Activity Patterns in Process-aware Information Systems: Basic Concepts and Empirical Evidence

Lucinéia Heloisa Thom*, Manfred Reichert

Institute of Databases and Information Systems,
Ulm University, James-Frank-Ring, 89069 Ulm, Germany
E-mail: {lucineia.thom, manfred.reichert@uni-ulm.de}
*Corresponding author

Cirano Iochpe

Institute of Informatics,
Federal University of Rio Grande do Sul, Porto Alegre, Brazil
E-mail: ciochpe@inf.ufrgs.br

Abstract: Recently, a variety of workflow patterns was suggested for capturing different aspects in process-aware information systems (PAISs) including control and data flow, resources, process change, and exception handling. All these patterns are highly relevant for implementing PAISs and for designing process modeling languages. However, current patterns provide only a partial answer to the question which business functions a designer might want to reuse when modeling processes. This paper presents a revised version of a collection of activity patterns to deal with this challenge. Each of them is related to a recurrent business function as it can be frequently found in process models (e.g., task execution request, notification, approval). We describe the identified activity patterns and their variants in detail. The main purpose of our paper is to discuss results from empirical studies, in which we analyzed more than 200 process models in order to evidence the practical relevance of the patterns. This includes a detailed analysis of the context in which activity patterns occur as well the frequency of this occurrence. These empirical findings can be used for the design of more intelligent, pattern-based process modeling tools.

Keywords: workflow activity patterns; business functions; business process management; business process modeling.

Reference to this paper should be made as follows: Thom, L.H., Reichert, M., Iochpe, C. (2009) 'Activity Patterns in Process-aware Information Systems: Basic Concepts and Empirical Evidence', *Int. J. Business Process Integration and Management*.

Biographical notes:

Lucinéia Heloisa Thom is a visiting scientist at the University of Ulm in the group of Manfred Reichert. She received a Bachelor degree in Computer Science from the University of Santa Cruz do Sul, Brazil and a Master degree in Computer Science from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 1999 and 2002, respectively. In 2006 she received her PhD in Computer Science from UFRGS. From 2004 to 2005 she developed part of her thesis research abroad at the Institute for Parallel and Distributed Systems of University of Stuttgart. Her research interests are in the area of workflow systems with a special focus on meta models, business process modeling and workflow patterns. She has published many articles in these fields.

Manfred Reichert received a PhD in Computer Science and a Diploma in Mathematics. Since January 2008 he has been Full Professor at the University of Ulm. From 2005 to 2007 he worked as Associate Professor at the University of Twente (UT). At UT, he was also Leader of the strategic research initiatives on E-health (2005 - 2007) and on Service-oriented Computing (2007), as well as member of the Management Board of the Centre for Telematics and Information Technology, which is the largest ICT research institute in the Netherlands. He has worked on advanced issues related to process management technology, service-oriented computing, and databases and information systems. Together with Peter Dadam, he pioneered the work on the ADEPT process management system, which

currently provides the most advanced technology for realising flexible process-aware information systems. Manfred was PC Co-chair of the BPM'08 conferene in Milan, Italy and will be General Co-chair of the BPM'09 conference in Ulm, Germany.

Cirano Iochpe received a PhD in Computer Science from the University of Karlsruhe, Germany in 1989 and a M.Sc. in Computer Science from the Federal University of Rio Grande do Sul (UFRGS), Brazil in 1984. In 1990 he started working as Associate Professor at the Informatics Institute of UFRGS. Since 2005 he has led the projects sector of PROCEMPA, the ITC Public Company of the city of Porto Alegre. Cirano has coordinated several research projects in the area of information systems, especially in the context of business process management and geographical information systems. Related to these areas he published several papers. In 2008 he received the UTC APEX AWARD from UTC and Motorola as recognition of his efforts in social as well as digital inclusion oriented projects, particularly in the areas of telehealth systems and applications.

1 INTRODUCTION

For several reasons companies are developing a growing interest in improving the efficiency and quality of their internal *business processes* and in optimizing their *interactions* with customers and business partners (Mutschler, 2008a), (Dadam, 2000), (Lenz, 2007), (Müller, 2006). During the last years we have seen an increasing adoption of *business process management* (BPM) tools by enterprises as well as emerging standards for business process specification and execution (e.g., BPMN, BPEL) in order to meet these goals (Weske, 2007). Respective technologies (e.g., workflow management systems, case handling tools) enable the definition, execution, and monitoring of the operational processes of an enterprise (Mutschler, 2008b). In connection with Web service technology, in addition, the benefits of business process management from within a single enterprise can be transferred to cross-organizational business processes as well (Reichert, 1999), (Khalaf, 2006).

1.1 Problem Statement

For (computerized) business processes there exists a variety of *business functions* and *process fragments*, respectively, which can be understood as self-contained activity blocks with a specific and well defined semantics (Thom, 2006), (Thom, 2007b). In particular, a certain *process fragment* (e.g., enabling document approval) may occur several times within one or different process models; i.e., multiple logical copies of the same process fragment may be used with same or different parameterization (e.g. approval by a single actor vs. approval by multiple actors). As example consider Figure 1. The depicted *travel booking process* includes the following partially ordered activities: (a) receiving a flight booking request; (b) a secretary verifies whether there is an available flight for the requested period; (c) if there is no available flight the booking requestor will be notified accordingly; (d) otherwise, a financial manager will authorize the purchase of the tickets; (e) if no approval is given, the secretary and the requestor will be notified that ticket purchase has not been authorized; (f) after approval the secretary must proceed with buying the electronic ticket which is then sent to the requestor. Altogether, the structure of this business process comprises a set of *fragments* related to the following ***activity patterns***: *Request for Activity Execution* (activity *a*), *Decision Making* (activity *b*), *Notification* (activities *c* or *e*), and *Approval* (activity *d*). We explain the semantics of these and other activity patterns later.

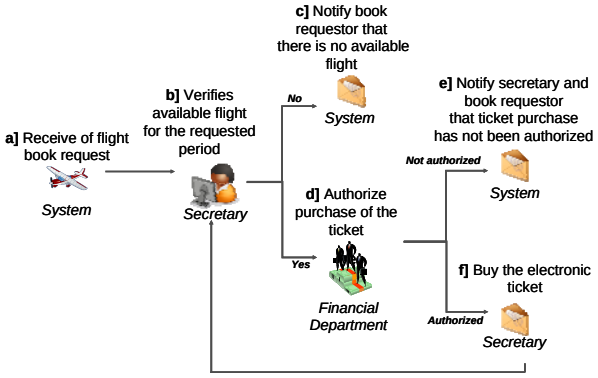

Figure 1 *Travel booking process*

Usually, such process fragments (Medina-Mora, 1992), (Flores, 1998), (zur Muehlen, 2002), (Malone, 2004) are re-implemented in almost every process-oriented application. Although they can be precisely characterized in their semantics, there is only little research relating this kind of process building blocks to patterns. Furthermore, contemporary process modeling tools do neither acknowledge these fragments as patterns nor provide any support for users to define, query, or even reuse activity patterns in a proper way.

While numerous workflow patterns have been introduced related to control flow (Russell, 2006a), data flow (Russell, 2005), resources (Russell, 2004), exception handling (Russell, 2006b), service interaction (Barros, 2005), process change (Weber, 2008a), (Rinderle-Ma, 2008), and application–oriented aspects (Bancroft, 1998), there has been no mapping of activity patterns onto process (meta) models yet and no process modeling tool implements them properly. Furthermore, little or no effort has been devoted on research showing how frequently these patterns are used in practice when designing processes.

1.2 Approach and Contributions

We present results obtained in our ProWAP project. We first introduce a revised version of the seven workflow activity patterns (WAP) we had introduced in earlier work (Thom, 2006). Each of them represents a usual business

function as it can be frequently found within business processes and as discussed in literature as well (Flores, 1998), (Medina-Mora, 1992), (Bancroft, 1998), (zur Muehlen, 2002), (Andrews, 2003), (Malone, 2004). Examples of such activity patterns include *Notification*, *Approval*, *Question-answer*, *Decision*, *Information Request*, and *Request for Activity Execution with / without answer* (which we denote as *Uni-/Bi-directional Performative*, respectively*)*. We consider the block activity concept (WfMC, 2005) as being suitable for representing activity patterns as SESE fragments (Borbrik, 2007); i.e., process fragments with single entry and single exit points. This allows us to encapsulate the well-defined semantics of the patterns and to represent their atomic characteristics; i.e., all steps defined inside a block activity must be completed before the super-ordinated process may continue its execution. By defining activity patterns as SESE fragments we also provide the basis for pattern implementation, pattern reuse within process modeling tools, and pattern composition.

The major contributions of our ProWAP approach as described in this paper can be summarized as follows:

- We present a revised version of seven activity patterns for business process modeling. This pattern set is closer to the vocabulary and abstraction level at which business processes are usually described by domain experts. Generally, multiple activity patterns can be composed in a process model using workflow patterns (e.g., Sequence, AND-Split, AND-Join, XOR-Split). We believe that activity pattern reuse and composition can reduce efforts for process design and modeling.
- Through an empirical study, in which we analyzed 214 real-world process models, the existence of the seven activity patterns has been confirmed. In this context, a process model constitutes a computerized (i.e. formal) representation of either a working procedure or business process that controls the order in which a set of tasks has to be performed (Bardram, 1997). We further observed that in most cases the analyzed process models can be designed based on investigated patterns; i.e., the set of identified activity patterns is necessary as well as sufficient to design the 214 process models, at least at a certain level of granularity. Thereby, a particular activity pattern may occur multiple times within a particular process model as well. Our empirical research is fundamental to evidence the relevance of activity patterns for process modeling and the user assistance they can add to existing BPM tools.
- For selected process categories (e.g., processes with human interventions vs. fully automated processes) we investigate the frequency of co-occurring activity patterns. Our intention is to use the results of this second analysis for developing a BPM tool, which fosters the modeling of business processes based on the reuse of activity patterns. Given some additional information about the kind of process to be designed, for instance, the results of our analysis can be further used by this tool to suggest a ranking of the activity patterns best suited to succeed the last applied pattern.

The identified activity patterns are independent of a concrete process modeling language; i.e., they can be integrated into any process modeling tool. To achieve a precise semantics we have formalized activity patterns using π-calculus. A process model specified in π-calculus can express the dynamic behavior of the process, thus making it possible to verify formal properties of the model like soundness (e.g., absence of deadlocks and livelocks) and model equivalence (Li, 2008a). A formalization of the activity patterns, however, is outside the scope of this article (for details we refer to (Nascimento, 2007)).

Section 2 describes characteristic properties of the seven activity patterns identified and discusses pattern variants in this context. In Section 3 we present the results of an empirical study that we performed in order to investigate the existence of activity patterns in real-world process models. Section 4 discusses related work and Section 5 concludes with a summary and an outlook on future research.

2 ACTIVITY PATTERNS: CHARACTERISTICS AND VARIANTS

We use the term *workflow activity pattern* (WAP) – *activity pattern* for short – to refer to the description of a recurrent business function as it can be frequently found in business processes. Typical examples include task execution requests (similar to the speech-act-theory proposed by (Flores, 1988) and (Medina-Mora, 1992)), notifications, and approvals.

Altogether we have derived a set of seven activity patterns based on an extensive literature study about business process types. These seven activity patterns are as follows: *Approval*, *Question-answer*, *Uni- / Bi-directional Performative*, *Information Request*, *Notification*, and *Decision Making*. For each pattern we provide a *name*, a *description*, an illustrative *example*, a description of the *problem* it addresses, specific *issues*, a couple of *design choices* (determining different pattern variants), a reference to *related patterns*, and remarks regarding *pattern implementation.* Design Choices allow for the parameterization of patterns keeping the number of distinct patterns manageable. They comprise different options applicable in a particular context. For example, in the context of the approval pattern, a particular object (e.g., a business document) has to be approved by one or more organizational roles; this is required before proceeding with the flow of control. 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). Note that these variants were identified based on our observation considering the process models we analysed.

In the following, we informally summarize pattern semantics based on UML activity diagrams (cf. Fig. 2).

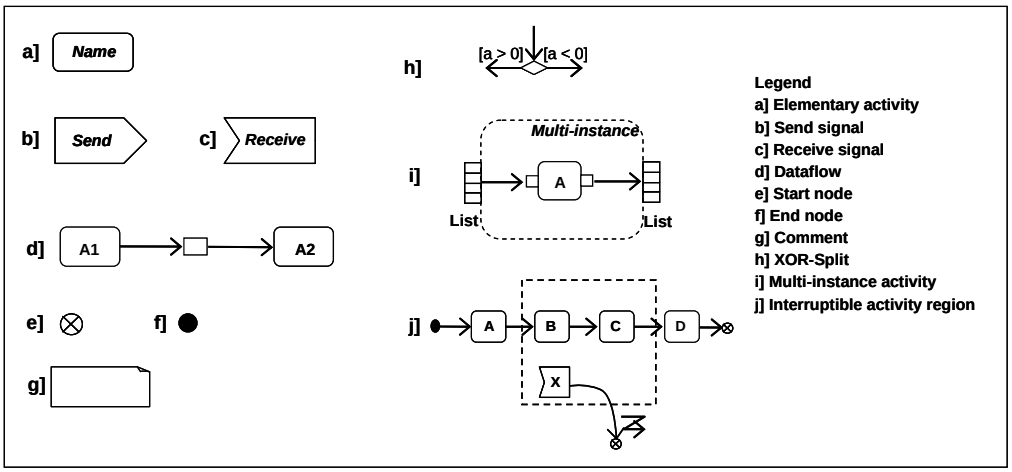

Figure 2 *UML notation (Activity Diagrams) used to informally summarize the activity pattern semantics*

2.1 Pattern Description

In the following we describe the seven activity patterns in a systematic and detailed way. We first consider the APPROVAL pattern, which can be used to express different kinds of approvals in the context of a business process.

Pattern WAP1: APPROVAL

Description: 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 one time (single approval) or multiple times. In the latter variant, it can be either accomplished in sequence (iterative approval) or in parallel (concurrent approval).

Example: In a change management process, for example, a particular change request may have to be concurrently approved by all organizational roles concerned by the change. If one of these roles rejects the change request, it will be not approved.

Problem: During the execution of a business process, object approval by one or multiple organizational roles is required before proceeding with the flow of control.

Issues:

a. The number of organizational roles, who must give their approval, may vary depending on the level of centralization of theauthority present in the respective organization.

b. The approval activity may be performed multiple times in parallel (concurrent approval) or in sequence (iterative approval) according to the number of organizational roles being involved. Concurrent approval is characteristic for *flat organizations,* whereas iterative approval can be often found in *vertical organizations*. In the latter case, the approval activity can be aborted as soon as one role decides for rejection.

c. Final decision can be made manually (i.e., by a user) or automatically according to some rules.

A. Design Choices: Single Approval, Iterative Approval or Multiple Approval:

Major design choice is whether approval shall be done by a single role or by multiple roles either concurrently or iteratively. This, in turn, results in three pattern variants with the following informal semantics:

1. *Single Approval* (cf. Fig. 3): A requestor sends an approval request to exactly one reviewer. This reviewer then performs the revision either resulting in approval or rejection.

2. *Iterative Approval* (cf. Fig. 4): Based on a list of reviewers a requestor sends an approval request for the first reviewer from the list. This reviewer then performs the approval resulting either in approval or rejection. If approved the next reviewer from the list will receive a request for approval, and so on; if one reviewer rejects, all previous approvals (in case they exist) will be cancelled and the overall approval procedure will be aborted. At the end, a final decision – approval or rejection – is made concerning the object under revision.

3. *Concurrent Approval* (cf. Fig. 5): Given a list of reviewers a requestor sends an approval request to all reviewers simultaneously. After all reviewers have performed their approvals the final decision is made.

Related Patterns: Bi-directional Performative (WAP 4) and Decision (WAP 7). Send/Receive and One-to-many Send/Receive (Barros, 2005), Multi-Instance with a a-priori Runtime Knowledge (Russell, 2006a).

Implementation: The approval pattern can be implemented based on the *Send/Receive* pattern (Design choice A(1)) as introduced by (Barros, 2005). Regarding concur-

rent approval (Design Choice A(3)) implementation can be based either on the *Multi-Instance with a priori Runtime Knowledge* pattern or the *One-to-Many Send/Receive* pattern being connected to an XOR-Split (Russell, 2006a). In the latter case, several instances of a task are created and executed in parallel with synchronization being done when all tasks instances are completed.

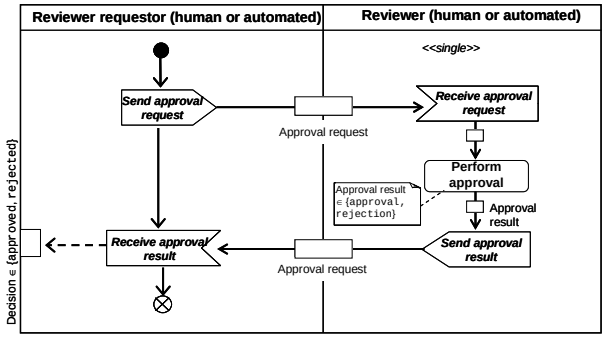

Figure 3 *Single Approval*

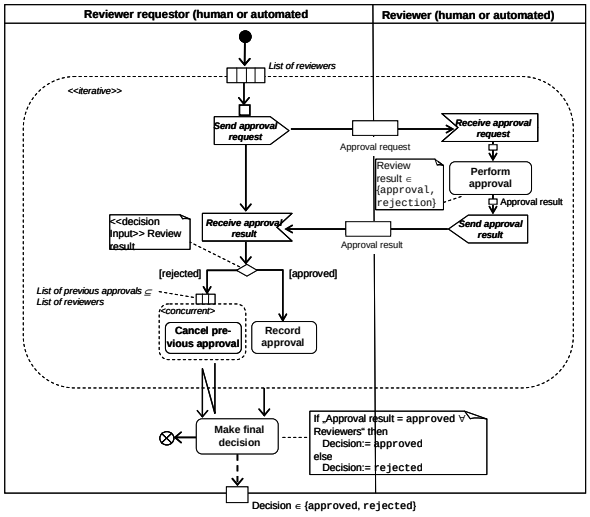

Figure 4 *Iterative Approval*

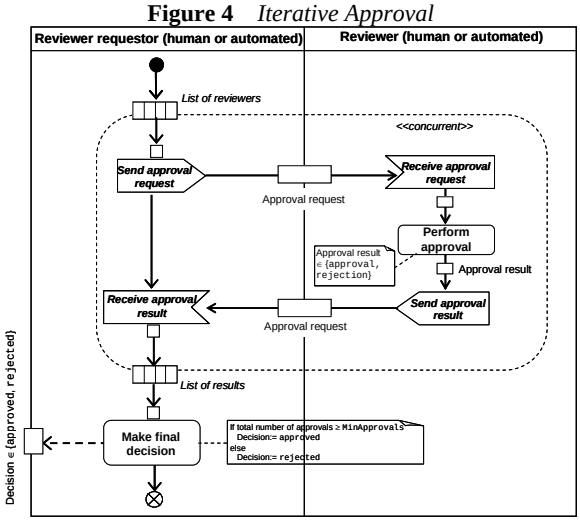

Figure 5 *Concurrent Approval*

The second pattern we discuss is the QUESTION-ANSWER pattern. It can be used to design a question-answer-based interaction where one or more specific participants of the process are chosen to reply to the question.

Pattern WAP2: QUESTION-ANSWER

Description: When performing a process, an actor might have a question before working on the process or on a particular activity. The QUESTION-ANSWER pattern allows to formulate such question, to identify an organizational role who is able to answer it, to send the question to the respective actor filling this role, and to wait for response (*single question-answer*). As generalization, the question can be sent to multiple roles or actors resulting in multiple answers (*multi-question-answer*).

Example: A process for authorizing the construction of a large shopping center close to a protected area requires a license from the government. The process includes several activities such as the creation of the licensing document. In particular, the author of the document may have specific questions concerning governmental rules. Such questions are then forwarded and answered by an organizational role with respective expertise (e.g., a technician from the Licensing division).

Problem: During process execution an actor might have a question regarding the performance of process activities. This requires system support for forwarding questions and answers as well as experts with appropriate abilities or knowledge to answer the questions.

Issues:

a. Based on its description, the question is assigned and forwarded to the role with best expertise in the respective domain (e.g., an actor with specific knowledge about the Java language).
b. The sender of the question waits until the corresponding reply (i.e., the answer to the question) arrives and then continues with process execution.
c. Usually, the question is answered by humans.

B. Design Choices: Single-Question-answer vs. Multi-question-answer

Major design choice is whether the question will be send to one or multiple roles and actors, respectively. This, in turn, results in two pattern variants with the following informal semantics:

1. *Single-Question-Answer* (cf. Fig. 6): Based on a question description an organizational role (i.e., specialist) with expertise in the respective domain is chosen to answer the question. The sender waits until the response arrives and then continues process execution.
2. *Multi-Question-Answer* (cf. Fig. 7): Based on a question description multiple organizational roles (specialists) with expertise in the respective domain are chosen to answer the question. The sender waits until all responses arrive and then continues process execution.

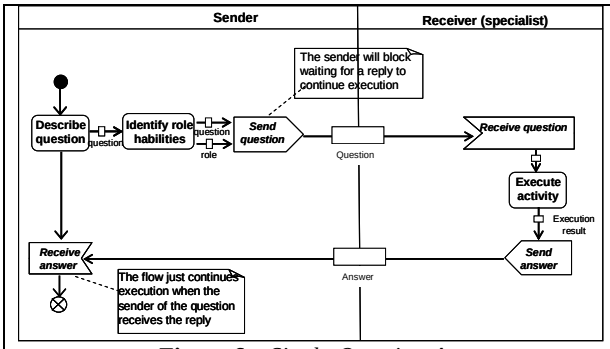

Figure 6 *Single-Question-Answer*

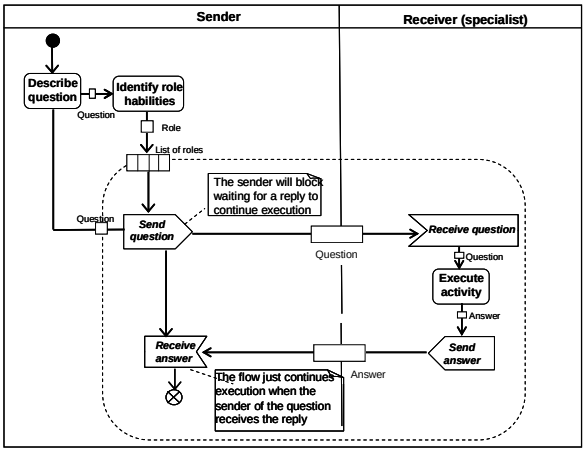

Figure 7 *Multi-Question-Answer*

Related Patterns: *Bi-directional Performative* (WAP 4), *Send/Receive* and *One-to-many* (Barros, 2005), *Multi-Instance* with a priori Runtime Knowledge (Russell, 2006a).

Implementation: The *Single-Question-Answer* pattern variant (Design Choice B(1)) can be implemented based on the *Send/Receive* pattern. Furthermore, the *Multi-Question-Answer* pattern variant (Design Choice B(2)) can be realized either by using the *Multi-Instance with a priori Runtime Knowledge* pattern or the *One-to-Many Send/Receive* pattern.

We now discuss the UNIDIRECTIONAL PERFORMATIVE PATTERN. This pattern represents an unidirectional performative message, i.e., it is used by a sender to request the execution of an activity from a receiver. The sender continues execution immediately after having sent the request (Flores, 1998), (zur Muehlen, 2002).

Pattern WAP3: UNIDIRECTIONAL PERFORMATIVE

Description: A sender requests the execution of a particular activity from a receiver (e.g., a human or a software agent) involved in the process. The sender continues execution of his part of the process immediately after having sent the request.

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 immediately after the cancel activity is requested.

Problem: In the course of a process an activity execution request must be included as process step; the sender of the request must continue execution without waiting for a response.

Issues:

a. A response by the receiver is not required.

b. The process of the sender continues its execution without waiting for the completion of the requested activity.

c. The requested activity either is accomplished by a human or by a software agent.

C. Design Choices: Single-Request vs. Multi-Request

Major design choice is whether the activity execution request shall be sent to one or multiple actors. This results in two pattern variants with following informal semantics:

1. *Single-Request* (cf. Fig. 8): A requestor sends an activity execution request to a receiver and continues process execution without waiting for response.

2. *Multi-Request* (cf. Fig.9): A requestor sends an activity execution request to multiple receivers simultaneously and continues process execution afterwards, i.e., without waiting for any response.

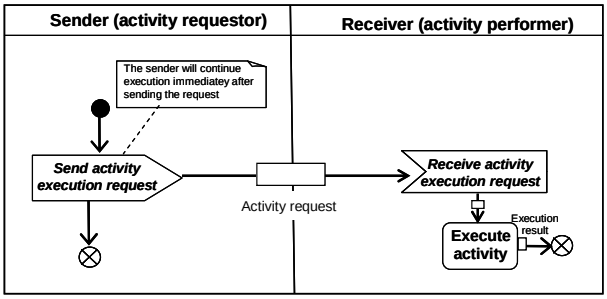

Figure 8 *Single-Request*

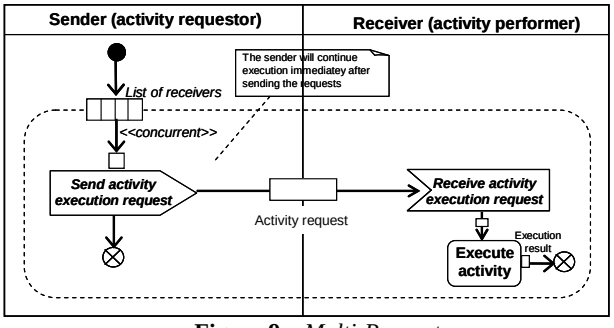

Figure 9 *Multi-Request*

Related Patterns: *Bi-directional Performative* (WAP 4), *Send* and *One-to-Many Send* (Barros, 2005).

Implementation: This pattern can be implemented based on the *Send* pattern (Design Choice C(1)) or based on the *One-to-Many* pattern (Design Choice C(2)) (Barros, 2005).

Next we describe the BI-DIRECTIONAL PERFORMATIVE PATTERN. It represents a bi-directional performative message, i.e., a sender requests the execution of an activity from a particular organizational role. The sender continues execution after this role has notified him about completion of the requested activity (Flores, 1998), (zur Muehlen, 2002).

Pattern WAP4: BI-DIRECTIONAL PERFORMATIVE

Description: 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.

Example: A customer requests 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.

Problem: Within a particular process an activity execution request has to be included as process step (i.e., activity); the sender of this request shall wait with the continuation of his process until the receiver notifies him about completion of the requested activity.

Issues:

a. A response by the receiver (i.e., a notification about performance of the requested activity) is mandatory.

b. The sender process is blocked after sending out the activity execution request. It continues after being notified by the activity performer about the completion of the respective activity.

c. The requested activity can be performed either by a human or by a software agent.

D. Design Choices: Single-Request-Response vs. Multi-Request-Response

Major design choice is whether the activity execution request is sent to one or multiple actors. This results in two pattern variants with the following informal semantics:

1. *Single-Request-Response* (cf. Fig. 10): A requestor sends an activity execution request to one receiver. He waits with continuation of his part of the process until the receiver notifies him about the performance of the requested activity.

2. *Multi-Request-Response* (cf. Fig. 11): A sender sends an activity execution request to multiple receivers simultaneously and continues execution only after having received respective notifications from all performers.

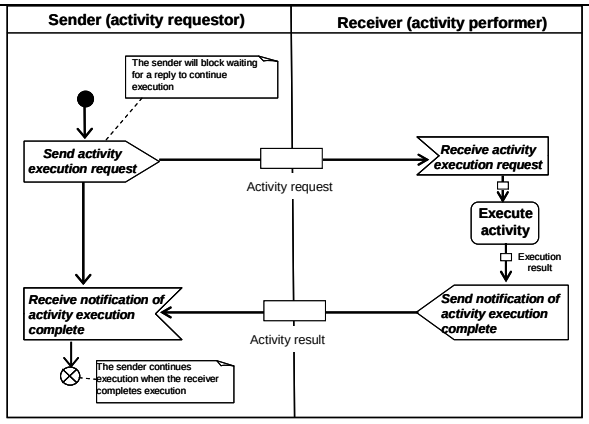

Figure 10 *Single-Request-Response*

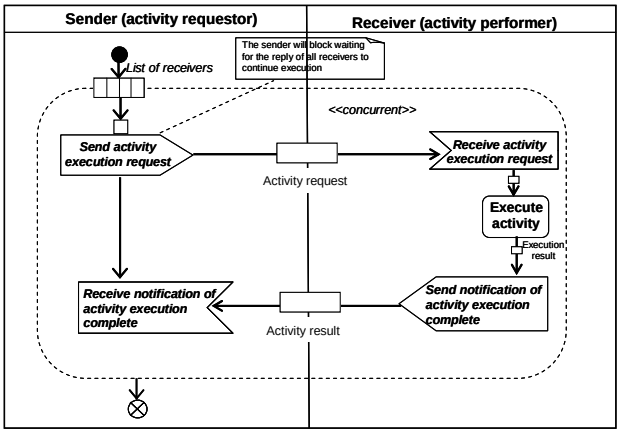

Figure 11 *Multi-Request-Response*

Related Patterns: *Unidirectional Performative* (WAP 3), *Send-Receive* (Barros, 2005), *Multi–Instance with a priori Runtime Knowledge* (Russell, 2006a), *Scatter-gather* (Hohpe, 2004).

Implementation: The *Single-Request-Response* pattern variant (Design Choice D(1)) can be implemented based on the *Send-Receive* pattern. For implementing the *Multi-Request-Response* pattern variant ((Design Choice D(1)) we can use the *One-to-Many Send/Receive* pattern or the *Multi-Instance with a priori Runtime Knowledge* pattern.

The next pattern we present is the NOTIFICATION PATTERN. It comprises a notification activity that either informs actors about the completion of an activity execution or posts news relevant in the context of the modeled process (zur Muehlen, 2002). Regarding the former case, the sender sends a notification informing actors about the result of an executed activity. In our present approach the notification activity is being treated as a self-contained activity.

The description of the NOTIFICATION PATTERN is followed by the one of the INFORMATION REQUEST PATTERN. This pattern is based on an *information request message*, i.e., an actor requests particular information from a process participant. Since a response from the receiver is mandatory, this pattern can be considered as a specialization of the BI-DIRECTIONAL PERFORMATIVE PATTERN (WAP4).

Pattern WAP5: NOTIFICATION

Description: The status or result of an activity execution is communicated to one or more process participants.

Example: When planning a meeting in the context of an engineering process a notification has to be sent to the engineers informing them about meeting details (e.g., location, date, meeting hours, subject).

Problem: During process execution participants have to be informed about the status (e.g., completed, running, waiting) or result (e.g., document approved, rejected) of an activity execution.

Issues:

a. The notification must be sent electronically to one or more process participants.

b. The process does not have to wait for any response of the actors receiving the notification.

c. The notification informs about the status or results of a process activity to be monitored.

E. Design Choices: Single-Notification vs. Multi-Notification:

Major design choice is whether the notification is to be sent to one or multiple actors. This results in two pattern variants with following informal semantics:

1. *Single-Notification* (cf. Fig. 12): A sender sends a notification to a single receiver.
2. *Multi-Notification* (cf. Fig. 13): A sender sends a notification to multiple receivers simultaneously.

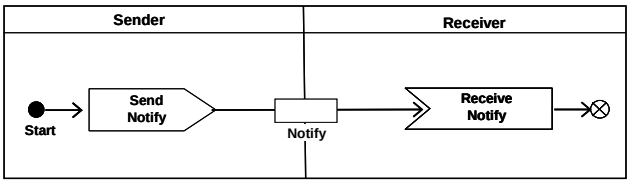

Figure 12 *Single-Notification*

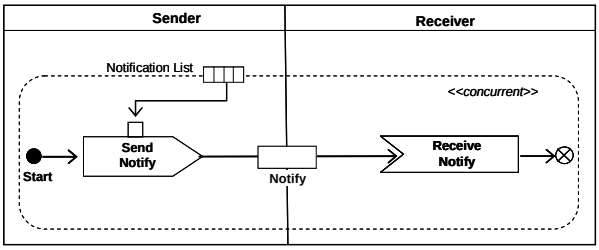

Figure 13 *Multi-Notification*

Related Patterns: *One-Way Send* and *One-to-Many Send* (Barros, 2005).

Implementation: This pattern is supported by several workflow management systems. It can be implemented based on pattern *One-Way-Send* (Design Choice E(1)) or *One-to-Many Send* (Design Choice E(2)).

Pattern WAP6: INFORMATION REQUEST

Description: An actor requests certain information from a process participant. He continues process execution after having received the desired information.

Example: While ordering an airline ticket the customer has to provide personal data (e.g., complete name, address, and credit card number) via a Web browser interface. The process continues afterwards.

Problem: In a process an information requesting activity (e.g., implemented as a form to be filled out) has to be included as explicit process step.

Issues:

a. A response by the receiver is mandatory.

b. The sender continues process execution only after having received the requested information.

c. The requested information is provided by a human or software agent.

F. Design Choices: Single-Information Request vs. Multi-Information Request

Major design choice is whether the information request is sent to one or multiple actors. This results in two pattern variants with following informal semantics:

1. *Single-Information Request* (cf. Fig. 14): A sender sends an information request to a receiver and does not continue process execution before having received the requested information.
2. *Multi-Information Request* (cf. Fig. 15): A sender sends an information request to multiple receivers simultaneously and does not continue process execution before having received responses from all receivers.

Related Patterns: *Send/Receive* (Barros, 2005), *One-to-Many Send/Receive* (Barros, 2005), *Synchronous Transfer* (Mulyar, 2005).

Implementation: This pattern can be implemented based on the *One-Way Send* pattern (Design Choice F(1)) or the *One-to-Many Send/Receive* pattern (Design Choice F(2)).

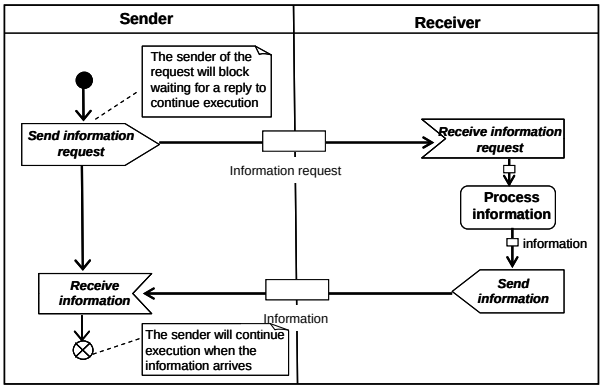

Figure 14 *Single-Information-Request*

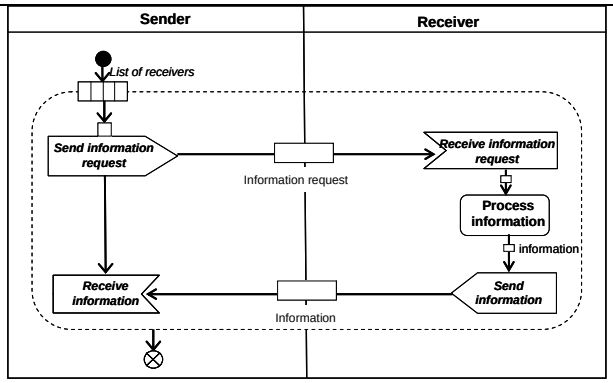

Figure 15 *Multi-Information-Request*

Last but not least, we describe the DECISION PATTERN. It allows to include a decision activity in the flow with connectors to different subsequent execution branches. Exactly those branches are selected for execution whose transition conditions evaluate to true during runtime.

Pattern WAP7: DECISION

Description: During process enactment, the performance of one or multiple activities is requested. Depending on the results of the requested activity executions the process continues execution with one or several branches. More precisely, pattern WAP7 allows to include a decision activity with connectors to different subsequent execution branches (each of them associated with a specific transition condition). Exactly those branches are selected for execution whose transition condition evaluates to true.

Example: To get feedback from a user concerning a particular service the user shall indicate his or her satisfaction degree by giving grades from 0 to 10. Depending on the specified grade the process takes one or several branches based on the conditions (e.g., grade between 0 and 4) associated with them.

Problem: In a process an explicit decision step has to be included. The final decision is made based on the execution result(s) of requested activities.

Issues:

a. A response by the receiver with the result of the activity is required.

b. Based on the response one or several subsequent branches are selected for execution.

c. The final decision is usually made automatically based on the execution result(s) of previous activities.

G. Design Choices: Single-Decision vs. Multi-Decision

Major design choice is whether the final decision is based on the results of one single activity or a set of activities. This leads to two pattern variants with the following informal semantics:

1. *Single-Decision* (cf. Fig. 16): Based on the execution result of an activity one or several succeeding branches are executed.

2. Multi-Decision (cf. Fig. 17): An activity execution request is sent to multiple performers. Based on the results of the activities one or several succeeding branches are executed.

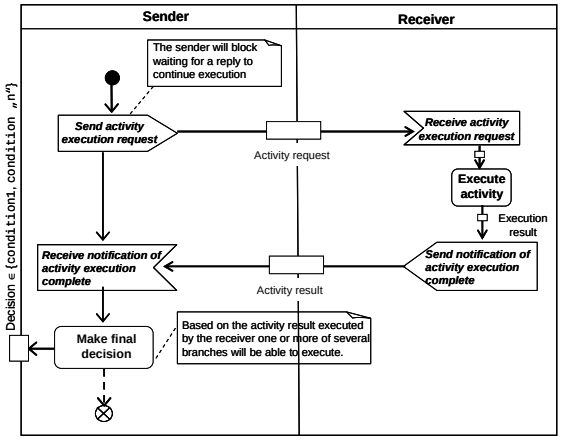

Figure 16 *Single-Decision*

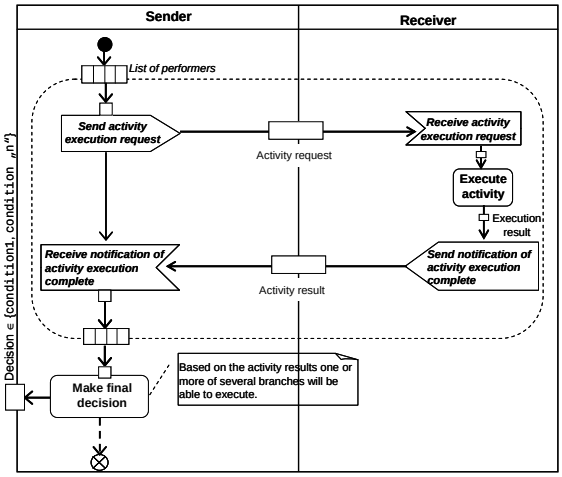

Figure 17 *Multi-Decision*

Related Patterns: *OR-Split* (WfMC, 1999), *OR-Split and Deferred choice* (Russell, 2006a).

Implementation: *WAP7* can be implemented as composition of pattern *WAP4* and an OR-Split. Another implementation option is provided by the Deferred Choice pattern.

2.2 Activity Pattern Categorization

Considering the specific characteristics of the patterns we classify them into two categories (cf. Fig. 18):

- *Activity patterns based on organizational structural aspects.* By tuning or adjusting some structural aspects to the desired performance, the organization gets its final structure (Davis, 1996). Among the most important aspects to be dealt with in the design of an organizational structure, literature emphasizes the degree of centralization on decision-making, the types

of co-ordination mechanisms used (e.g., standardization of abilities to task execution), and the degree of dependencies between activities (Mintzberg, 1995), (Crowston, 1994). This first pattern category therefore comprises exactly those two activity patterns that are related to one or more organizational structural aspects: *Approval* and *Question-answer*.

- *Activity patterns based on recurrent functions.* This category comprises patterns related to general recurrent business functions, i.e., any kind of process model might contain patterns from this category independent of the application domain (e.g., healthcare, automotive engineering) or the kind of organization (e.g., process-oriented, functional, matrix, etc). This category comprises the following five patterns: *Uni-* and *Bi-directional Performative Pattern*, *Information Request Pattern*, *Notification Pattern*, and *Decision Pattern.*

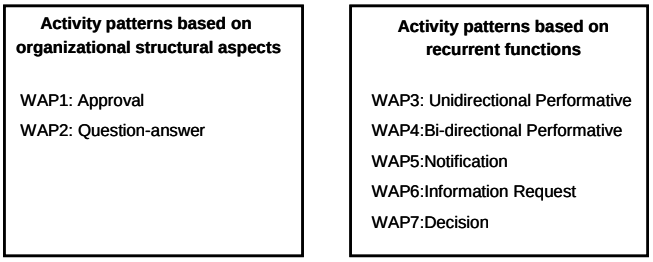

Figure 18 *Classification of activity patterns*

3 EVIDENCING THE EXISTENCE OF ACTIVITY PATTERNS IN REAL-WORLD PROCESS MODELS

We investigate the occurrence of the described activity patterns in real-world applications by presenting results from an empirical study. We analyzed 214 process models and workflow models respectively. Most analyzed models have been created with the Oracle Builder tool or an UML-based process modeling tool. Altogether the considered process models stem from 13 different organizations and are related to different application domains (cf. Table 1).

Two major results can be obtained from our empirical study:

- evidence with high probability that the described activity patterns exist in real-world workflow applications and process-aware information systems respectively;
- evidence that the set of patterns is necessary and sufficient to model all 214 process models analyzed, at least at a certain level of granularity.

3.1 Applied Method

To our best knowledge there exist no mining techniques to extract activity patterns from real-world process models; i.e., contemporary process mining tools like ProM (van der Aalst, 2007) analyze event logs (e.g., execution or change logs) related to process executions and do not extract information related to the semantics and the (internal) logic of process activities (van der Aalst, 2005), (Günther, 2006), (Günther, 2008). Therefore, we perform a manual analysis in order to identify relevant activity patterns as well as their co-occurrences within the 214 process models.

Table 1 *Core characteristics of the process models analyzed in our empirical study*

Size of the company	Kind of decision-making	Application domain	Number analyzed process models
1 small	Decentralized	Management of internal activities	17
1 large	Decentralized	TQM and management of activities	11
6 large	Centralized	TQM; control of software access; document management	133
4 large	We had no access to information about these companies	Help Desk, User feedback; document approval	29
1 large	Centralized	Electronic Change Management	24

For each activity pattern WAP* we calculate its support value S_{WAP*}, which represents the relative frequency of the respective pattern within the set of analyzed process models; i.e., S_{WAP*}:= Freq(WAP*)/214 where Freq(WAP*) denotes the absolute frequency of WAP* within the collection of the analyzed 214 models; for each process model we count at most one occurrence of a particular pattern.

Initially, we manually identify and annotate activity patterns in all analyzed process models. Following this, we determine the absolute frequency of each pattern as described. Obtained results are then divided by the total number of analyzed process models (i.e., 214 models in our case).

3.2 Analyzing Results of our Empirical Study

We present detailed results of our empirical study in which we investigate the frequency with which each activity pattern occurs within the set of 214 process models. This study has been performed in order to verify whether the considered business functions (task execution request, approval, decision, etc.) can be really considered as patterns, and to check whether there is potential for reusing them in the context of process modeling. Figure 19 shows the frequency with which the activity patterns from the two categories introduced above occur. We discuss these results in the following sections.

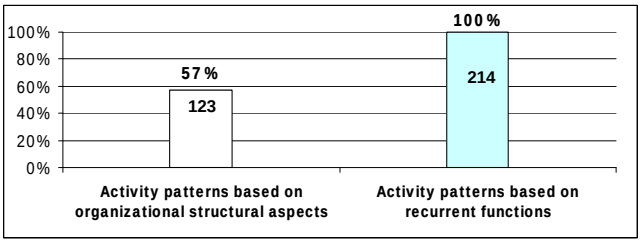

Figure 19 *Results by categories of workflow activity patterns*

3.2.1 Frequency of Organization–based Activity Patterns in real Process Models

This category comprises patterns related to specific organizational structure aspects (i.e., *Approval pattern* (WAP1) and *Question-answer pattern* (WAP2)). In particular, *Approval* can be identified with high frequency within the analyzed set of process models (cf. Fig. 20). First, this can be explained with the high centralization on decision-making we can find in the organizations whose process models we analyze. Usually, such a high degree of centralization implies the use of approval activities. Second, some of the analyzed process models are related to applications explicitly dealing with approvals. The low frequency of the *question-answer pattern* can be partially explained with the fact that most of the analyzed activities are executed by actors with enough knowledge to perform the activity and because the question-answer activities are mainly done informally and not as part of a specified model.

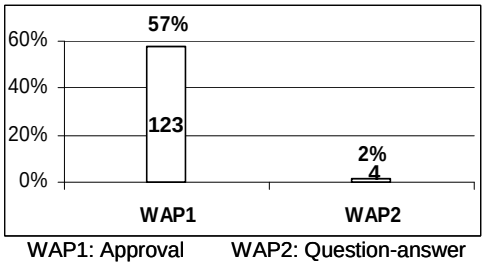

Figure 20 *Frequency of organization-based activity patterns in real process models*

Figure 21 graphically illustrates the frequency of organization–based activity patterns within a set of 63 process models from a highly centralized telecommunication company (see the gray coloured bars in the depicted diagram). It shows that 63% of the process models contain at least one occurrence of the *Approval pattern*. This observation can be explained by the fact that the roles associated with the activities are not high up in the organizational hierarchy, which increases the need for approval. The *Question-answer pattern*, in turn, cannot be identified in this collection of process models. This can be explained by the fact that question-answer activities are mainly done informally and are not explicitly included within process models.

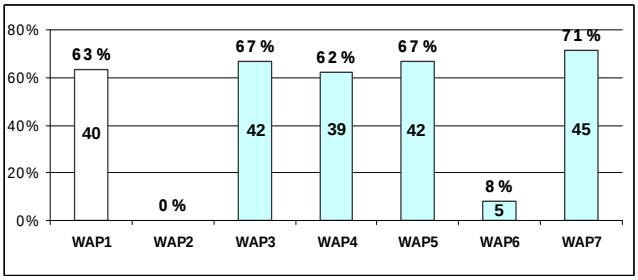

Figure 21 *Frequency of organization-based activity patterns within 63 models from a telecommunication service company*

3.2.2 Frequency of Activity Patterns based on Recurrent Business Functions in Real Process Models

This category contains patterns related to the description or modeling of arbitrary process models: *Uni- / Bi-directional Performative* (WAP3 and WAP4), *Notification* (WAP5), *Information Request* (WAP6), and *Decision* (WAP7). Respective patterns are not dependent on a specific application domain or organizational structure aspect. This explains why we can identify them with high probability in practically all analyzed process models (cf. Fig. 22).

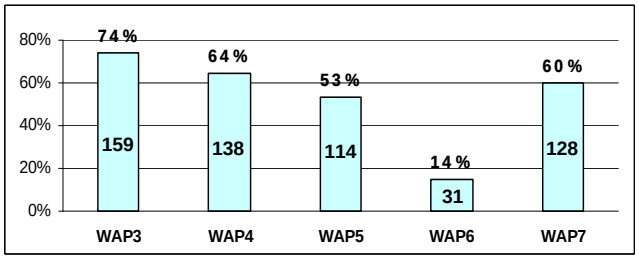

Figure 22 *Frequency of activity patterns based on recurrent business functions in real process models*

Figure 23 illustrates the frequency of activity patterns based on recurrent business functions within a set of 32 workflow models as being executed in a Financial Market Company (gray bars). The diagram shows that most models include at least some of the patterns from this category. Moreover, as the respective organization is highly centralized, the *Approval pattern* can be identified with high probability as well.

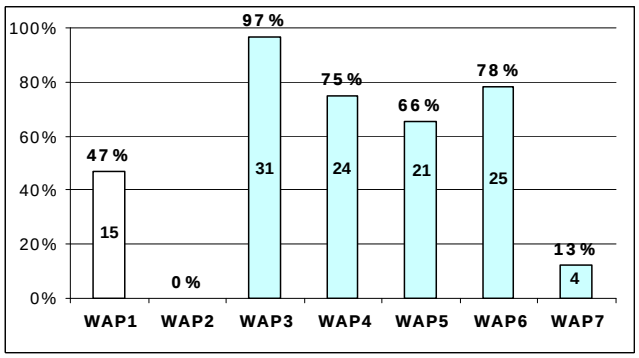

Figure 23 *Frequency of workflow activity patterns in 32 process models of a Financial Market Company*

Results from another interesting case study, which we conducted in the automotive domain, are depicted in Figure 24. In total, we analyzed 24 process models from the field of electronic change management. Due to the very detailed models, in this case study it has become possible to analyze the frequency of each pattern variant (i.e., design choice) as introduced in Section 2.2. For example, Figure 24 shows that Design Choice A(1) of the *Approval Pattern* has higher frequency than the two pattern variants based on design choices A(2) and A(3), respectively. One explanation for this is that the roles associated with the activities are high up in the hierarchy, which reduces the need for iterative

approvals, for example. By contrast, both the *Question-Answer* and the *Information Request* pattern could not be identified. Generally, question-answer activities and information request activities do not frequently occur and – if needed – they are handled informally without being represented in the process model.

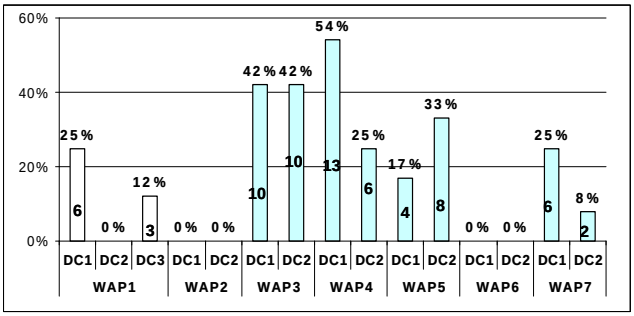

Figure 24 *Frequency of activity patterns in 24 process models from a automotive industry*

3.3 Identifying Co-occurrences of Activity Patterns

For selected process categories, we discuss results of an additional analysis, in which we investigate the frequency of co-occurring activity patterns (Chiao, 2008). To obtain the frequencies for pattern co-occurrences, we analyze the sequences of activity patterns in 154[1] of the 214 studied process models. These results are used in our BPM tool in order to be able to recommend the most suited activity pattern to be used in conjunction with the one applied before. In addition, this tool informs users about the frequency with which pattern pairs were used in the past.

Before performing this analysis we classified the business process models into human–oriented processes (i.e., processes with human interventions during their execution) and fully automated ones (i.e., processes without any human intervention). We verified that certain activity patterns can be found more often in one of the two categories. This analysis has been inspired by a classification provided by Le Clair who distinguishes between *system-* and *human-intensive* business processes (Le Clair, 2007).

When classifying a subset of 154 process models, for which respective information is available, into these two categories, we obtain 123 human-intensive and 31 system-intensive process models. In a next step we evidenced the occurrence of the seven activity patterns with respect to the two categories of process models. Figure 25 shows the support value (i.e., the relative frequency) of the activity patterns in both the *system-* and the *human-intensive* process models. As can be seen, some of the patterns (i.e., *Approval*, *Information Request* and *Question-answer*) do not appear in system-intensive process models at all. Obviously, these patterns are usually related to human activities; i.e., they are executed by an organizational role.

[1] When performing this analysis we had access to only 154 out of the 214 studied process models.

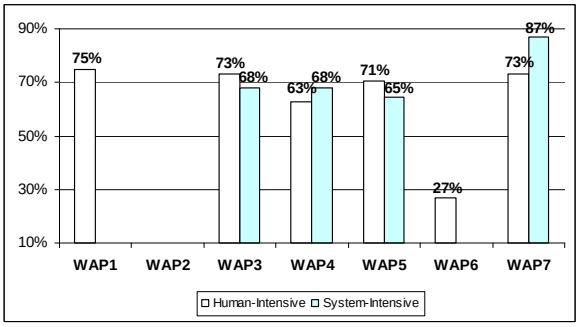

Figure 25 *Frequency of activity patterns in human- and system-intensive process models (Chiao, 2008)*

In another analysis we have searched for frequent and recurrent co-occurrences of activity patterns within process models. Relying on the results of this analysis, we have implemented a process modeling tool, which, among other things, displays to the process designer a ranking of the activity patterns which most frequently follow the pattern the user has applied before during process design. For example, our analysis has shown that patter pair DECISION → NOTIFICATION occurs more often in *system-* than in *human-intensive* processes. Opposed to this, the pattern pair DECISION → APPROVAL occurs more frequently in *human-intensive* process models (see Fig. 26).

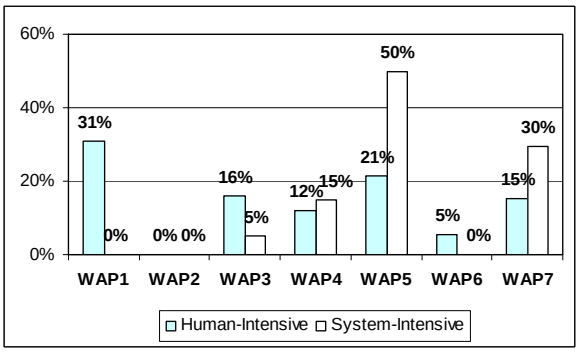

Figure 26 *How often does an activity pattern directly follow the DECISION pattern? (Chiao, 2008)*

3.4 How Representative are Activity Patterns with Respect to Process Modeling?

While some patterns can be identified with the process models solely based on the analysis of the activity descriptions (e.g., *Decision*, *Approval* and *Notification*), other patterns require a more detailed analysis. For instance, activity pattern *Information Request* (WAP6) can be identified in connection with activities for which the user enters information to the system during activity execution (e.g., by filling in data fields in an electronic form). Regarding the patterns *Bi-directional performative* (WAP 4) and *Notification* (WAP 5), both the activity description and its execution result (i.e., mandatory or not to trigger the next activity in the process) have been important for our analysis.

What surprises is the fact that the analyzed process models can be composed out of the considered patterns in combination with specific control flow patterns; i.e., these activity patterns are necessary and allow to design the 214

process models that are subject of our study. Of course, from these empirical findings we must not conclude that the identified set of patterns and pattern variants is sufficient for modeling all business processes we can find in practice. At least, however, we are able to prove that the seven activity patterns occur frequently in different domains and allow to model a variety of business processes. Note that the latter presumes the support of different variants for each pattern as described in Section 2.

It is important to mention that the pattern variants we identified for the different process collections from the considered domains partially depend on the underlying modeling notation. For example, for the analyzed *change management process*, corresponding models are expressed in terms of UML activity diagrams. Here, we are able to identify all described variants of the *Approval* pattern, i.e., single, iterative and concurrent approval (cf. Section 2.1). The latter two variants, however, necessitate the support of the multi-instance workflow pattern, which is the case for the UML notation. We also analyzed processes described with a notation less expressive than UML and not supporting multi-instance activities. Consequently, only single approvals or approvals with an a-priori fixed number of reviewers (modeled as AND-split with a static number of branches) can be identified.

Generally, lack of expressiveness of the given modeling notation might cause a "bias" regarding the results of pattern analysis. When considering workflow patterns, for example, control structures like discriminator or n-out-of-m have been not supported by any of the process modeling notations based on which our 214 process models are described. Obviously, these workflow patterns can be relevant for expressing more specific approval scenarios; e.g., when an approval request is sent to a set of roles and the first role to respond (or a predetermined number of responses) will immediately trigger the continuation of the process without synchronization. Since these advanced workflow patterns are not supported by the notations of the analyzed process models, we are also not able to observe respective variants of the approval pattern; consequently we do not include them in the suggested set of patterns and pattern variants respectively. Exactly for this reason, we have started several case studies to analyze not yet documented business processes from different domains and projects in order to avoid notation dependencies.

Also note that in this paper we have used UML activity diagrams ourselves in order to illustrate the different variants of the patterns identified so far. To avoid dependencies on a particular notation, however, we provide a formalization of our patterns based on π-calculus. Its presentation is outside the scope of this paper.

In summary, at a certain level of abstraction we can show that the business functions behind the seven patterns (e.g., approval, decision, notification) are sufficient to model the 214 process models we analyzed. In future, we will consider the results of the aforementioned case studies as well as the analyses of other process models (e.g. from the MIT process handbook) in order to derive new pattern variants and patterns, respectively.

We can further observe that in each of the analyzed process models, a particular activity pattern may occur zero or multiple times in combination with other patterns. Note that such correlations are quite interesting and also raise new questions to be investigated as part of a future work. One challenging question, for example, is how helpful the identified set of patterns is when being integrated into a process modeling tool. One could think of a BPM tool which relies on a repository comprising such high-level activity patterns. In particular, this would help designers to complete their process model design and to improve model quality. We have presented a first initiative towards this direction in (Thom, 2007a). Figure 27 shows the travel booking process, we presented in Section 1, built up exclusively by combining activity patterns.

Altogether, we consider the contribution of this paper as an important step towards more empirical research in the context of pattern identification and pattern use.

4 RELATED WORK

Patterns were first used by C. Alexander (Alexander, 1977) to describe solutions to recurring problems and best practices in architectural design. Patterns also have a long tradition in computer science. For example, (Gamma, 1995) applied the same concepts to software engineering and described 23 design patterns. Patterns for workflow modeling are still subject of discussion and research (Barros, 2005). 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).

(Russell, 2006a) proposes 43 workflow patterns for describing process behavior (i.e., control flow). Each pattern represents a routing element (e.g., sequential, parallel and conditional routing) which can be used in process modeling. These workflow patterns have been also used for evaluating workflow languages and workflow modeling tools (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 data patterns are based on collections of characteristics that occur repeatedly in different workflow modeling paradigms. In another work, (Russell, 2004) presents resource patterns. Each resource pattern describes a way through which resources can be represented and utilized within processes. A resource is an entity that is capable of doing work (e.g., human, machine).

Recently, (Russell, 2006b) has presented a pattern-based classification framework for characterizing exception handling in workflow management systems. This framework has been used to examine the capabilities of workflow management and BPM systems, and to evaluate process specification as well as process execution languages. As a result, the author emphasizes the limited exception handling support in existing workflow management systems.

Mulyar 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 *service interaction patterns* which allow for web services 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 the workflow patterns provided by Russell and Barros provide a thorough examination of the various perspectives that need to be supported by a process specification language and process modeling 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: e.g., 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). 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, process change patterns have been introduced by (Weber, 2007) (Weber, 2009). A formalization of these change patterns is given in (Rinderle-Ma, 2008). (Weber, 2008b) further show how change patterns have to be applied to foster the refactoring of large business process models.

PICTURE proposes a set of 37 domain specific process building blocks (Becker, 2007). More precisely, these building blocks are used by end users in Public Administrations to capture the process landscape. The building blocks are currently being evaluated in the area of Public Administrations (and are also specific to this domain).

(Mutschler, 2007) presents value-based evaluation patterns for modeling and analyzing cost as well as impact factors in process-aware information systems.

All these approaches have significantly contributed to the improvement of process design, exception handling, and process change. However, a set of activity patterns representing recurrent business functions in workflow models is still missing. In addition, most of the aforementioned approaches discuss the implementation of patterns in existing process modeling tools. They do not focus on how often these patterns are used for process modeling.

5 SUMMARY AND OUTLOOK

This paper has reported on activity patterns for designing process models. Each of these patterns is based on a recurrent business function and process fragment, respectively (e.g., task execution request, notification activity, approval) as they can be frequently found in business processes.

To measure the frequency with which each activity pattern occurs in real process models we performed an empirical study. In this study we analyzed 214 process models from different application domains. This analysis was accomplished in order to verify whether candidate process fragments may be considered as patterns with high probability for reuse. Our empirical study has shown that the detected patterns are well suited for defining both business processes and workflows from a variety of application domains. In future we want to empirically validate whether the activity patterns contribute to reduce real efforts for designing process models: (i.e., to increase productivity during process design time (Thom, 2007a).

Main advantages of our work can be summarized as follows: (a) sufficiency and necessity of the activity patterns for process design has been evidenced at least with respect to process models similar to the ones we analyzed; (b) activity patterns are tool-independent which makes it easy to adopt them to any BPM suite; and (c) activity patterns can be also useful to deal with other fundamental tasks in process management; e.g., to accomplish dynamic process changes (Reichert, 1997), (Reichert, 1998) (Reichert, 2003) at a higher level of abstraction when compared to contemporary approaches (Rinderle, 2004), (Müller, 2008).

As future work we intent to perform further analyzes considering process models from additional application domains (e.g., healthcare). The intention behind these additional analyses is to increase the support value of each pattern as well as to identify frequent sequences of related or combined patterns. We also intend to identify variants of each pattern concerning specific application domains. For example, we want to figure out what kind of approvals occurs most frequently in the healthcare and the automotive domain (Lenz, 2007), (Müller, 2006).

We also want to continue with the development of a BPM tool which we have prototypically implemented in the ProWAP project (Thom, 2008a). This tool fosters the modeling of business processes based on the reuse of activity patterns. In principle, the basic concepts behind this tool can be added as extensions to existing BPM tools as well; e.g., Intalio (Intalio, 2006), Aris Toolset (2007), and ADEPT2 Process Composer (Reichert, 2005). Furthermore, configuration and visualization support for business process models (Bobrik, 2006) (Bobrik, 2007) (Hallerbach, 2008) can be improved utilizing the semantics of the used activity patterns. 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.

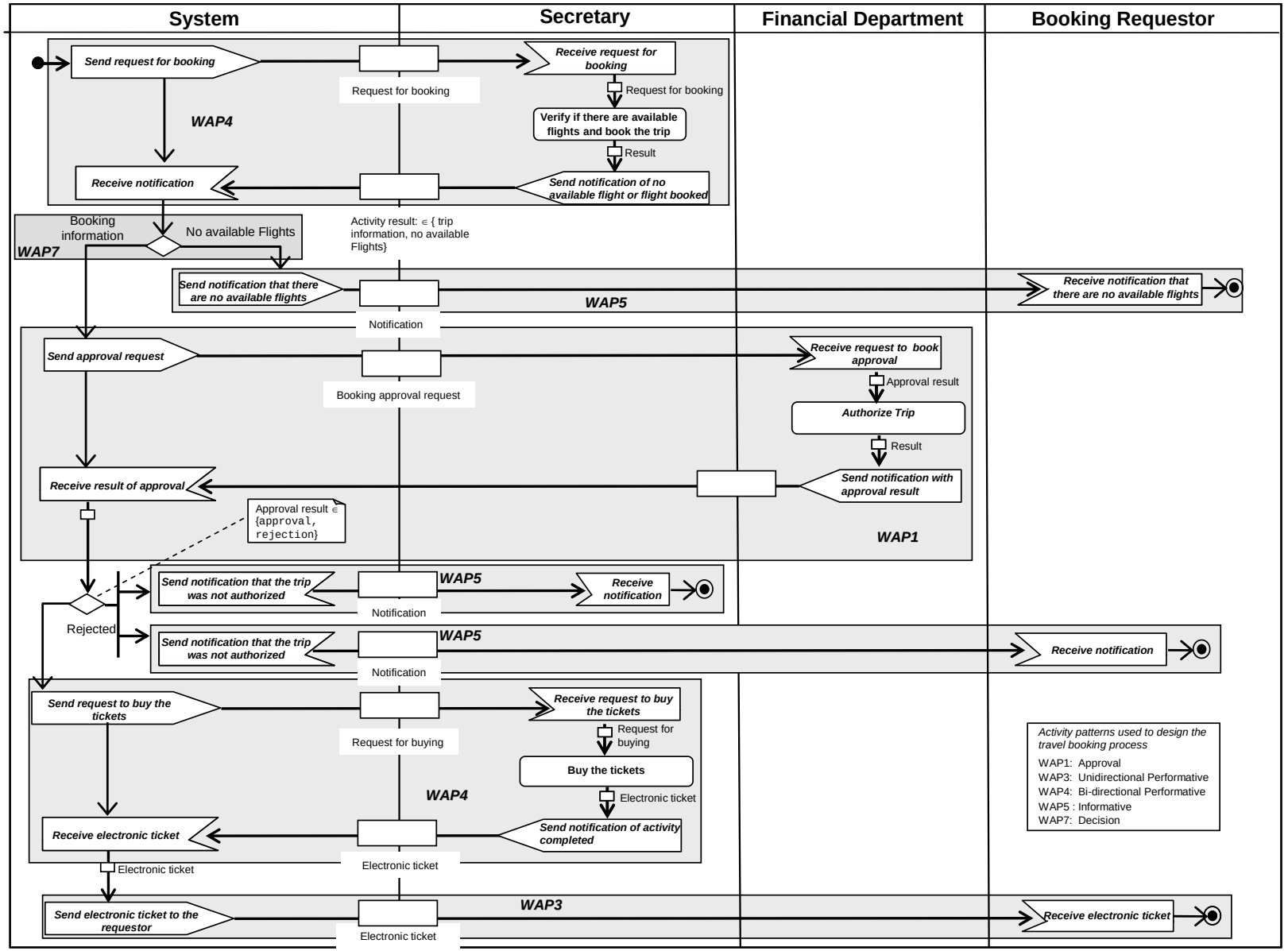

Figure 27 *A real travel booking process built up exclusively from the combination of workflow activity patterns*

REFERENCES

Alexander, C., Ishikawa, S. and Silverstein, M. (1977) A Pattern Language, Oxford University Press, New York.

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

Andrews, T. et al. (2003) Business Execution Language for Web Services, V. 1.1.

Bancroft, N., Henning, S. and Sprengel, A. (1998) Implementing SAP R/3. 2nd ed. Greenwich, Conn.: Manning, XXIV, 310 p.

Bardram, J. E. (1997) Plans as Situated Actions: An Activity Theory Approach to Workflow Systems. In: Proc. ECSCW'97 Conference, Lancaster, UK.

Barros, A., Dumas, M. and ter Hofstede A. (2005) Service Interaction Patterns. In: Proc. 3rd Int'l Conf. 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.

Bobrik, R., Bauer, T. and Reichert, M. (2006) Proviado – personalized and configurable visualizations of business processes. In: Proc. 7th Int'l Conf. on Electronic Commerce and Web Technologies (EC-WEB'06), Krakow, Poland, LNCS 4082, pp. 61-71.

Bobrik, R., Reichert, M and Bauer, T. (2007). View-based process visualization. In: Proc. of the 5th Int'l Conf. on Business Process Management (BPM'07), Brisbane, Austalia, LNCS 4714, pp. 88-95.

Chiao, C., Thom, L. H., Iochpe, C. and Reichert, M. (2008) Verifying Existence, Completeness and Sequences of Workflow Activity Patterns in Real Process Models. In: IV Brazilian Symp. of Information Systems (SBSI), Rio de Janeiro, Brazil.

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

Dadam, P., 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). A Practical Guide to Organization Design. Boston: Crisp Publications.

Flores, F. (1988) Computer Systems and the Design of Organizational Interaction. In: ACM Transactions on Office Information Systems, 6(2):153–172.

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

Günther, C.W., Rinderle, S., Reichert, M. and van der Aalst, W.M.P. (2006) Change Mining in Adaptive Process Management Systems. In: Proc. of the 14th Int'l Conf. on Cooperative Information Systems (CoopIS'06), Montpellier, France, LNCS 4275, pp. 309-326.

Günther, C.W., Rinderle, S., Reichert, M. and van der Aalst, W.M.P. van der, Recker, J. (2008) Using Process Mining to Learn from Process Changes in Evolutionary Systems. In: Int'l 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), Barcelona, Spain, pp. 154-161.

Hohpe, G. and Woolf, B. (2004) Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley.

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

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

Khalaf, R., Keller, A. and Leymann, F. (2006): Business processes for Web Services: Principles and applications. IBM Systems Journal, 45(2):425-446

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. Data and Knowledge Engineering , 61(1): 39-58.

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

Li, C., Reichert, M. and Wombacher, A. (2008b) 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

Malone, T.W., Crownston, K. and Herman, G.A. (2004) Organizing Business Knowledge: The MIT Process Handbook.

Medina-Mora, R. (1992) The action workflow approach to workflow management technology. In: Proc. CSCW'92. pp. 281-288.

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: 4th Int'l Conf. on Business Process Management (BPM'06), Vienna, Austria, 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. In: 20th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'08), Montpellier, France, 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 Rinderle, S. (2007). Analyzing the Dynamic Cost Factors of Process-aware Information Systems: A Model-based Approach. In Proceedings of the 19th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'07), Trondheim, Norway, LNCS 4495, pp. 589-603.

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

Mutschler, B., Weber, B. and Reichert, M. (2008b) Workflow Management versus Case Handling: Results from a Controlled Software Experiment. In: 23rd Annual ACM Symposium on Applied Computing (SAC'08), Fortaleza, Ceará, Brazil. ACM Press, pp. 82-89.

Nascimento, Gleison Samuel do. (2007) Notação formal para representação de processos de negócio (Formal Notation of business process definition). Technical report, 2007, UFRGS, Porto Alegre, Brazil.

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

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

Reichert, M., Bauer, T. and Dadam, P. (1999): Enterprise-wide and Cross-enterprise Workflow-Management: Challenges and Research Issues for Adaptive Workflows. In: Proc. Workshop on Enterprise-wide and Cross-enterprise Workflow Management, CEUR Workshop Proceedings, Vol. 24, pp. 56-64.

Reichert, M., Dadam, P., and Bauer, T. (2003). Dealing with Forward and Backward Jumps in Workflow Management Systems. Int'l Journal Software and Systems Modeling (SoSyM), 2(1):37-58.

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.

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, S. and Reichert, M. (2006) Data-Driven Process Control and Exception Handling in Process Management Systems. Proc. 18th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'06), Luxembourg, LNCS 4001, pp. 273-287.

Rinderle-Ma, S., Reichert, M. and Weber, B. (2008) On the Formal Semantics of Change Patterns in Process-aware Information Systems. In: Proc. 27th Int'l Conf. on Conceptual Modeling (ER'08), October 2008, Barcelona, LNCS 5231, pp. 279-293.

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), 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: Proc. CAiSE'06, pp. 288-302.

Thom, L.H., Iochpe, C., Amaral, V. and Viero, D. (2006) Towards Workflow Block Activity Patterns for Reuse in Workflow Design. In: WfMC Workflow Handbook 2006. pp. 249-260.

Thom, L.H., Lau, J. M., Iochpe, C. and Mendling, J. (2007a) Extending Business Process Modeling Tools With workflow pattern Reuse. In: 9th Int'l Conf. on Enterprise Information Systems (ICEIS'07), Funchal, Portugal, pp. 447-452

Thom, L.H., Iochpe, C. and Reichert, M. (2007b) Workflow Patterns for Business Process Modeling. In: 8th Int'lWorkshop on Business Process Modeling, Development, and Support (BPMDS'07), CAiSE'07 workshop, Trondheim, Norway.

Thom, L.H., Reichert, M., Chiao, C. and Iochpe, C. (2008) Applying Activity Patterns for Developing an Intelligent Process Modeling Tool. In: 10th Int'l Conf. on Enterprise Information Systems (ICEIS'08), Barcelona, Spain, pp. 112-119.

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

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

van der Aalst, W.M.P., van Dongen, B., Günther, C.W., Mans, R., Alves de Medeiros, A., Rozinat, A., Rubin, V., Song, M., Verbeek, H. and Weijters A. (2007) ProM 4.0: Comprehensive Support for Real Process Analysis. In: Proc. 28th Int'l Conf. on Applications and Theory of Petri Nets and Other Models of Concurrency, Siedcle, Poland, LNCS 4546, pp. 484-494

Weber, B., Reichert, M., Rinderle, S. and Wild, W. (2006) Towards a Framework for the Agile Mining of Business Processes. In: BPM'05 Workshop Proceedings, LNCS 3812, pp. 191-202.

Weber, B., Rinderle, S. and Reichert, M. (2007) Change Patterns and Change Support Features in Process-Aware Information Systems. In: Proc. 11th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'07), Trondheim, Norway, 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 , 66(3):438-466.

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

Weber, Barbara and Reichert, Manfred and Wild, Werner and Rinderle-Ma, Stefanie (2009) Providing Integrated Life Cycle Support in Process-Aware Information Systems. Int'l Journal of Cooperative Information Systems (IJCIS), 18 (1) (to appear).

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

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. BPM'06, Vienna, pp. 161-176.

Workflow Management Coalition (2005). Process Definition Interface: XML Process Definition Language. Doc. Number: WFMC-TC-1025.

Workflow Management Coalition. (1999) Terminology & Glossary. Bruxelas, 65p.

zur Muehlen, M. (2002) Workflow-based process controlling: foundations, design, and application of workflow-driven process information systems. Logos Verlang Berlin: Berlin. 299 p.

ACKNOWLEDGEMENT

We are grateful for the Coordination for the Improvement of Graduated students (CAPES), the Institute of Databases and Information Systems of the University of Ulm (Germany) and the Informatics Institute of Federal University of Rio Grande do Sul (Brazil).