Static Weaving in Aspect Oriented Business Process Management

Authors:

Amin Jalali

This is the authors version of a work that was submitted/accepted for publication in:

Johannesson, Paul and Lee, MongLi and Liddle, StephenW. and Opdahl, AndreasL. and López, ÓscarPastor (Eds.)
Conceptual Modeling
34th International Conference, ER 2015, Stockholm, Sweden, Oct 19-22, 2015. Proceedings
Lecture Notes in Computer Science, Volume 9381, Springer International Publishing, Sweden, pp. 548-557.

The original publication is available at SpringerLink

Static Weaving in Aspect Oriented Business Process Management

Amin Jalali

Stockholm University, Sweden
aj@dsv.su.se

Abstract. Separation of concerns is an important topic in business process modelling that aims to reduce complexity, increase the re-usability and enhance the maintainability of business process models. Some concerns cross over several business processes (known as cross-cutting concerns), and they hinder current modularization techniques to encapsulate them efficiently. Aspect Oriented Business Process Modelling aims to encapsulate these concerns from business process models. Although many researchers proposed different aspect-oriented business process modelling approaches, there is no analysis technique to check these models in terms of soundness. Thus, this paper proposes a formal definitions and semantics for aspect-oriented business process models, and it enables the analysis of these models in terms of soundness at design time through defining a static weaving algorithm. The algorithm is implemented as an artefact that support weaving aspect-oriented business process models. The artefact is used to analyse different scenarios, and the result of analysis reveals the situations that can introduce different problems like deadlock. In addition, an example of such scenario is given that shows how the artefact can detect the problems at design time. Such analysis enables process modellers to discover the problems at design time, so the problems will not be left to be discovered at runtime - which apply a lot of costs to correct them.

Keywords: Business Process Modelling, Aspect Orientation, Weaving

1 Introduction

Separation of concerns has long been used as an effective method by people to deal with a complex phenomenon by reducing the dimensions of the complexity through increasing the level of abstraction. Using these techniques, a system is decomposed into different other modules that specify the functionality of the system in overall. Modularizing a system not only increases the level of abstraction but also supports re-usability of modules, so designing a system can become simpler with the wise choice of modularization techniques. Some concerns like security in information systems are not limited to one module, and they cross over different modules - which are known as cross-cutting concerns. It is challenging to encapsulate these concerns in a module since they are scattered through different modules, and those modules are tangled to these concerns.

Aspect Orientation is a paradigm in information systems that aims to encapsulate the cross-cutting concerns. This paradigm is well researched in the

programming area, where languages like AspectJ are developed to support encapsulation of cross-cutting concerns in software codes [11]. It is also investigated in Requirement Engineering where Aspect Oriented Requirements Engineering (AORE) aims to support encapsulation of cross-cutting concerns [13]. In the Business Process Management (BPM) area, although it has been investigated how these models should be enacted at runtime (dynamic weaving) [9,10], there is still a gap with regard to static weaving (merging the aspect oriented business process models at design time) and its semantics. Thus, it is not possible to investigate if an aspect oriented business process model is sound. This gap applies a lot of cost in enacting such models, because it is not possible to identify problems in these models at design time. Therefore, the problems are left to be discovered at runtime, which can introduce a lot of difficulties to address them correctly.

Therefore, this paper defines the formal definition and semantics for Aspect Oriented Business Process Modelling (AO-BPM) using Petri-nets. The semantics is implemented as an artefact using "Business Process Technologies 4 Java" (jBPT) [12] to weave and analyse the aspect oriented business process models. The implemented artefact is used for analysing different scenarios.

It should be mentioned that we have defined the operational semantics of dynamic weaving for aspect oriented business processes models in [9,10]. These works specify how the Workflow Management Systems should enact these process models at runtime. In contrast, this work focus on weaving AO-BPM at design time. There are also different modeling approaches for AO-BPM which are compared and explained in [8].

The remainder of this paper is organized as follows. Section 2 introduces basic terminologies in aspect oriented business process management. Section 3 gives a brief definition of Petri nets and Workflow nets. Section 4 defines aspect oriented nets. It also defines an algorithm to weave these models to translate them to workflow nets. Section 5 explains the implemented artefact and experience of using it for analysing different scenarios. Finally, Section 6 concludes the paper and introduces future works.

2 Basic Terminologies

This section introduces basic terminologies in AO-BPM through a fictitious scenario. Fig. 1 shows the scenario, where several cross-cutting concerns should be considered by different business processes.

The left side of the figure shows the relation between these concerns and business processes, where concerns like security, logging, auditing and traceability should be considered by "Transfer Money", "Deal for speculation", "Change asset deal", and "Issue a bank draft" processes. A concern should be considered by a process model in this example if it crosses over the process. For example, the "Transfer Money Process" should consider security and logging concerns.

The right side of the figure shows the "Transfer Money Process" which is modelled using standard BPMN. The process starts when the customer fills a form. As a security concern, the customer should sign the transaction if (s)he

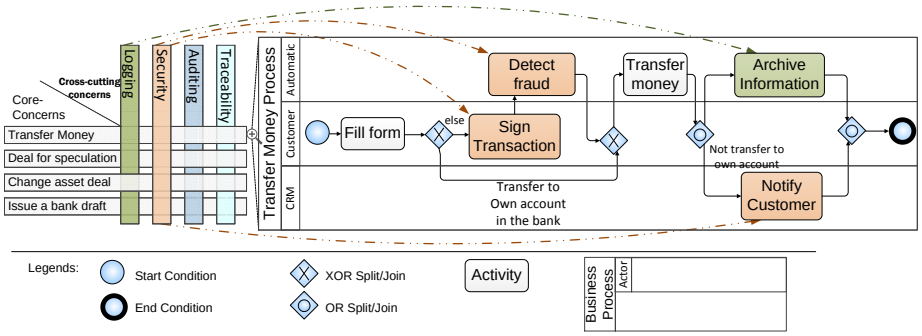

Fig. 1. Cross-cutting in Business Processes

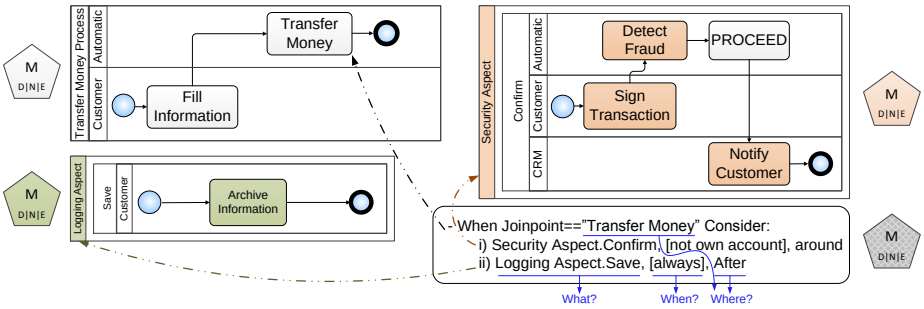

Fig. 2. Separation of cross-cutting concerns

aims to transfer money to someone else account. In such a case, the system investigates potential fraud automatically. Thereafter, the money can be transferred. If the money is transferred to someone else account, the customer should be notified. The information about money transformation should be archived as a logging concern.

AO-BPM suggests encapsulating cross-cutting concerns into separated modules called *advices* (marked by a red and green pentagons in Fig. 2). `Save` and `confirm` are two advices that encapsulate logging and security cross-cutting concerns in our example. The encapsulation of similar cross-cutting concerns are called *aspects*, i.e. `Logging Aspect` and `Security Aspect` in the figure. After separating these concerns from process models, the models only contains the *core-concerns* (marked by a white pentagon in the figure), i.e. `Transfer Money Process`. Each activity in the core concern is called a *join point*, e.g. `Fill Information` activity. To relate core and cross-cutting concerns, AO-BPM introduces a set of rules, called *pointcuts*. Each pointcut relates a join point to a set of advices. The join point that is related to an advice is called an *advised join point*, e.g. `Transfer Money` activity.

Pointcuts contain information about what advice should be considered when and where. For example, for considering the logging cross-cutting concern, the `Save` advice (what) should be considered `always` (when) `After Money Transfer` advised join point (where). Each advice can be considered before, after or around of an advised join point. Around advices has a special activity called *proceed* that defines the place of the advised join point around which the advice should be considered, e.g. `PROCEED` activity in the `Confirm` advice.

3 Preliminaries

This section gives a summary of definitions of Petri nets and workflow nets that are used to define aspect oriented nets later. The definitions are borrowed from [2, 4].

Definition 1 (P/T-nets [2, 4]). A Place/Transition net (P/T-net) is a tuple $N = (P^N, T^N, F^N)$,[1] where:

- P^N is a finite set of *places*;
- T^N is a finite set of *transitions*, such that $(P^N \cap T^N = \emptyset)$; and
- $F^N \subseteq (P^N \times T^N) \cup (T^N \times P^N)$ is a set of *directed arcs*, called the flow relation.

A place is represented by a circle, and a transition is represented by a rectangle. A flow is represented by a directed arrow connecting a place to a transition, or vice versa. The *preset* of an element $x \in P^N \cup T^N$ in the net N is defined as ${}^{N}\bullet x = \{\forall y \in (P^N \cup T^N) | (y, x) \in F^N\}$. The *postset* of an element $x \in P^N \cup T^N$ in the net N is defined as $x^{N}\bullet = \{\forall y \in (P^N \cup T^N) | (x, y) \in F^N\}$. P, T and F represent the universe of all places, transitions and flows, respectively.

The state of a system is determined by distribution of tokens in the places of the net. The distribution of tokens in the places of a net is called the *marking* of the net. The transitions of the net can change the markings of the net. A transition is *enabled* iff there is at least one token in every input places. The result of execution of the transition is a new marking, where a token is removed from every input place of the transition and a token is added to every output place of the transition. The initial state of the system is defined by an initial marking. A marking is called *reachable* iff there exists a sequence of enabled transitions whose firing leads from the initial marking to that marking. The sequence of all executed transition from a marking to another is called the *firing sequence*.

A net is *weakly connected*, or simply connected iff for every two nodes in the net, there are number of flows and nodes that connect one of the node to the other. The net is *strongly connected* iff for every two nodes in the net, there are number of flows and nodes that connect each of them to the other. A marked net is *bounded* iff the set of reachable markings is finite. It is *safe* iff, for any reachable marking, the number of tokens in every place is less or equal than one. A transition in the net is called *dead* iff there is no reachable markings such that the transition be enabled. The net is called *live* iff for every reachable marking, there is also at least another reachable marking from that marking.

Definition 2 (Workflow nets [2, 4]). Let $N = (P^N, T^N, F^N)$ be a P/T-net and $\bar{t}$ an element not in $P^N \cup T^N$. N is a workflow net (WF-net) *iff*: P^N contains an input place i such that ${}^{N}\bullet i = \emptyset$; P^N contains an output place o such that $o^{N}\bullet = \emptyset$; $\bar{N} = (P^N, T^N \cup \{\bar{t}\}, F^N \cup \{(o, \bar{t}), (\bar{t}, i)\}$ is strongly connected.

[1] The superscript N for a set specifies the net that contains the elements of the set.

A workflow net is sound iff these four properties are hole, i.e. i) Safeness: the net with the initial given marking is safe; ii) Proper completion: for any marking that is reachable from the initial marking and has a token in the output place, there is no other token in other places; iii) Option to complete: for any marking that is reachable from the initial marking, there is reachable marking that has a token in the output place (I call this marking as end marking); and iv) Absence of dead tasks: the net contains no dead transitions.

4 Aspect Oriented Nets

I provide formal definitions and semantics for aspect oriented nets in this section. The formal definitions are derived from requirements that are defined in [5,6,10]. As explained earlier, an aspect oriented business process model consists of at least three models for i) core-concern, ii) cross-cutting concerns (advices) and iii) pointcuts. These elements are defined as follows. Then, the static weaving is defined, so these models can be merged to a WF-net. Thus, these models can be analysed through existing analysis techniques for WF-nets.

Definition 3 (Core net). A core net is a workflow net that encapsulates the core functionality of a business process. The set of all Core-nets are denoted by $\mathcal{C}$. Every tasks in a core net is called a join point.

Definition 4 (Advice). An advice $W = (P^W, T^W \cup \mathbb{P}(\{t^W_{proceed}\}), F^W)$, is a workflow net that encapsulates a cross-cutting functionality, where:

- $W \notin \mathcal{C}$. The advice net is not a part of core nets, and
- $t^W_{proceed}$ is a special task in the advice net which is a place-holder for a join point in a Core net. $\mathbb{P}$ denotes the power set, so an advice can have zero or one place-holder.

An advice W has zero or one $t^W_{proceed}$ place-holder. The advice with one place-holder is called an around advice. The set of all around advices are denoted by $\mathcal{A}^{\bullet}_{\ \ \square}$. The advice with zero place-holder is called a free advice. The set of all free advices are denoted by $\mathcal{A}_{\ \ \square}^{\circ}$. The set of all advices are denoted by $\mathcal{A} = \mathcal{A}^{\bullet}_{\ \ \square} \cup \mathcal{A}_{\ \ \square}^{\circ}$, where $\mathcal{A}^{\bullet}_{\ \ \square} \cap \mathcal{A}_{\ \ \square}^{\circ} = \emptyset$.

Definition 5 (Pointcut). A pointcut is a function that relates a join point to an advice. There are three sort of pointcuts, i.e. before, after and around - denoted by $\mathcal{P}_{\circ\square}$, $\mathcal{P}_{\square\circ}$ and $\mathcal{P}^{\bullet}_{\ \ \square}$ respectively. The set of all pointcuts are denoted by $\mathcal{P} = \mathcal{P}_{\circ\square} \cup \mathcal{P}_{\square\circ} \cup \mathcal{P}^{\bullet}_{\ \ \square}$.

- A before pointcut is a function $\mathcal{P}_{\circ\square} : T \rightarrow 2^{\mathcal{A}_{\ \ \square}^{\circ}}$.
- An after pointcut is a function $\mathcal{P}_{\square\circ} : T \rightarrow 2^{\mathcal{A}_{\ \ \square}^{\circ}}$.
- An around pointcut is a function $\mathcal{P}^{\bullet}_{\ \ \square} : T \rightarrow 2^{\mathcal{A}^{\bullet}_{\ \ \square}}$.

This means that before and after advices can relate a set of free advices to a join point, and around advices can relate a set of around advices to a join point. It is also possible that a pointcut relates an empty set of advices to a join point. The join points that are related to at least one advice are called advised join points.

Definition 6 (Aspect Oriented nets). An Aspect Oriented net (AO-net) is a tuple $AO = (C, \mathcal{A}, \mathcal{P})$, where: $C \in \mathcal{C}$ is a core net.

Definition 7 (AO-net Soundness). An AO-net $AO = (C, \mathcal{A}, \mathcal{P})$ is sound iff $W = staticWeaving(AO)$ is sound. Algorithm 1 defines the $staticWeaving$.

Algorithm 1: Algorithm for weaving an AO-net

Algorithm `staticWeaving`$(AO = (C = (T^C, P^C, F^C), \mathcal{A}, \mathcal{P}))$
- $W \leftarrow C$;
- **foreach** *task* t *in* T^W **do**
  - **if** $\mathcal{P}_\bullet(t) \neq \{\}$ **then**
    - $\overset{\square}{T}{}^W \leftarrow T^W \cup \{t^W_{t.split}\} \cup \{t^W_{t.join}\}$;
  - **else**
    - **if** $\mathcal{P}_{\circ\square}(t) \neq \{\}$ **then**
      - $T^W \leftarrow T^W \cup \{t^W_{t.split}\}$;
    - **if** $\mathcal{P}_{\square\circ}(t) \neq \{\}$ **then**
      - $T^W \leftarrow T^W \cup \{t^W_{t.join}\}$;
  - **foreach** *Advice* A' *in* $\mathcal{P}_{\circ\square}(t)$ **do**
    - $addAdvice(W, A', t^W_{t.split}, t)$;
  - **foreach** *Advice* A' *in* $\mathcal{P}_{\square\circ}(t)$ **do**
    - $addAdvice(W, A', t, t^W_{t.join})$;
  - **foreach** *Advice* A' *in* $\mathcal{P}_\bullet(t)$ **do**
    - $addAdvice(W, A', t^{W\,\square}_{t.split}, t^W_{t.join})$;
    - **foreach** *Place* p *in* $\overset{A}{\bullet}t^A_{proceed}$ **do**
      - $F^W \leftarrow F^W \cup \{(t,p)\}/\{(t^A_{proceed}, p)\}$;
    - **foreach** *Place* p *in* $t^A_{proceed}\overset{A}{\bullet}$ **do**
      - $F^W \leftarrow F^W \cup \{(p,t)\}/\{(p, t^A_{proceed})\}$;
    - $T^W \leftarrow T^W / t^A_{proceed}$;
- **return** W;

Function `addAdvice`$(W \in \mathcal{C}, B \in \mathcal{A}, initiator \in T^W, terminator \in T^W)$
- $A \leftarrow B$;
- $T^W \leftarrow T^W \cup T^A$;
- $P^W \leftarrow P^W \cup P^A$;
- $F^W \leftarrow F^W \cup F^A \cup \{(initiator, i^A)\} \cup \{(o^A, terminator)\}$;
- **return** W;

This algorithm gets an AO-net and converts it to a Workflow net, so the nets can be analysed through techniques available for Workflow nets. It weaves related advices to the advised join points in the core net. To discover advised join

points, the algorithm investigates if every task in the core net is an advised join point. If a task t is an advised join point, the algorithm should weave i) "before advices" before the task; ii) "around advices" before and after the task; and iii) "after advices" after the task. Therefore, it adds a helper split and join tasks for weaving advices (line 3-13). Thereafter, it weaves before (line 14-16), after (line 17-19) and around advices (line 20-30) to the net. To weave around advices, the algorithm should also replace the proceed to the advised join point (line 22-28). Finally, the algorithm return the woven net. The function *addAdvice* relates an advice to an advised join point, which is used in weaving before, after and around advices.

5 Experiments

I have implemented a prototype tool of the algorithm to investigate the soundness of different scenarios of aspect oriented business process models. I implemented the static weaving algorithm in Java using the jBPT library [12]. This library makes it easier to extend the implementation to support the weaving of other process model notations like EPC and BPMN. The algorithm gets a core-net and set of advices as Petri nets in the Petri Net Markup Language (PNML) format. It also gets the pointcuts in the XML format which is defined in [10]. The result is a woven net as Petri nets in the PNML format. This format is supported by many Petri nets editors. To analyse the result, I used WoPeD [7]. WoPeD supports analysis of Petri nets using Woflan - an advanced Petri-net-based workflow analyser [3].

The result of analysing different scenarios shows that the woven result is always sound only if i) there is no advised join point in the model for which more than one advice is defined, or ii) no around advice is defined for an advised join point for which more than one advice is defined. Otherwise, the soundness must be checked by the tool.

Fig. 3 shows three nets for a scenario that can have deadlock after weaving. It contains a core process containing three tasks, i.e. A, B and C. A pointcut relates two advices to task B in the main process model. One of the advices is "BEFORE" advice (named before), and the other one is "AROUND" (named around_loop). As it can be seen in the around advice, it is possible to fire the PROCEED by firing loop transition. Fig. 4 shows the result of weaving of the nets for this scenario.

The graph bellow the figure shows the reachability graph, which is generated by WoPeD. The reachability graph shows all possible reachable states in the process model. The analysis based on the reachability graph is a very powerful method because it can be used to prove or disprove all kinds of properties [1].

The reachability graph shows that there is a possible deadlock in this process model. The deadlock can occur if the loop transition is fired. The problem is that the advised join point (task B) requires a token in both input places. However, there will be no token in the output place of the "before advice" net because it is consumed by the advised join point previously. This introduces a deadlock in the woven net.

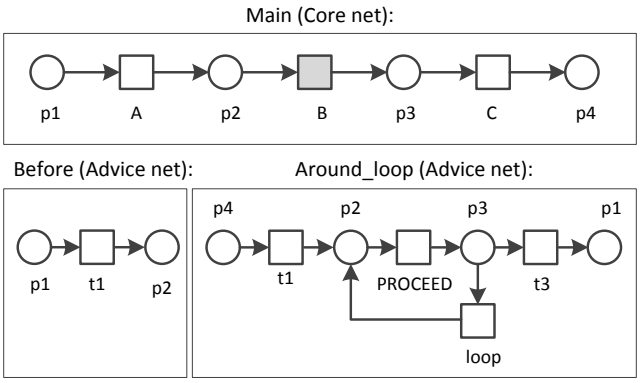

Fig. 3. Aspect Oriented nets defined for our sample scenario

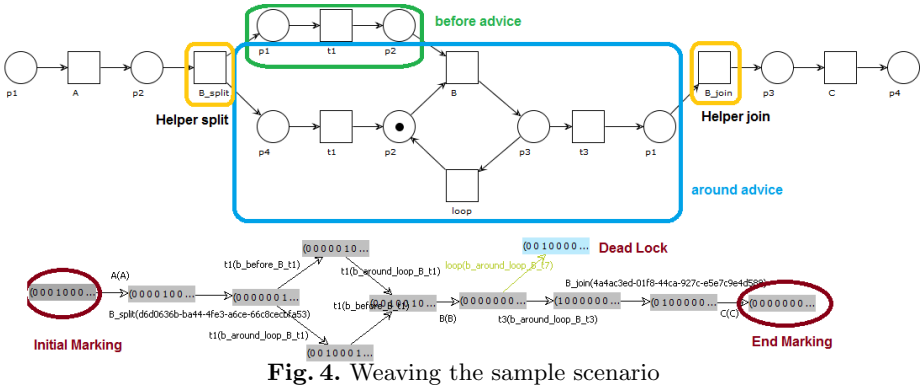

Fig. 4. Weaving the sample scenario

6 Conclusion

This paper defined the formal definition and semantics for aspect oriented business process models and an algorithm to weave these models at design time, known as static weaving. Therefore, it enables analysis of these models in terms of soundness at design time. The algorithm is implemented as an artefact in Java using jBPT library, which has the potential to be extended for other modelling languages like EPC, UML, BPMN, YAWL and etc. The implemented artefact is used to analyse different scenarios. The experience shows that the algorithm can detect different sort of problems that can appear in such models at design time, so they can be prevented to be faced at runtime.

For future works, it would be interesting to i) use the static weaving through real case studies; ii) combine static and dynamic weaving to investigate hybrid weaving; and iii) extend the artefact to support other business process modelling languages.

References

1. W.M.P. van der Aalst. Interval timed coloured petri nets and their analysis. In Marco Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, vol-

ume 691 of *Lecture Notes in Computer Science*, pages 453–472. Springer Berlin Heidelberg, 1993.
2. W.M.P. van der Aalst. Verification of workflow nets. In Pierre Azma and Gianfranco Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer Berlin Heidelberg, 1997.
3. W.M.P. van der Aalst. Woflan: a petri-net-based workflow analyzer. *Systems Analysis Modelling Simulation*, 35(3):345–358, 1999.
4. W.M.P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1128–1142, Sept 2004.
5. C. Cappelli, F.M. Santoro, J.C.S. do Prado Leite, T. Batista, A.L. Medeiros, and C.S.C. Romeiro. Reflections on the modularity of business process models: The case for introducing the aspect-oriented paradigm. *BPM Journal*, 16:662–687, 2010.
6. Anis Charfi, Heiko Mller, and Mira Mezini. Aspect-oriented business process modeling with ao4bpmn. In Thomas Khne, Bran Selic, Marie-Pierre Gervais, and Franois Terrier, editors, *Modelling Foundations and Applications*, volume 6138 of *Lecture Notes in Computer Science*, pages 48–61. Springer Berlin Heidelberg, 2010.
7. A Eckleder and T Freytag. Woped a tool for teaching, analyzing and visualizing workflow nets. *Petri Net Newsletter*, 75:3–8, 2008.
8. Amin Jalali. Assessing aspect oriented approaches in business process management. In Bjrn Johansson, Bo Andersson, and Nicklas Holmberg, editors, *Perspectives in Business Informatics Research*, volume 194 of *Lecture Notes in Business Information Processing*, pages 231–245. Springer International Publishing, 2014.
9. Amin Jalali, Petia Wohed, and Chun Ouyang. Operational semantics of aspects in business process management. In Pilar Herrero, Herv Panetto, Robert Meersman, and Tharam Dillon, editors, *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, volume 7567 of *Lecture Notes in Computer Science*, pages 649–653. Springer Berlin Heidelberg, 2012.
10. Amin Jalali, Petia Wohed, Chun Ouyang, and Paul Johannesson. Dynamic weaving in aspect oriented business process management. In Robert Meersman, Herv Panetto, Tharam Dillon, Johann Eder, Zohra Bellahsene, Norbert Ritter, Pieter De Leenheer, and Deijing Dou, editors, *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, volume 8185 of *Lecture Notes in Computer Science*, pages 2–20. Springer Berlin Heidelberg, 2013.
11. Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and WilliamG. Griswold. An overview of aspectj. In JrgenLindskov Knudsen, editor, *ECOOP 2001 Object-Oriented Programming*, volume 2072 of *Lecture Notes in Computer Science*, pages 327–354. Springer Berlin Heidelberg, 2001.
12. Artem Polyvyanyy and Matthias Weidlich. Towards a compendium of process technologies: The jbpt library for process model analysis. In *Proceedings of the CAiSE'13 Forum at the 25th International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 106–113. Sun SITE Central Europe, 2013.
13. A. Rashid, P. Sawyer, A. Moreira, and J. Araújo. Early aspects: A model for aspect-oriented requirements engineering. In *Requirements Engineering, 2002.*, pages 199–202. IEEE, 2002.