 7). We can identify these five patterns with high frequency in almost all process models we analyzed. The APPROVAL pattern, in particular, can also be identified with high frequency because of the high degree of centralization regarding decision-making within the considered organizations. This high centralization implies the use of approval activities [Davis 1996]. By contrast, most of the analyzed process models do not contain QUESTION-ANSWER activities. Figure 7 graphically illustrates the relative frequency of each activity pattern with respect to the set of analyzed process models.

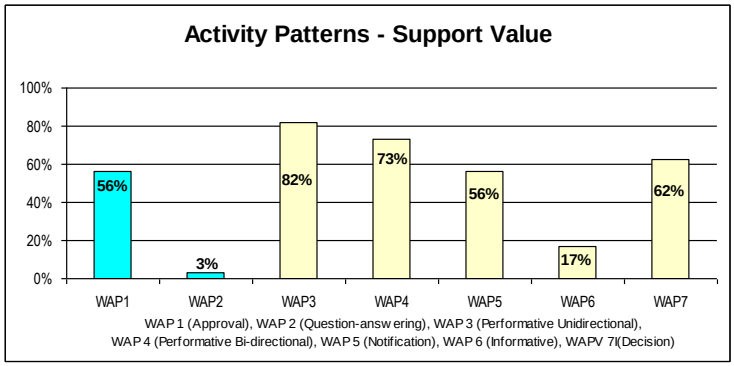


Figure 7. Frequency of activity patterns within real process models

4.2.1. Analyzing Process Models from an Austrian Textile Manufacturer

As data source we consider process models in the context of facility management. The models have been collected at a medium-sized enterprise operating in the Austrian textile industry. The decision-making within the company can be described as rather centralized. In total we analyzed 25 processes which comprise several hierarchy levels and which contain between 2 and 21 activities (for an overview see Fig. 8).

Facility Management	
Job Entry	10 process models
Job Rotation	8 process models
Job Leave	7 process models

Figure 8. Analyzed process models

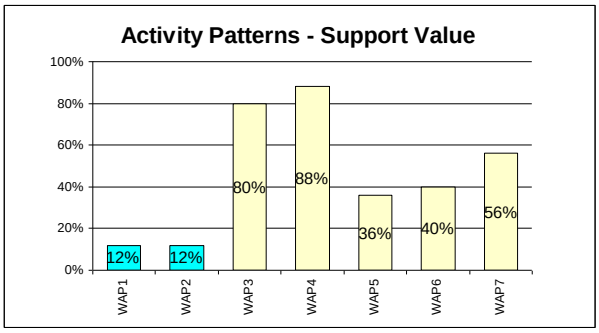


Figure 9. Frequency of activity patterns within Facility Management

Fig. 9 graphically illustrates the frequency of our activity patterns within the set of 25 process models from facility management. The diagram shows that the activity patterns based on organizational structural aspects (WAP 1 and WAP 2) only occur in 12% of the analyzed process models. Interestingly, WAP 1 (*Approval*) occurs relatively rarely for a company with centralized decision-making. This observation can be explained by the fact that the roles associated with the activities are relatively high up in the hierarchy, which reduces the need for approvals. In contrast, the activity patterns based on recurrent functions (WAP 3 – WAP 6) occur much more frequently. In 88% of all process models at least one of these patterns can be identified. WAP 4, for example, occurs much more frequently than all the other activity patterns (in average 4 times per process model). This high frequency of WAP 4 is mainly due to the nature of the analyzed process models, which primarily deal with the distribution of work.

4.2.2. Analyzing Process Models from an Automotive Company

For this analysis we consider process models in the context of Electronic Change Management. The models have been collected at a large enterprise working in the automotive industry. Decision-making within the company can be described as rather centralized. In total we analyze 24 processes which have several hierarchy levels and comprise between 2 and 8 activities.

In this study for each activity pattern we determine its absolute frequency as well as its support value, which represents the relative frequency of the respective patterns within the 24 analyzed process models. We also calculate the support value by design choice

(e.g., single, iterative, concurrent approval). Fig. 10 graphically illustrates the frequency of each activity pattern within the set of 24 process models. The diagram shows that WAP1 – i.e., single approval was identified in 24% of the process models and concurrent approval in 12%. Interestingly, WAP 1 (*Approval*) occurs relatively rarely for a company with centralized decision-making. Like in the facility management processes (cf. Section 4.2.1), the roles associated with the activities are relatively high in the hierarchy, which reduces the need for approvals. By contrast, the activity patterns based on recurrent functions (WAP 3, WAP 5 and WAP7) occur much more frequently. In 54% of all process models at least one of these patterns can be identified.

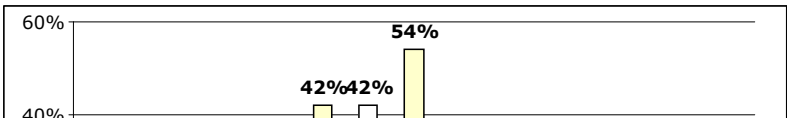


Figure 10. Frequency of activity patterns within Electronic Change Management

4.3. Identifying Co-occurrences of Activity Patterns in Real Process Models

One of the use cases for the *knowledge base* of our BPM tool (cf. Section 6) is based on a mechanism that gives design time recommendations with respect to the most suited activity patterns to be combined with an already used pattern. This mechanism utilizes empirical data we gathered during our case studies, which we also summarize in this section. To obtain the frequencies for pattern co-occurrences, we analyze the sequences of the co-occurring activity patterns in 154 of the 239 process models studied. When this analysis was performed only 154 of the 239 process models were available. In the future we intend to analyze the complete set of processes models, i.e., the 239 as well as further processes we might have access.

In earlier work, we have shown that when classifying the process models into human–oriented (i.e., with human intervention during execution) and fully automated (i.e., with no human intervention during execution) we can identify certain activity patterns more often in one of the two categories [Chiao 2008]. We tried to classify the processes based on common characteristics (e.g., application domain), also considering classifications from the literature in this context. However, most of the studied classifications are based on specific application domains of the related process models [Dowson 1987], [Harrington 1991] and [Weske 2007]. Accordingly, those approaches are not applicable to our analysis because the set of the process models we have been investigating does not cover all the categories addressed by these approaches.

We therefore decided to apply Le Clair (2007) approach which classifies business processes into *system*- and *human-intensive*. System-intensive processes are character-ized by being handled on straight-through basis, i.e., there is minimal or no human intervention and few exceptions occur. Human-intensive processes, in turn, require

people to get work done by relying on business applications, databases, documents, and other actors interacting with them. This type of process requires human intuition or judgment for decision-making during individual process activities.

By classifying our set of process models in those two categories, we obtain 123 human-intensive and 31 system-intensive process models respectively. Remember that in this analysis we consider 154 of the 239 process models. The next step is to evidence the occurrence of the activity patterns in the two categories of process models. Figure 11 shows the frequency of the activity patterns with respect to *system-intensive* and *human-intensive processes*. Note that some patterns (i.e. approval, informative, question-answer) do not appear in the system-intensive process models at all. These patterns are usually related to human activities, i.e., they are executed by an organizational role.

Figure 11. Frequency of co-occurrences of activity patterns

In another analysis we search for frequent co-occurrences of activity patterns within the process models. Relying on these common occurrences the knowledge base should present to process designers a ranking of the most frequent activity patterns which normally follow the activity pattern the user has applied before for process modeling.

For example, Figure 12 shows how often the PERFORMATIVE UNIDIRECTIONAL PATTERN is applied immediately after one of the other activity patterns when regarding the set of process models we analyzed. Note that for the two kinds of processes (i.e. human vs. system intensive processes), a specific pair of patterns may occur with different frequency. For example, the pattern pair PERFORMATIVE UNIDIRECTIONAL → PERFORMATIVE UNIDIRECTIONAL (WAP 3) occurs more often in *system-intensive* than in *human-intensive* processes *(60 % vs. 25 %).* By contrast, the pair PERFORMATIVE UNIDIRECTIONAL → APPROVAL occurs more frequently in *human-intensive processes.*

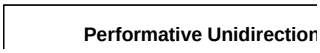


Figure 12. Frequency of co-occurrences of the decision pattern within other pattern

4.4. On the Completeness of Activity Patterns for Process Modeling

When analyzing the process models our main goal was to measure the frequency with which each of the activity patterns occurs within the set of considered process models. We did this in order to verify whether certain process fragments (e.g., task execution request, approval, notification) can be considered as patterns with high probability for reuse in the context of process modeling.

While some patterns can be identified by simply analyzing activity descriptions (e.g., in connection with decisions, approvals and notifications patterns), others require a more detailed analysis. For instance, the *information request pattern* (cf. Table 1) can be identified in the context of activities for which the user has to provide information to the system (e.g., by entering data via an electronic form). In case of the *uni-* and *bi-directional performative* patterns, in turn, both activity descriptions and activity results (e.g., whether results are mandatory or not to trigger the subsequent activity in the process) are important to measure how often patterns occur.

What surprises is the fact that all analyzed process models can be defined as a composition of the investigated patterns; i.e., the described set of activity patterns is necessary and sufficient to design all 239 process models that have been subject of our analysis. (Remember that Table 1 only shows selected pattern variants and does not cover all process fragments that can be configured). Furthermore, in each process, a specific activity pattern may appear multiple times in combination with other patterns.

The latter observation can be considered as very important one, which raises additional research issues. One challenging question, for example, is to what degree the identified patterns will be helpful when integrating them into a workflow modeling tool. In Section 6 we describe such a tool we are currently developing. In particular, it should foster reuse of activity patterns (cf. Section 6). Fig. 13 shows an example of a process model which has been composed solely based on the described activity patterns.

5. Formalizing Activity Patterns with π-Calculus

Basically, the activity patterns are independent of a concrete process modeling language; i.e., they can be adapted to any BPM tool if desired. In order to enable this, a precise formal semantics of the described patterns is needed. Such formal semantics is also fundamental for the correct modeling of business processes and thus for implementing robust process-aware information systems [Reichert 1998]. Thus, when composing process models out of activity patterns it is extremely important to verify their correctness.

Commercial BPM tools (e.g., Oracle BPEL Engine, Intalio), however, do not rely on a formal semantics in order to verify process model correctness. Due to this drawback, research on formal verification of process models (e.g., soundness) has received considerable attention in the research community for more than a decade. As examples of used formalisms consider Petri Nets [van der Aalst 2003a], CCS [Stefansen 2005], π−calculus [Dong 2003], [Puhlmann 2005], and trace notions (e.g., trace equivalence) [Rinderle-Ma 2008]. Respective formalisms apply mathematical methods and allow for formally specifying process models as well as related model properties or model transformations.

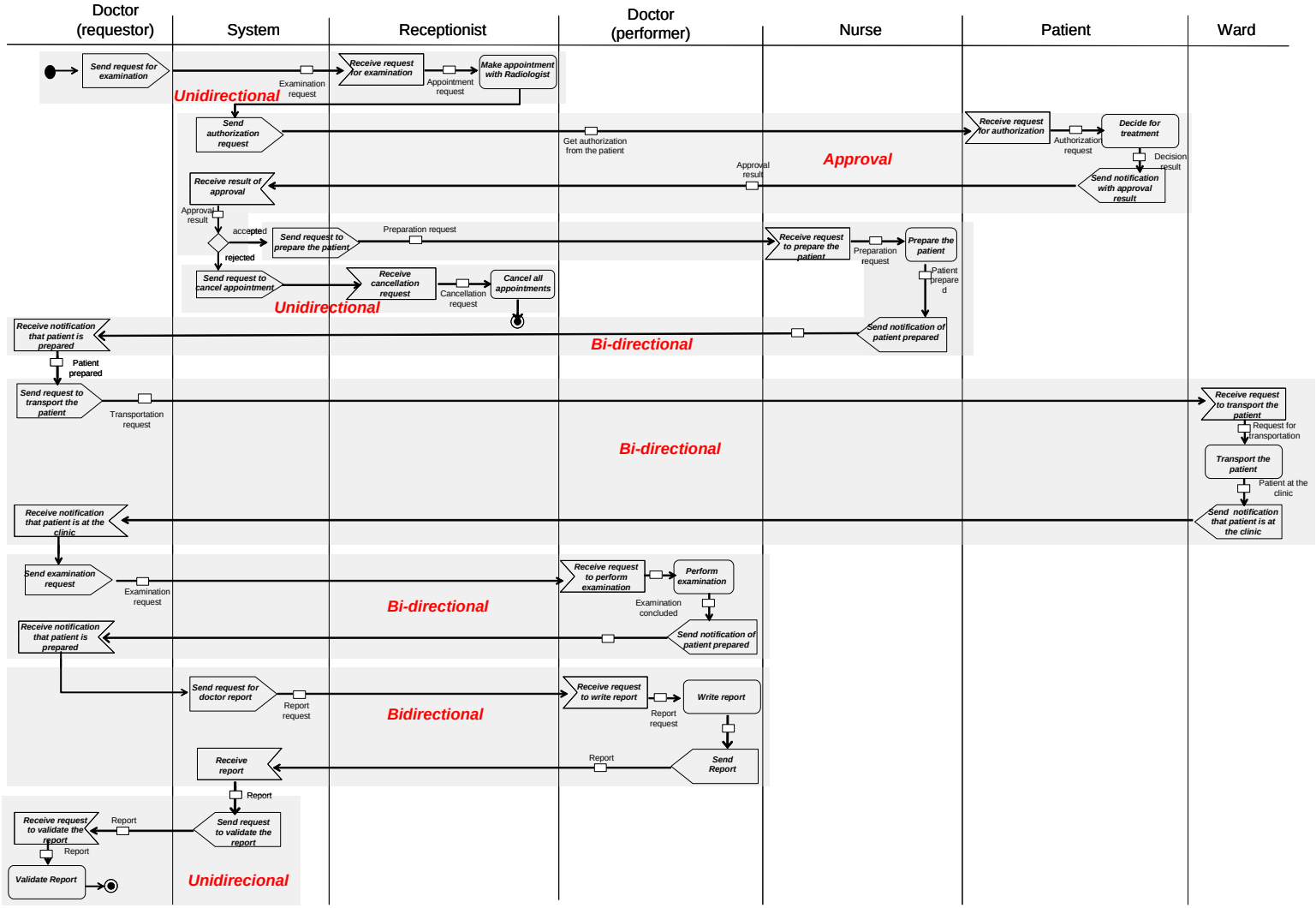


Figure 13. Real medical examination process build up exclusively from the combination of workflow activity patterns

Recently, a couple of approaches has emerged which allow to map business process models (e.g., specified with BPMN or UML) to formal process specifications (e.g., using Petri Nets or π-calculus) [Brogi 2004], [Puhlmann 2005]. Based on such formalizations, for example, it becomes possible to ensure formal process model properties (e.g., soundness [Dehnert 2005], to decide whether two process models are equivalent [Li 2008b], or to optimize and refactor process models [Weber 2008b].

In the following we propose a formalization of the activity patterns using π-calculus. The π-calculus formalism constitutes an extension of CCS [Szturc 1999]. In particular, it allows representing the behavior of concurrent and dynamic processes. In recent work π-calculus was used to formalize business process models and workflow patterns [Smith 2003], [Puhlmann 2005]. As mentioned, there exist other methods (e.g., Petri Nets) which can be used to formalize workflow patterns [van der Aalst 2003a]. Due to their dynamic properties, however, the formalization of specific workflow patterns (e.g., advanced synchronization patterns, multi-choice patterns and cancel patterns) is not trivial when using Petri Nets. In addition, as π-calculus is based on message exchanges, it provides an efficient formalism to represent cross-organizational processes.

Our BPM tool uses the formalization of the activity patterns to create the formal representation of the process models composed out of the different patterns. Thus the composed process model can be simulated and validated already at design time in order to exclude errors. In the following we show how the activity patterns can be formally represented based on the π-calculus formalism. We first introduce the π-calculus formalism, explain how it is used to formalize the activity patterns, and finally present the formal representation of a process modelled solely based on the activity patterns.

5.1. An Introduction to π-Calculus

The π-calculus formalism was introduced by [Milner 1992]. Its underlying theory captures core characteristics of concurrent systems such as mobility and dynamics. Basically the π-calculus consists of processes and names, whereas the names define links between the processes. For the present work we use the monadic version of the π-calculus in accordance with the following grammar (see [Milner 1992]):

$$P, Q ::= 0 \mid \alpha.P \mid (\nu\, a)P \mid P+Q \mid P|Q \mid !P$$

$$\alpha ::= \tau \mid a(b) \mid \bar{a}b \mid [a = b]\alpha$$

Here *a* and *b* are meta-variables over elements of a countable and infinite set of names (*N*). Processes can be the stop process *0*, prefixed process $\alpha.P$, restricted process $(\nu\, a)P$, sum of processes $P+Q$, parallel composition $P|Q$, and replication $!P$. A prefix α can be:

- a silent prefix $\tau.P$ representing an internal action of the process;
- an input prefix $a(b).P$, which receives a new name through *a,* whereupon the process *P* continues with all the occurrences of *b* substituted by the new name;
- an output prefix $\bar{a}b.P$ where the name *b* is sent to another process through *a*;
- a match prefix $[a = b]\alpha.P$; i.e., if *a* equals *b,* the process will continue as $\alpha.P$.

In this work we are considering the semantics as presented in [Milner 1992]; we use a graphical notation to visualize the processes. Figure 14 shows the graphical representation for the following simple process: $A|B$ with $A = a(b).0$ and $B = \overline{ab}.0$.

Figure 14. Example of Graphical Representation

In the used graphical notation, a process is represented by a circle with its name inside the circle (in the following denoted as π-process). Furthermore, the processes are connected by a line representing a communication channel between them. The end of the line, which connects the processes, represents the input prefix. It is marked with a point followed by the communication channel name.

5.2. Mapping Workflow Activities to π-Processes

Our formalization is based on the following idea: all activities contained in a process model are mapped to corresponding π-processes, whereas each π-process has its pre- and post-conditions. In principle, this corresponds to Event-Condition-Action (ECA) rules [Puhlmann 2005]. An ECA rule has three components: an Event, a Condition, and an Action. The *Event* component specifies when a rule must be evaluated. If an event occurs, the *Condition* component will have to be verified. If the condition evaluates to "true", the *Action* component will have to be executed. For process models we must additionally consider other kinds of rules including Event-Condition-Action-Action (ECAA) and Event-Action (EA). It is beyond the scope of this paper to describe ECA rules in details. Further information can be found in [Puhlmann 2005].

Regarding a π-process events are modeled as input prefix. Following such an input prefix one can include an optional condition to be evaluated. This condition is modeled as a match operator (e.g., [a = b]) of the π-calculus. Furthermore, actions are divided in two parts. In the first part the execution of the activity is represented by a silent action of the π-calculus (τ). The second part corresponds to an output prefix that enables synchronization with a subsequent π-process. Thus, a π-process that represents a workflow activity is defined as follows:

$$x\,.\,[a = b]\,.\,\tau\,.\,\bar{y}\,.\,0$$

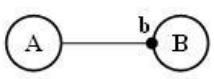
A π-process begins with a trigger of its input prefix *x* which corresponds to the event of an **E**CA rule. Afterwards, matching operator [a = b] is evaluated and mapped to a condition of the E**C**A rule. The component ($\tau\,.\,\bar{y}$) corresponds to the action of an EC**A** rule**;** the π-process executes internal action τ and synchronizes with a subsequent π-process through output prefix *y*.

5.3. Basic Control Flow

Recently, the semantics of the control flow patterns [van der Aalst 2003a] was specified using π-calculus [Puhlmann 2005]. Control flow patterns are particularly useful for defining workflow behavior. In the following we exemplarily consider the control flow patterns *Sequence*, *AND-Split*, and *AND-Join* in order to better understand how process

behavior can be formalized using π-calculus. In addition, the respective control flow patterns will be later used in the formalization of our activity patterns.

Sequence: Using this control flow pattern, an activity is enabled after completion of its preceding activity (within same process model). As aforementioned, activities can be represented as π-processes. Thus, a sequence between two π-processes A and B can be realized by sending a synchronization message from A to B as shown in Figure 15.

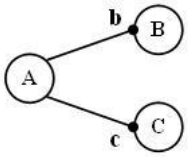


$$A = \tau_A.\bar{b}.\mathbf{0}$$
$$B = b.\tau_B.B'$$

Figure 15. Sequence Pattern

Note that representation of $\tau_A.\ \bar{b}\langle x\rangle\ .\ 0 \mid b(y)\ .\ \tau_B$. B' is reduced to $\tau_A.\ \bar{b}\ .\ 0 \mid b\ .\ \tau_B\ .\ B'$; i.e., the names exchanged between the π-processes are not relevant with respect to the control flow between the two activities. Accordingly, in this paper the label of the message to be transmitted has no influence regarding the execution of the patterns.

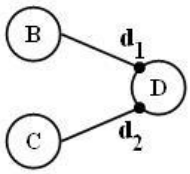
AND-Split: A single thread of control splits into multiple threads of control that can be executed in parallel, thus allowing activities to be executed simultaneously or in any order. Accordingly, a π-process *A* needs to synchronize with the other two π-processes *B* and *C* in parallel, i.e. after executing *A*, activities *B* and *C* are executed in parallel. This representation is depicted in Figure 16.



$$A = \tau_a.(\bar{b}.\mathbf{0}|\bar{c}.\mathbf{0})$$
$$B = b.\tau_B.B'$$
$$C = c.\tau_C.C'$$

Figure 16. AND-Split Pattern

AND-Join: After the execution of two parallel processes it becomes necessary to synchronize them. For example, synchronization of two π-processes *B* and *C* with a subsequent π-process *D* is illustrated by Figure 17.



$$B = \tau_B.\overline{d_1}.\mathbf{0}$$
$$C = \tau_C.\overline{d_2}.\mathbf{0}$$
$$D = d_1.d_2.\tau_D.D'$$

Figure 17. AND-Join Pattern

5.4. Formalizing Workflow Activity Patterns with π-Calculus

π-calculus has proven to be an adequate formalism for enabling the verification of correctness properties of a process [Yang 2003], [Puhlmann 2005], [Szturc 1999]. In the following we sketch how some of the described activity patterns can be formalized based on π-calculus. For a complete formalization we refer to [Nascimento 2007].

Approval Pattern: Fig. 18 shows the formal definition of the single approval pattern, using π-calculus. Note that we represent the approval cycle for exactly one organizational role as bidirectional pattern. The *sender* is represented by `WORKFLOW_APP` and the *receiver* by `REVIEWER`. The pattern synchronizes with an approval (represented through the π-process *NA'*) or with a disapproval (represented through the π-process *NR'*).

$$
\begin{aligned}
APPROVAL(doc) &::= (\nu\ a,b,c)(START(doc)|WORKFLOW\_APP|REVIEWER) \\
\\
START(x) &::= \tau\ .\ \overline{a}\langle x\rangle\ .\ 0 \\
WORKFLOW\_APP &::= a(y)\ .\ \overline{b}\langle y\rangle\ .\ c\ .\ (NA' + NR') \\
REVIEWER &::= b(z)\ .\ \tau_z\ .\ \overline{c}\ .\ 0
\end{aligned}
$$

Figure 18. Formal Definition of the Single Approval Pattern.

Unidirectional Performative Pattern: In order to formalize the unidirectional performative pattern in π-calculus we translate the roles *Sender* and *Receiver* into two concurrent π-processes `SENDER` and `RECEIVER` respectively. Furthermore, as for other patterns there must be a concurrent π-process (here denoted as `START`) to enable interaction with others activities of the workflow. The processing of the pattern begins with the π-process `START`. This π-process synchronizes with the π-process `SENDER` through a message labeled *a*. In the sequel π-process `SENDER`, synchronizes with the π-process `RECEIVER` through a message labeled *b*. After triggering this π-process, it can execute the respective task (here denoted as τ_z). Note that after synchronizes with `RECEIVER` the π-process `SENDER` itself continues as *NS'* without waiting for a response from the π-process `SENDER`. Here, *NS'* means that the pattern can synchronize with a subsequent activity through a message with label *c* (substituting *NS'* for c.0) or terminates the process with *0* (substituting *NS'* for 0). This mapping is presented in Figure 19.

$$
\begin{aligned}
UNIDIRECTIONAL(inf) &::= (\nu\ a,b)(START(inf)|SENDER|RECEIVER) \\
\\
START(x) &::= \tau\ .\ \overline{a}\langle x\rangle\ .\ 0 \\
SENDER &::= a(y)\ .\ \overline{b}\langle y\rangle\ .\ NS' \\
RECEIVER &::= b(z)\ .\ \tau_z\ .\ 0
\end{aligned}
$$

Figure 19. Formal Definition of the Unidirectional Performative Pattern

In Figure 19 the *unidirectional performative pattern* is instantiated passing *inf* as parameter. This label represents the information request which is transmitted to the receiver through the exchange of messages between the π-processes. Note that *inf* is transmitted as message to `SENDER` using label *a* in `START`. The label *inf* then substitutes parameter *x* transmitted to `SENDER`. The same applies to `SENDER` which forwards the message received in *y* through label *b* to `RECEIVER`. Thus `RECEIVER` receives the message for *z*; the notation τ_z represents the internal processing of the

message by RECEIVER. Finally, another important aspect depicted in Figure 19 concerns the scope of the labels between activities. Labels *a* and *b* are used to enable communication between the π-processes START (SENDER) and SENDER (RECEIVER). These labels are restricted to the scope of the pattern. If *NS'* represents the synchronization of the pattern with the following workflow activity, *NS'* will have a label not restricted to the pattern to enable this communication (e.g., *c.0*).

Bi-directional Performative Pattern: To formalize the *bidirectional performative pattern* we synchronize the π-processes SENDER and RECEIVER immediately after RECEIVER has processed its message (τ_z). This synchronization is represented using label *c*. The SENDER continues execution with *NS'* similar to the *unidirectional performative pattern*. Figure 20 shows the π-calculus representation of this pattern.

$$BIDIRECTIONAL(inf) ::= (\nu\ a,b,c)(START(inf)|SENDER|RECEIVER)$$

$$\begin{aligned} START(x) &::= \tau\ .\ \overline{a}\langle x\rangle\ .\ 0 \\ SENDER &::= a(y)\ .\ \overline{b}\langle y\rangle\ .\ c\ .\ NS' \\ RECEIVER &::= b(z)\ .\ \tau_z\ .\ \overline{c}\ .\ 0 \end{aligned}$$

Figure 20. Formal Definition of the Bidirectional Pattern.

5.5. Example of a Process Model Formalized with π-Calculus

To illustrate the use of π-calculus we formalize the healthcare process from Fig. 13. Fig. 21 shows the resulting representation in π-calculus. We use names *an*, *bn* and *cn* (where *n* is a number) as internal names of each activity. The names *sn* (where *n* is a number) represent names to synchronize the activities.

The first activity *OE* (*Order Entry*) begins with τ and the subsequent activities synchronize with the previous ones. For example, *MAR* (*Make Appointment with Radiologist*) synchronizes with *OE* through *s1*. Observe that *VMR* (*Validate Medical Report*) has no name for synchronization (s_n) since this activity terminates the process.

Another important aspect is the activities *MAR* (*Make Appointment with Radiologist*) and *GAP* (*Get Authorization from the Patient*) which represent a loop in the process. When a disagreement occurs the two activities may be repeated. This particular situation is represented through the replication command of π-calculus (!). The two activities are replicated after their execution.

$$
\begin{aligned}
MEDICALBP &::= OE \mid MAR \mid GAP \mid PP \mid TP \mid PE \mid WMR \mid VMR \\
\\
OE(order) &::= (\nu\ a1,b1)(\tau\ .\ \overline{a1}\langle order\rangle\ .\ 0 \mid a1(x)\ .\ \overline{b1}\langle x\rangle\ .\ \overline{s1}\ .\ 0 \mid \\
&\quad\ b1(y)\ .\ \tau_y\ .\ 0) \\
MAR(inf) &::= !((\nu\ a2,b2)(s1\ .\ \overline{a2}\langle inf\rangle\ .\ 0 \mid a2(x)\ .\ \overline{b2}\langle x\rangle\ .\ \overline{s2}\ .\ 0 \mid \\
&\quad\ b2(y)\ .\ \tau_y\ .\ 0)) \\
GAP(auth) &::= !((\nu\ a3,b3,c3)(s2\ .\ \overline{a3}\langle auth\rangle\ .\ 0 \mid \\
&\quad\ a3(x)\ .\ \overline{b3}\langle x\rangle\ .\ c3\ .\ (\overline{s3}\ .\ 0\ +\ \overline{s1}\ .\ 0) \mid \\
&\quad\ b3(y)\ .\ \tau_y\ .\ \overline{c3}\ .\ 0)) \\
PP(patient) &::= (\nu\ a4,b4)(s3\ .\ \overline{a4}\langle patient\rangle\ .\ 0 \mid a4(x)\ .\ \overline{b4}\langle x\rangle\ .\ \overline{s4}\ .\ 0 \mid \\
&\quad\ b4(y)\ .\ \tau_y\ .\ 0) \\
TP(patient) &::= (\nu\ a5,b5)(s4\ .\ \overline{a5}\langle patient\rangle\ .\ 0 \mid a5(x)\ .\ \overline{b5}\langle x\rangle\ .\ \overline{s5}\ .\ 0 \mid \\
&\quad\ b5(y)\ .\ \tau_y\ .\ 0) \\
PE(patient) &::= (\nu\ a6,b6,c6)(s5\ .\ \overline{a6}\langle patient\rangle\ .\ 0 \mid a6(x)\ .\ \overline{b6}\langle x\rangle\ .\ c6\ .\ \overline{s6}\ .\ 0 \mid \\
&\quad\ b6(y)\ .\ \tau_y\ .\ \overline{c6}\ .\ 0) \\
WMR(report) &::= (\nu\ a7,b7,c7)(s6\ .\ \overline{a7}\langle report\rangle\ .\ 0 \mid a7(x)\ .\ \overline{b7}\langle x\rangle\ .\ c7\ .\ \overline{s7}\ .\ 0 \mid \\
&\quad\ b7(y)\ .\ \tau_y\ .\ \overline{c7}\ .\ 0) \\
VMR(report) &::= (\nu\ a8,b8)(s7\ .\ \overline{a8}\langle report\rangle\ .\ 0 \mid a8(x)\ .\ \overline{b8}\langle x\rangle\ .\ 0 \mid \\
&\quad\ b8(y)\ .\ \tau_y\ .\ 0)
\end{aligned}
$$

Legend: OE (*Order Entry*), MAR (*Make Appointment with Radiologist),* GAP (*Get Authorization from the Patient,* PP (*Prepare Patient*), TP (*Transport Patient*), PE (*Perform Examination*), WMR (*Write Medical Report*), VMR (*Validate Medical Report*)

Figure 21. Formal Definition of the Medical Examination Process (cf. Fig. 13)

6. Towards a BPM tool based on Activity Patterns

This section presents basic components of a BPM tool, which allows for the design of process models based on activity patterns and their reuse. The patterns are described by means of an ontology, which has been described in details in [Thom 2008]. This ontology must be used for implementing the recommendation mechanism of our BPM Tool. Altogether, the main goal of this ontology is to better structure the activity patterns, their attributes, and their relationships. In addition, the ontology maintains the use statistics of each activity pattern (in the context of process modeling) as well as the co-occurrences of patterns pairs (cf. Section 4.3).

In principle, basic concepts behind this BPM tool can be added as extension to existing BPM tools as well (e.g., Intalio [Intalio 2006], Aris Toolset [Scheer 2007], or ADEPT2 Process Composer [Reichert 2005]).

Core functionalities of our BPM tool are as follows:

Assisting users in designing proper process models: First, the process designer selects the kind of business process (e.g., human intensive) to be modeled, which is then matched to a set of business functions as maintained in the ontology; i.e., the BPM tool adopts a set of business functions to be used for process modeling in the given context. Following this, the process designer chooses a business function and provides contextual data (e.g., about the organization). This information is then matched to an

activity pattern as maintained in the aforementioned ontology. Furthermore, the BPM tool recommends the use of the respective activity pattern and helps to apply corresponding design choices; i.e., to configure a concrete *pattern variant*. Afterwards, the tool recommends the most suitable activity patterns to be used together with the activity pattern applied before. In addition, it informs the user about how frequently each pair of activity pattern (i.e., the previously applied activity pattern plus the recommended activity pattern) was used in earlier modeling. This module is developed based on the analysis results presented in Section 4.3.

Ontology construction for activity patterns: Our ontology for activity patterns does not only maintain the patterns themselves, but also the frequency with which each pattern has co-occurred with a previously used one. Through analyses of additional process models (e.g., from the automotive as well as the healthcare domain) we aim at increasing the support value of such pairs of activity patterns (cf. Section 4.3). Thus, at design time the pattern pairs being recommended help to design a process model which is closer to the business process as we can find it in the organization.

Core components of our BPM Tool are as follows (cf. Fig. 22):

- **Query Component:** It provides a query mechanism for *matching* the activity patterns maintained by the ontology with the given kind of business process (e.g., human intensive), business function (e.g., approval), organizational context (e.g., level of centralization in decision-making), and corresponding design choice as selected by the user (if not set automatically).
- **Ontology Manager:** It comprises *ontology* and *query mechanism* (enabling queries for business functions and activity patterns respectively). The ontology describes the activity patterns (cf. Fig. 6) and their properties (e.g., attributes and relationships with other patterns). The provided query and update mechanisms give design time recommendations with respect to the most suited activity patterns to be combined with a previously used one. An example of a query would be the selection of the business functions which occur more frequently in system-intensive process models. In addition, our update mechanism has to be used to adapt the relative frequency of each pattern pair (e.g., based on the analysis of new process models) as identified in our process model analysis.
- **Scheme Translation:** This component is responsible for translating a process model (based on translation algorithms) which uses activity patterns as building blocks (stored in XML code) to either a conventional notation (e.g., BPMN) or an existing process execution language (e.g., WS-BPEL). The use of this translation component is optional, i.e., it will be only applicable if the user wants the respective process model being translated to another notation and process execution language respectively.
- **Business Process Model Checker (BPMC)**: The objective of this module is to verify the correctness of the process models as composed out of the activity patterns. The verification is done through simulation and verification of the properties of the composed process models. This module consists of two parts:
  - *Translation of process model into a π-calculus representation*: consists of algorithms to translate a composed process model into a π-calculus

representation. This translation is done based on the formalization of the activity patterns as sketched in Section 4.3.

- *Model Checking Algorithms*: the obtained π-process is simulated through model checking algorithms as known from π-calculus theory. The simulation must prove the correctness properties of the modeled process. Related to that we are considering the use of existing tools for model checking in π-calculus (e.g., [Björn 1994]). Thus the BPMC module will only interact with the model checking tool, without considering the complexity of the algorithms used to simulate a π-process execution.

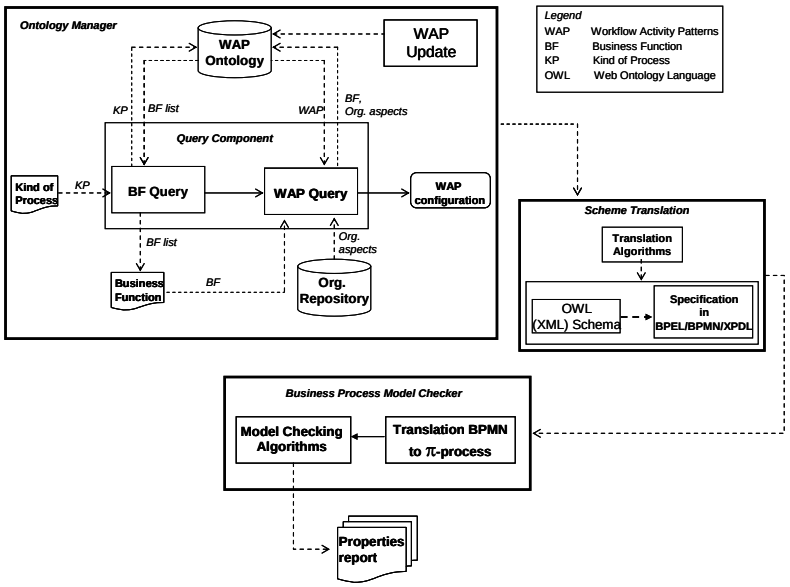


Figure 22. Architecture of the Process Modeling Tool

Altogether our BPM tool fosters the modeling of business processes based on the reuse of activity patterns. In particular, the recommendation mechanism supported by this tool can be considered as a very important functionality which shall optimize the process design and increase its correctness. Finally, we plan to use our tool for conducting a series of experiments in which we compare process modeling with and without activity pattern support.

7. Summary and Outlook

This paper has deal with workflow activity patterns, each of them based on a frequent process fragment (e.g., a task execution request, notification activity, approval). The patterns are tool-independent, which make them easier to be adapted for any business process modeling (BPM) tool. Moreover, based on a relatively small set of patterns a multitude of processes can be modeled. This, in turn, contributes to reduce complexity with user learning. The analysis of 239 processes models, stemmed from different organizations and application domains, has evidenced that the patterns appear to be necessary and enough to design at least the process models we have analyzed. We have also investigated how often specific co-occurrences of activity patterns appear in a

selected set of the studied process models. These analyses must be used in the development of a recommendation BPM tool.

By formally representing the patterns using π-calculus we expect to reduce their ambiguity. Moreover, a workflow model written in π-calculus can express the dynamic behavior of the business process, thus making it possible to verify formal properties of the model, such as soundness, deadlocks, livelocks and equivalence.

The specification of processes in π-calculus can also improve workflow optimization. A business process written in π-calculus can be optimized through the identification of active names. Nascimento (2006) shows algorithms to optimize π-processes through active names collection. Thus, we could map business processes to π-calculus and use algorithms to reduce the size of the business process.

The π-calculus is a textual notation (action based) and it has no intuitive graphical notation. To minimize this problem, we intend to map some conceptual language such as BPMN to the π-calculus. This would allow us to use the π-calculus as an internal representation of a framework for workflow modeling. By doing that we expect to enable the specification based on a business process in high-level notation, but without losing the possibility of formal verification.

As future work we will consider the results of the aforementioned case studies as well as the analyses of other process models in order to derive new pattern variants and patterns, respectively. We also intend to make an experiment for comparing process modeling with and without workflow activity patterns support. Last but not least, we intend to investigate how to transform process models defined with our tool (and being based on the activity patterns) into conventional notations and languages respectively.

Acknowledgements

This work was done in the ProWAP project (see http://www.uni-ulm.de/in/iui-dbis/forschung/projekte/prowap.html). The authors would like to acknowledge the Coordination for the Improvement of Graduated students (CAPES), the Institute of Databases and Information Systems of the University of Ulm (Germany), the Informatics Institute of Federal University of Rio Grande do Sul (Brazil) and the Quality Engineering Research Group of University of Innsbruck (Austria).

References

Ambler, S. W. (1998) "Introduction to Process Patterns". Ambsoft WP http://www.ambysoft.com/processPatterns.pdf.

Barros, A., Dumas, M., and ter Hofstede, A. (2005) "Service Interaction Patterns". In: *Proc. 3rd Int'l Conference on Business Process Management (BPM'05),* LNCS 3649, pp. 302-318.

Becker, J., Lis, L., Pfeiffer, D., and Räckers, M. (2007) "A Process Modeling Language for the Public Sector – the PICTURE Approach". In: *Wybrane Problemy Elektronicznej Gospodarki*, pp. 271-281.

Björn, V., and Faron, M. (1994) "The Mobility Workbench: A Tool for the pi-Calculus". In: *Proc. Conf. Computer Aided Verification (CAV'94)*, Springer, LNCS 818, pp. 428-440.

Brogi, A., Canal, C., Pimentel, E., and Vallecillo, A. (2004) "Formalizing web service choreographies". In: *Electr. Notes Theor. Comput. Science.* pp. 73-94.

Buschmann, F. (1996) “Pattern-oriented software architecture: a system of patterns”. John Wiley, New York.

Crowston. K. (1994) “A Taxonomy of Coordination Dependencies and Coordination Mechanisms”. *MIT Centre for Coordination Science*, Cambridge, MA.

Chiao, C., Thom, L.H., Iochpe, C., and Reichert, M. (2008) “Verifying Existence, Completeness and Sequences of Semantic Process Patterns in Real Workflow Processes”. In: *Proc. of the Simpósio Brasileiro de Sistemas de Informação*. Rio de Janeiro: UNIRIO, Brazil. p. 164-175.

Chiavenato, I. (2000) Business: theory, process and practice. 3rd edition, Makron Books, São Paulo.

Dadam, P. and Reichert, M., and Kuhn, K. (2000) Clinical Workflows - The Killer Application for Process-oriented Information Systems? In: *Proc. 4th Int'l Conf. on Business Information Systems (BIS'00)*, Poznan, Poland. Springer, pp. 36-59.

Davis, M.R. and Weckler, D.A. (1996). “Practical Guide to Organization Design”. Crisp Publications, Boston.

Dehnert, J., and Zimmermann, A. (2005): “On the Suitability of Correctness Criteria for Business Process Models.” In: *Proc. 3rd Int’l Conf. on Business Process Management (BPM’05)*, Berlin. LNCS 3649. pp. 386-391.

Dong, Y. and Shen-Sheng, Z. (2003) “Approach for workflow modeling using π-calculus”. In: *Journal of Zhejiang University Science*, pp. 643–650.

Dowson, M. (1987) “Interaction in the Software Process Review of the 3rd International Software Process Workshop”. In: *Proc. of the 9th Int’l Conference on Software Engineering*, Monterey, CA, pp. 36-41.

Eshuis, H. (2002) “Semantics and Verification of UML Activity Diagrams for Workflow Modelling”. PhD thesis, University of Twente, The Netherlands.

Ellis, C. (2006) “Workflow Mining: Definitions, Techniques, Future Directions”. In *Workflow Handbook 2006*. Lighthouse Point: Future Strategies, pp. 213-228.

Eriksson, H. E., and Penker, M. (2001) “Business Modeling with UML”. John Wiley & Sons.

Flores, F. (1998) “Computer Systems and the Design of Organizational Interaction”. In: *ACM Trans. Inf. Syst*. New York, USA. 6(2):153-172.

Gamma, E. (1995) “Design Patterns”. Addison-Wesley.

Günther, C.W., Rinderle-Ma, S., Reichert, M., van der Aalst, W.M.P., and Recker, J. (2008) “Using Process Mining to Learn from Process Changes in Evolutionary Systems”. In: *Int. Journal of Business Process Integration and Management*, 3(1):61-78.

Hallerbach, A., Bauer, T., and Reichert, M. (2008) Managing Process Variants in the Process Lifecycle. In: *Proc. of the 10th Int'l Conf. on Enterprise Information Systems (ICEIS'08)*, June 2008, Barcelona, Spain. pp. 154-161

Harrington, H. J. (1991) “Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness”. McGraw-Hill.

IDS Scheer (2007) “Aris Platform: Product Brochure”. http://www.ids-scheer.com/set/82/PR_09-07_en.pdf.

Intalio (2006) “Creating Process Flows”, http://bpms.intalio.com.

Le Clair, C. and Teubner, C. (2007) “The Forrester Wave: Business Process Management for Document Processes”, Q3.

Lenz, R., and Reichert, M. (2007) “IT support for healthcare processes - premises, challenges, perspectives.” In: *Data and Knowledge Engineering*, 61:39-58

Li, C., Reichert, M., and Wombacher, A. (2008a) “Discovering reference process models by mining process variants”. In: *Proc. 6th Int'l Conference on Web Services (ICWS'08)*, September 2008, Beijing, China. IEEE Computer Society Press.

Li, C., Reichert, M., and Wombacher, A. (2008b) “On Measuring Process Model Similarity based on High-level Change Operations.” In: *Proc. 27th Int’l Conference on Conceptual Modeling (ER'08)*, Barcelona, Spain. LNCS 5231, pp. 248-264.

Yang, D., and Zhang, S. (2003) “Approach for workflow modeling using Pi-calculus”. In: *Journal of Zhejiang University SCIENCE*. 4(6):643-650.

Malone, T.W., Crownston, K., and Herman, G.A. (2004) “Organizing Business Knowledge: The MIT Process Handbook”. ISBN 0-262-13429-2.

Medina-Mora, R. (1992) “The action workflow approach to workflow management technology”. In: *Proc. of the 1992 ACM conference on Computer-supported cooperative work*. Toronto, Ontario, Canada. pp. 281-288.

Milner, R., Parrow, J., and Walker D. (1992) “A Calculus of Mobile Processes”. Technical Report in Information and Computation, 100(1), University of Edinburgh.

Mintzberg, H. (1995) “Structure in Fives: Designing Effective Organizations”. São Paulo: Atlas.

Müller, D., Herbst, J., Hammori, M., and Reichert, M. (2006). “IT support for release management processes in the automotive industry”. In: *Proc. 4th Int'l Conf. on Business Process Management (BPM'06)*, Vienna, LNCS 4102, pp. 368-377.

Müller, D., Reichert, M., and Herbst, J. (2007) “Data-driven Modeling and Coordination of Large Process Structures”. In: *15th Int'l Conf. on Cooperative Information Systems (CoopIS'07)*, Vilamoura, Portugal. LNCS 4803, pp. 131-149.

Müller, D., Reichert, M., and Herbst, J. (2008) “A New Paradigm for the Enactment and Dynamic Adaptation of Data-driven Process Structures”. I: *20th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'08)*, Montpellier, France. Springer, LNCS 5074, pp. 48-63.

Mulyar, A., and van der Aalst, W.M.P.(2005) “Patterns in Colored Petri Nets”. WP 139, Eindhoven University of Technology, The Netherlands.

Mutschler, B., Reichert, M., and Bumiller, J. (2008) “Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors and Implications”. In: *IEEE Transactions on Systems, Man, and Cybernetics*, 38(3):280-291.

Nascimento, G. N. and Moreira, A. (2006) “Identificação de Nomes Ativos em Cálculo-p baseada em Tipos”. In: *Proc. of the 9th Brazilian Symp. On Formal Methods*, Natal, Brazil. pp. 89-104.

Puhlmann, F. and Weske, M. (2005) “Using the π-calculus for formalizing workflow patterns”. In: *Proc. 3rd Int’l Conference on Business Process Management*

(BPM'05), Nancy, France, Springer, LNCS 3649. pp. 153-168.

OASIS (2007) "Web services business process execution language version 2.0". Technical report, OASIS Standard.

Smith, H., and Fingar, P. (2003) "Business Process Management: The Third Wave". Meghan-Kiffer Press.

Szturc, R., Vondrak, I. and Kruzel, M. (1999) "Application of π-Calculus for Software Process Formalization". Technical University of Ostrava.

Reichert, M. and Dadam, P. (1998) "ADEPTflex-Supporting Dynamic Changes of Workflows Without Losing Control". In: *Journal of Intelligent Information Systems*, 10(2):93-129.

Reichert, M., Rinderle, S., Kreher, U., and Dadam, P. (2005): "Adaptive process management with ADEPT2". In: *Proc. Int'l Conf. on Data Engineering (ICDE'05)*, Tokyo, Japan. IEEE Computer Society Press, pp. 1113-1114.

Reichert, M., Dadam, P. (1997) "A Framework for Dynamic Changes in Workflow Management Systems". In: *Proc. 8th Int'l Workshop on Database and Expert Systems Applications*, Toulouse, France. pp. 42-48.

Rinderle, S., and Reichert, M. (2006) "Data-Driven Process Control and Exception Handling in Process Management Systems". In: *Proc. 18th Int'l Conf. on Advanced Information Systems Engineering (CAiSE)*, Luxembourg, LNCS 4001, pp. 273-287.

Rinderle, S., Reichert, M., and Dadam, P. (2004) "Correctness criteria for dynamic changes in workflow systems - a survey". In *Data and Knowledge Engineering*, 50(1):9-34.

Rinderle-Ma, S., Reichert, M., and Weber, B. (2008) "On the Formal Semantics of Change Patterns in Process-aware Information Systems". In: *Proceedings 27th Int'l Conference on Conceptual Modeling* (ER'08), Barcelona, Spain. Springer, LNCS 5231, pp. 232-247.

Russell, N. (2004) "Workflow Resource Patterns". FIT-TR-2004-01, Queensland University of Technology, Brisbane.

Russel, N., Hofstede, A., and Edmond, D. (2005) "Workflow Data Patterns: Identification, Representation and Tool Support". In: *Proc. 24th Int'l Conf. on Conceptual Modeling (ER'05)*, Springer, LNCS 3716, pp. 353 -368.

Russell, N., ter Hofstede, A., van der Aalst, W.M.P., and Mulyar, N. (2006a) "Workflow Control-Flow Patterns: A Revised View". BPM Center Report BPM-06-22, BPMcenter.org.

Russell, N., van der Aalst, W.M.P, and ter Hofstede, A. (2006b) "Workflow Exception Patterns". In: *Proceedings CAiSE'06*, pp.288-302.

Stefansen, C. (2005) "Smawl: A Small Workflow Language Based on CCS". In: 17th *Conference on Advanced Information Systems Engineering*, Porto, Portugal.

Thatte, S. XLANG (2001) "Web Services for Business Process Design". Technical Report, Microsoft Corporation.

Thom, L., Iochpe, C., Amaral, V., and Viero, D. (2006a) Toward Block Activity Patterns for Reuse in Workflow Design. In: *Workflow Handbook*, pp. 249–260.

Thom, L. (2006b) "A Pattern –Based Approach for Business Process Modeling", PhD

thesis, Federal University of Rio Grande do Sul, Porto Alegre, Brazil.

Thom, L., Reichert, M., Chiao, C.M., Iochpe, C., and Hess, G. N. (2008) "Inventing Less, Reusing More and Adding Intelligence to Business Process Modeling". In: *Proc. 19th Int'l Conference on Database and Expert Systems Applications (DEXA '08)*, Turin, Italy, LNCS 5181, pp. 837-850.

Thom, L. H., Reichert, M., and Iochpe, C. (2009) "Activity Patterns in Process-aware Information systems: Basic Concepts and Empirical Evidence". In: *International Journal of Process Integration and Information Management (IJPIM)*. (accepted for publication).

Thomas, O. and Scheer, A. W. (2006) "Tool Support for the Collaborative Design of Reference Models - A Business Engineering Perspective". In: *Proc. 39th HICSS*, CD-ROM.

Weske, M. (2007) "Business Process Management: Concepts, Languages, Architectures". Springer, Berlin.

Weber, B., Rinderle, S., and Reichert, M. (2007) "Change Patterns and Change Support Features in Process-Aware Information Systems". In *Proc. 19th Int'l Conference on Advanced Information Systems Engineering (CAiSE)*, Springer, LNCS 4495, pp. 574-588.

Weber, B., Reichert, M., and Rinderle-Ma, S. (2008a) "Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems". *Data and Knowledge Engineering*, Vol. 66, pp. 438 – 466.

Weber, B. and Reichert, M. (2008b) "Refactoring Process Models in Large Process Repositories". In: *Proc. 20th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'08)*, June 2008, Montpellier, France, LNCS 5074, pp. 124-139.

Weber B., Reichert M., Wild W. and Rinderle-Ma S. (2009) "Providing Integrated Life Cycle Support in Process-Aware Information Systems". In: *International Journal of Cooperative Information Systems*, 18(1), (to appear).

Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A., and Russell, N. (2006) "On the Suitability of BPMN for Business Process Modeling". In: *Proc. 3rd Int'l Conf. Business Process Management (BPM'06)*, pp. 161-176.

van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A. (2003a) "Workflow Patterns". In: *Distributed and Parallel Databases*, 14(3): 5-51.

van der Aalst, W.M.P.(2003b) "Patterns and XPDL: A Critical Evaluation of the XPDL Process Definition Language". QUT Technical Report FIT-TR-2003-6, Queensland University of Technology, Australia.

van der Aalst, W.M.P.(2005) "YAWL: Yet Another Workflow Language". In: *Information Systems*, 30(4): 245-275, 2005.

zur Muehlen, M. (2002) "Workflow-based process controlling: foundations, design, and application of workflow-driven process information systems". Logos Publ.