

Dynamic Routing as paradigm for decentralized flexible process management

Alistair Barros
SAP Research Centre
Brisbane, Australia
alistair.barros@sap.com

Gero Decker*
Hasso-Plattner-Institute
Potsdam, Germany
gero.decker@hpi.uni-potsdam.de

Abstract

Since companies face continuously changing business processes, centralized process repositories and execution engines have been integrated into the heart of enterprise systems. These systems are then able to support repetitive processes. In order to extend the reach of process orientation into more collaborative settings, employees must be given the possibility to adapt and refine the processes to their individual needs. Personalized process templates are applied to tasks and running process instances are subject to massive changes by the participants. These flexible processes require for process management infrastructures of a more peer to peer style. We argue that the concept of Dynamic Routing is a valuable paradigm for supporting highly flexible processes. We identified three patterns for Dynamic Routing and sketch future enterprise system architectures implementing these. We present two case studies where dynamically routed interactions are at the center of attention.

1. Introduction

Process-orientation has served as paradigm for enterprise systems for many years now and implementations in companies help to adapt to changing business practices. Long-running collaborative business processes are supported and a shift from merely focusing on static system structure to describing behavioral aspects of system components can be observed. In addition to intra-organizational process models (workflows) global interaction models have been introduced as means to specify business to business (B2B) processes from a global point of view. However, like it was the case in the beginning of workflows, the demand for allowing flexible conversations increases. Especially, in highly knowledge-intensive ad-hoc settings, current process management systems fail to support flexible collaboration

scenarios.

We argue that given a usable environment users could participate in defining (simple) collaborative processes and changing these processes at runtime. E.g. users should be able to set up templates for performing a set of tasks requiring collaboration with other participants and to then apply the templates when particular tasks are to be performed.

Bringing processes close to the users requires for shifting away from a purely top-down-approach of traditional process management systems, where processes are centrally defined, executed and monitored. In an additional bottom-up-approach where users can define and change limited parts of processes and thus have their personalized version of process parts, best practices might evolve faster within the organization. Having (simple) process management in every users' workplace results in a more peer-to-peer fashion of process management where employees share processes and new organization-wide processes are found using consensus among the employees.

We present Dynamic Routing as paradigm for a new generation of enterprise systems where processes are partially managed centrally and partially in a decentralized manner. At the center of dynamic routing we have a case or work package that is routed among different participants within one organization or located in different organizations. (Parts of) process descriptions are sent together with the messages exchanged and special infrastructure for every user takes care of processing these descriptions and allow for modifications by the user.

This paper is a first starting point of discussion about decentralized flexible process management systems and will give an insight into the ideas of Dynamic Routing as follows: The next section clarifies the relationship between Dynamic Routing and process execution, before architectural implications are presented in the third section. Three patterns for Dynamic Routing are given in section 4, before two case studies are introduced. Finally, related work is discussed and a conclusion is drawn.

*Research conducted while with SAP Research Centre Brisbane

2. Dynamic Routing vs. process execution

Routing originally stems from computer networks where a packet or message is to be routed from a source node to a target node via several intermediate nodes. In the case of static routing, the routing order, i.e. the path from source to target, is created at build-time and all the nodes have to adhere to it. In the case of dynamic routing, the routing order might be changed while having already started to route the message. A reason for that could be that e.g. a certain node is not available and an alternative path has to be taken.

Business processes could be seen in a similar way: There a process instance (case) is created and handed over from actor to actor until processing the case is completed. If the given process model is not suitable for optimally processing the case an alternative process model might be introduced by one of the actors. However, more sophisticated control flow than just simple paths is needed for business processes.

An interesting aspect about routing is that it requires for routing capabilities at each of the intermediary nodes. Using Dynamic Routing as paradigm for business processes thus implies bringing process execution capabilities to the different actors as well as capabilities to change the process when needed.

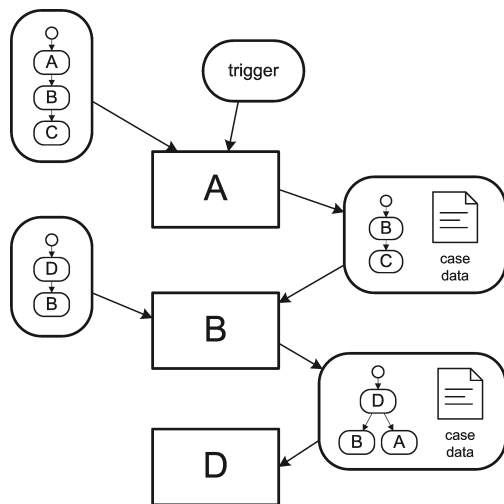


Figure 1. Dynamic routing for business processes

Figure 1 illustrates dynamic routing for business processes. An event triggers actor A to create a process instance. A has a suitable process model at hand where first A does an activity then actor B and finally actor C. After A has finished his task he computes the process description for the remaining tasks and hands it over to B together with the case data for the process instance. The process description now contains two tasks one of which has to be performed

by B before C performs the final task. However, after having finished his task B judges that the case requires actor D to perform a task before it is handed back to B and A who then both get involved again. Since such a setting where B includes D to perform a task (D might e.g. be the secretary of B or a consulting company giving advice to B) occurs very often, B has created a process template that he can apply whenever necessary. In this particular case he did not completely stick to this template but also included A.

Changing the process often consists of adding constraints and refining it, i.e. introducing new subtasks or interactions. This refinement might follow a similar pattern for a certain set of tasks. Thus, refinement templates are maintained for an actor where he can choose from for a particular case. This implies that the process environment for the actor does not only include execution functionality but also a repository for these templates as well as some kind of application that allows for selecting a template, modifying it or doing other ad-hoc changes for a particular case.

3. Architectural implications

Traditionally, process-oriented enterprise systems have central process management systems that include a repository of process models, execute process instances, allow for monitoring of process instances and also include analysis functionality for process models or completed process instances.

Figure 2 uses the FMC block diagram notation (see [5]) and depicts an enterprise system following the Dynamic Routing paradigm. In addition to the traditional process management systems, lightweight process engines become an essential part. These engines interpret routing orders and are able to create the routing orders for follow-up participants. Lightweight indicates in this context that the engines lack the sophistication of large-scale process engines, e.g. complex recovery and scheduling mechanisms. Furthermore, they are connected to the applications that are used to work on the case. Every user also owns a process repository and editing facilities are provided so he can create and modify process templates.

The diagram depicts that Dynamic Routing can be considered for both intra-organizational processes (workflows) or for inter-organizational processes. In both cases the central process management systems provide monitoring functionality and ensure that constraints that hold for the organizational unit are enforced. The central process repository contains process models that must be followed by every participant in the organizational unit. These models include explicit flexibility points where Dynamic Routing then kicks in during execution. The central repository also contains personal processes that proved useful and that can be downloaded by every participant. Sharing personal processes

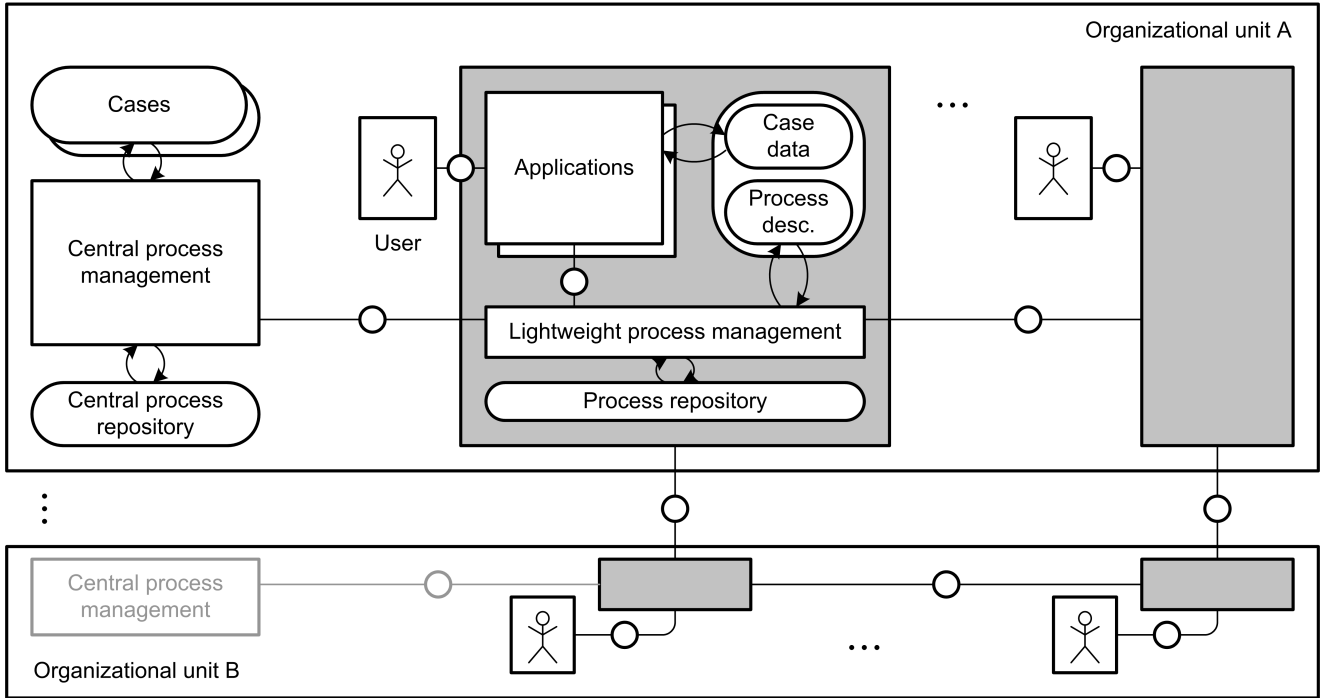


Figure 2. Architecture for decentralized flexible process management

also takes place in a peer to peer manner. E.g. when a new employee enters a company he gets a set of personal processes from his colleagues he can then adapt to best suit his needs.

4. Patterns for Dynamic Routing

In previous work we have introduced the Service Interaction Patterns ([1]). These patterns describe common interaction scenarios between two or more participants. Especially the “Dynamic Routing” pattern has received much attention and paved the way for truly conversational services.

In this section we want to refine this pattern using more precise descriptions. These new patterns can then be used to assess enterprise systems and languages for describing dynamic routing orders. For every pattern we give a short description, an example, issues / design choices and related patterns.

4.1. Late binding of participants

Description. The decision about which participants are going to be recipients later on can be refined or an already determined recipient can be re-bound.

Example. A buyer requests an investment proposal from an investor. The investor sets up the proposal and sends it back

to the buyer. Based on an option the buyer chooses within the proposal he might determine that a tax accountant who will come into play later on must be a CPA (Certified Practising Accountant) with experience in agriculture. The buyer sends his opinion on the proposal back to the investor and lets him know about his accountant requirements. The investor selects a particular accountant meeting these requirements who will participate in subsequent interactions with the investor and the buyer. Therefore, binding takes place in three steps: The abstract role “tax accountant” is present in the collaboration setting from the start (at design-time). The buyer refines this role to a more concrete role “CPA for agriculture”. Finally, the investor selects a particular accountant.

Issues / design choices.

- Lists of potential recipients or references to groups of potential recipients might be given, constraining the decision later on. Lists could mean explicitly naming participants whereas a reference could be e.g. a mailing list, an organizational role or a capability description.
- The order within participant lists might give a hint for the decision. Thus, reordering this list might influence the decision later on.
- Modifications to lists could be limited to inserting,

deleting or replacing participants. Inserting or replacing participants might be subject to restrictions e.g. by means of an additional group reference (e.g. “trusted partners”).

- Re-binding recipients might lead to deadlocks (see below).
- Participants might be required to accept getting involved in the process. This corresponds to the “Distribution by Offer - Single Resource” pattern. Several participants might be asked in parallel or one participant could be asked at a time (which would rather correspond to the “Contingent Request” pattern).

Related patterns. Deferred Allocation, Role-based Allocation, Capability-based Allocation, Distribution by Offer - Single Resource, Contingent Request

Deadlocks. Deadlocks can occur if two participants are working on the same case in parallel and one of them re-binds a future participant. Figure 3 illustrates an example. An initial process model prescribes that first A then B and

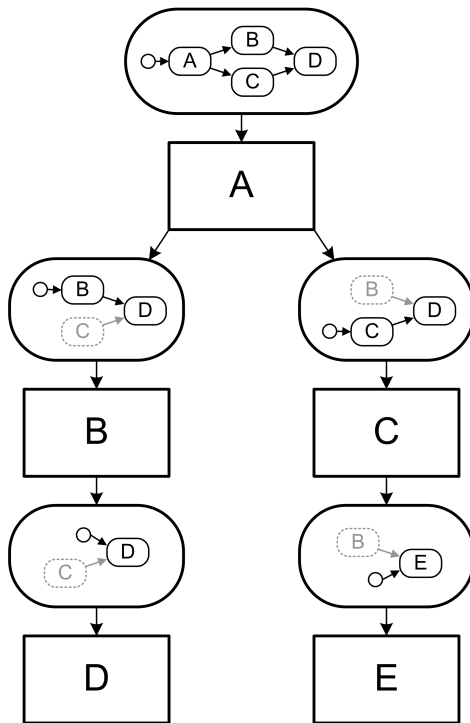


Figure 3. Re-binding of recipients leading to a deadlock

C in parallel and finally D are involved. After A has completed his task is passes the corresponding process descriptions over to B and C. B in turn performs his task and hands

the resulting process description over to D. D is told to wait for the results of C before he is allowed to start his task. In the meantime C decides to perform a re-binding: he replaces D by E. Therefore, C passes the resulting process description on to E. E is told to wait for the results of C before proceeding. Now, both D and E wait for results before continuing. Thus, the re-binding has finally led to a deadlock.

4.2. Order refinement

Description. The order of participants in a routing order is refined. Especially in the case of parallel and interleaved parallel routing additional ordering constraints are added.

Example. A project proposal initiated by a project coordinator is required to be passed through its work-package coordinators in any order, one at a time. For each route, all coordinators get copies of the document, however only one, i.e. the first expressing interest, is allowed to do the update. A coordinator updates the document and makes it available for the next coordinators’ “read only” copy, out of which one gets “write” access. After an update, it might be required that a particular coordinator is the next one to edit the document because of an interdependency with another project.

Issues / design choices.

- The participant might be restricted to do refinements in a limited area in the routing order (e.g. only in the current interleaved parallel routing order).
- Pull-oriented routing might be used to refine a routing order. I.e. all participants that could be the next recipients in the routing order are notified. The participants then respond as soon as they are ready to process the case.

Related patterns. Parallel Split, Interleaved Parallel Routing

4.3. Structural changes

Description. A participant can add or delete tasks and interactions to/from the routing order.

Example. In a negotiation process between a buyer and a real estate agent the buyer might decide to include an independent expert to check the offered property. The expert would not only interact with the buyer but would also interact with the real estate agent.

Issues / design choices.

- The update might be restricted to certain regions of the routing order. Special care has to be taken to avoid deadlocks.
- The participants involved in inserted interactions might be restricted to a specific group (e.g. only interaction with a specified set of experts might be allowed).
- The information passed in additional interactions might be restricted, i.e. only a limited view of the case is passed.

Related patterns. None

5. Case study: Task delegation

Task delegation is a central concept of collaborative work. When a task is assigned to an employee he might decide to only perform a part of the task by himself. For the rest he formulates subtasks and delegates them to other employees. In most cases he would first ask people if they can or want to perform the subtask. If nobody accepts to do so he might ask another group of people or just perform the task by himself. In order to abbreviate the process, the employee might tell the performer of a subtask to directly forward the results to somebody else. He might as well give hints about who the performer of the subtask could ask if again help or delegation is needed.

Some tasks that are assigned to employees might be similar. They require similar division into subtasks and similar selection of potential helpers. Therefore, a number of templates could be created for tasks. The employee then selects a template for his particular task.

All the Dynamic Routing patterns appear in the case of task delegation. Participants might bind another participant at different points in the process or might re-bind participants that have already been selected. Choices or repetitions might appear in process descriptions the conditions of which might need to be overwritten at some point in time. Finally, new process templates come into play every time a task is delegated and during execution short cuts might be taken (i.e. subtasks are skipped) or additional subtasks are being introduced.

Tasks are often formulated using emails. Therefore, simple process descriptions can be attached to emails that determine the rest of the life cycle of a task. A light-weight process engine can be implemented as extension to the email client filtering out process descriptions and taking care of creating new process descriptions for follow-up emails. A small process repository is introduced into the email client and process templates from this repository can be distributed to colleagues.

6. Case study: Legal case preparation

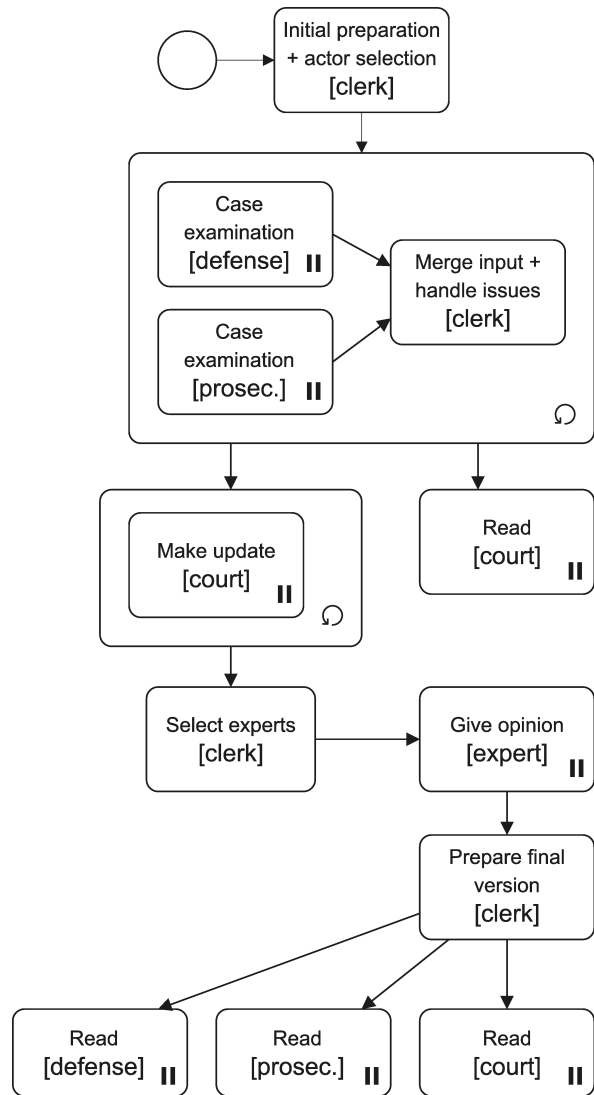


Figure 4. Legal case preparation

A legal case preparation service is utilized by law courts to reduce the number of hearing re-schedules. This is a costly problem for the courts and defers justice for litigants due to insufficient information to embark on hearings, decreed by judges typically within the first moments of a hearing. As part of improved preparation, the clerk of the court examines a scheduled hearing, obtains all relevant documents into a formal case preparation document, and determines the relevant legal or administrative actors required to examine the case for verification and additional input of the case preparation. The clerk sends “write” copies to the defense and prosecution representatives. Each is required to provide input and may raise issues which lead to further up-

date of the document. Several interactions may be required for this, and on each cycle both the defense and prosecution get updated copies. The next set of interactions involves the related courts. For this, a more conservative version of the previous interaction pattern occurs: all the courts are notified about the case through “read” access copies. Each court gets “write” access for its update in a strict order. Once all courts have made their updates, a further iteration may follow. Finally, expert opinion is sought depending on the gathered input from defense, prosecution and court inputs. Initially, a set of expertise sources are identified. Each gets a “read” copy of the case. Opinion is sought as each seeks a “write” request (pull-oriented routing). As opinion is sought, new expertise sources may be raised, updating the routing order. On finalization of updates, the defense, prosecution and courts get copies of the final version of the case preparation document.

Figure 4 illustrates the collaboration between the different parties. This case study two occurrences of the late binding pattern and also includes the Structural changes pattern. The clerk has to select the defense and prosecution representatives as well the experts that are being consulted. Since there might be a special order in which the experts are being asked (e.g. first three telecommunication experts have to comment before two medical doctors are asked for their opinion) Structural changes might apply as well.

This example also shows that the workflow pattern Interleaved Parallel Routing as well as the Multiple Instance patterns are of major importance in the context of Dynamic Routing.

Once again, lightweight process engines could be implemented as extension to the individual participants’ email clients. These local process management systems allow for monitoring and for coordination among the participants.

7. Related work

The basic concepts of Dynamic Routing in the field of computer networking serve as basic inspiration for the application to process-oriented enterprise systems. E.g. Kurose and Ross give a good overview of this area in [6].

In the field of service-oriented architecture work has been done to describe common interaction scenarios (Barros et al. [1]) and to introduce routing mechanisms for message passing between web services. WS-Routing ([7]) introduced an implementation of the Routing Slip pattern described by Hohpe and Woolf in their Enterprise Integration Patterns ([4]). However, WS-Routing was superseded by the new WS-Addressing standard ([2]).

Riss et al. have introduced in [8] a bottom-up process management approach especially suited for knowledge workers. Other authors such as Dustdar ([3]) have investigated the relationship between business process support

and collaborative knowledge management.

In the field of flexible workflows we can find different approaches to handling change at runtime. E.g. Shazia et al. have proposed a framework where the process of change is made part of the workflow process itself ([11]).

Major inspiration also originates from the work by Weber and Wild ([13]) who integrate aspects of workflow management and conversational case-based reasoning.

8. Conclusions

This paper has introduced the concept of Dynamic Routing and has shown how it can be applied to process execution. Architectural considerations were presented and three patterns for assessing modeling languages and systems have been described.

As a first application of Dynamic Routing there is an ongoing implementation effort toward a prototype for supporting task delegation.

As already mentioned this is only the starting point for further research in the area of decentralized and personalized process management. We are going to face major issues with these new enterprise system architectures. E.g. constraints and policies that are defined for an entire organizational unit must not be violated by individual employees. Thus, restrictions on updates etc. must be enforced in the local process management systems. Another big challenge of bringing processes to users is usability. Editing processes and applying them to a particular case or task must be easy and comprehensible. Appropriate notations and user interfaces have to be conceived.

For the underlying infrastructure we need to define a language for describing the process descriptions fulfilling the following requirements:

- It must be *expressive* enough so that it covers the patterns for Dynamic Routing and the most important workflow patterns (control flow [12] / data flow [9] / resource allocation [10]).
- It must be as *simple* as possible and *semantically unambiguous* to facilitate implementing process engines for a wide range of different platforms and applications.

It is still unclear how Dynamic Routing can be integrated with traditional process execution. At the moment both approaches are handled separately from each other. However, we could imagine having fixed parts in the process that are centrally executed as well as flexible parts that are then subject to Dynamic Routing. Process monitoring from both an organization-wide perspective as well as from the perspective of an individual employee are also subject to further investigations.

References

- [1] A. Barros, M. Dumas, and A. ter Hofstede. Service Interaction Patterns. In *Proceedings 3rd International Conference on Business Process Management (BPM 2005)*, pages 302–318, Nancy, France, 2005. Springer Verlag.
- [2] D. Box, E. Christensen, F. Curbera, D. Ferguson, J. Frey, M. Hadley, C. Kaler, D. Langworthy, F. Leymann, B. Lovering, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, E. Sindambiwe, T. Storey, S. Weerawarana, and S. Winkler. Web Services Addressing (WS-Addressing), W3C Member Submission. Technical report, August 2004.
- [3] S. Dustdar. Reconciling knowledge management and workflow management systems: The activity-based knowledge management approach. *Journal of Universal Computer Science*, 11(4):589–604, 2005.
- [4] G. Hohpe and B. Woolf. *Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, October 2003.
- [5] A. Knopf, B. Grone, and P. Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems*. Wiley, May 2006.
- [6] J. F. Kurose and K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [7] H. F. Nielsen and S. Thattle. Web Services Routing Protocol (WS-Routing). Technical report, October 2001.
- [8] U. Riss, A. Rickayzen, H. Maus, and W. van der Aalst. Challenges for business process and task management. *Journal of Universal Knowledge Management*, 0(2):77–100, 2005.
- [9] N. Russell, A. H. ter Hofstede, D. Edmond, and W. M. van der Aalst. Workflow Data Patterns. QUT Technical report FIT-TR-2004-01, Queensland University of Technology, Brisbane, Australia, 2004.
- [10] N. Russell, A. H. ter Hofstede, D. Edmond, and W. M. van der Aalst. Workflow Resource Patterns. BETA Working Paper Series WP 127, Eindhoven University of Technology, Eindhoven, The Netherlands, 2004.
- [11] S. Sadiq, W. Sadiq, and M. Orłowska. Pockets of Flexibility in Workflow Specifications. In *20th International Conference on Conceptual Modelling (ER 2001)*, pages 513–526, Yokohama, Japan, 2001.
- [12] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
- [13] B. Weber and W. Wild. Towards the agile management of business processes. In *Wissensmanagement*, pages 375–382, 2005.