Dynamic Weaving of Aspects for Business Process Management Systems

Amin Jalali¹, Petia Wohed¹, and Chun Ouyang²,³

¹ Department of Computer and Systems Sciences, Stockholm University, Sweden
{aj,petia}@dsv.su.se
² Science and Engineering Faculty, Queensland University of Technology, Australia
c.ouyang@qut.edu.au
³ NICTA, Queensland Research Laboratory, Brisbane, Australia

Abstract. Reducing complexity in Information Systems is a main concern in both research and industry. One strategy for reducing complexity is separation of concerns. This strategy advocates separating various concerns, like security and privacy, from the main concern. It results in less complex, easily maintainable, and more reusable Information Systems. Separation of concerns is addressed through the Aspect Oriented paradigm. This paradigm has been well researched and implemented in programming, where languages such as AspectJ have been developed. However, the research on aspect orientation for Business Process Management is still at its beginning. While some efforts have been made proposing Aspect Oriented Business Process Modelling, it has not yet been investigated how to enact such process models in a Workflow Management System. In this paper, we define a set of requirements that specifies the execution of aspect oriented business process models. We create a Coloured Petri Net specification for the semantics of a so-called Aspect Service that fulfils these requirements. Such a service extends the capability of a Workflow Management System with support for execution of aspect oriented business process models. The design specification of the Aspect Service is also inspected through state space analysis.

Keywords: Business Process Management, Workflow Management Systems, Aspect Oriented, Coloured Petri Nets, Weaving

1 Introduction

Reducing the complexity of models is an important issue in the Business Process Management (BPM) area. Business process models tend to quickly become complex [3], which makes them difficult to communicate, use, maintain and validate [21]. Various approaches have been proposed to reduce the complexity of process models (e.g. [15, 24, 25]). Some of these approaches have been analysed and systemised as a collection of patterns [21]. One of the patterns is called orthogonal modularization, and its purpose is to reduce the complexity of a model by separating different aspects of a process, such as security and privacy. Traditionally, these aspects are defined in a single process model, hence adding to the complexity of the model [26]. In contrast, orthogonal modularization advocates modeling the aspects as separate processes. These processes are related to the
main process, where they represent different pieces of the puzzle. The business process is described as a result of putting together all pieces of the puzzle. The mechanism that puts all aspects and the main process model together is called *weaving*, while the whole technique is called *aspect oriented modularization*.

Aspect oriented modularization so far has been realised as extensions to current business process modelling (e.g. [12, 14, 8]). However, the semantic of such a modularization is not yet defined. Therefore, it is important to develop a generic and rigorous solution that formally specifies the execution semantics of aspect oriented modularization focusing on the enactment of aspect oriented process models in a workflow management system (WfMS).

In this paper, we present a solution to the execution of aspect oriented process models independent of any specific WfMS. The proposed solution is formally specified using Coloured Petri Nets (CPNs) and the correctness of the solution is inspected through state space analysis. The CPN specification can be used as a blueprint for design and realisation of a so-called *Aspect Service* which extends the capability of a WfMS with support for aspect oriented process enactment. We select CPN as it is a widely-used formal technique for system design and verification and its application in the domain of workflow management has been well-established [2].

The remainder of this paper is structured as follows: Section 2 presents a background of aspect oriented business process modelling. Section 3 describes the architecture of the Aspect Service and defines the requirements for the service. Section 4 describes the formalization of the service in CPN, and Section 5 presents the analysis of the solution. Finally, Section 6 discusses related work, and Section 7 concludes the paper and outlines directions for future work.

2 Background

Process models encompass different activities which address different concerns of business processes. *Concerns* are non-functional requirements of a business process which are a matter of interest for stakeholders. Charfi enumerates compliance, auditing, business monitoring, accounting, billing, authorization, privacy and separation of duties as examples of concerns [12]. It is common that some of these concerns are scattered through several business process.

As a real example in Swedish public organizations, it is compulsory to inform citizens if a decision is made on their applications. Accordingly, an *inform* activity is scattered across all business processes that contain a *decide* activity. Moreover, a process may contain several *decide* activities, implying the need for several *inform* activities. If the *inform* activity is changed, or if the policy regarding the informing concern is modified, we have to find and update all business processes containing any *decide* activity. To be conformed to the law, when designing a new business process we should remember to add the *inform* activity after each *decide* activity. Such efforts add costs in designing, updating and monitoring business processes, and increase the risk of inconsistency when updating processes. Also concerns could not be reused since they are implemented separately in each business process. As a result, business processes become more complex, less reusable and more costly to design and maintain.
The Aspect Oriented Paradigm addresses these problems by separating different concerns from the main process. There are various works (e.g. [12, 14, 8]) which provide means for aspect oriented business process modelling. Aspect Oriented Business Process Modeling Notation (AO4BPMN) [12] is one such approach that defines the terminology and suggests a notation for modelling processes according to the aspect oriented principle.

Now let’s consider an example of a business process involving different concerns. Figure 1 describes a simplified version of a Transfer Money Process in the banking domain using BPMN\(^4\). First, a customer fills in information. Next, if she is transferring money to her own account, the transfer is performed; otherwise, she must sign the transaction beforehand. Finally, the transaction is archived. The Sign Transaction activity is part of the security aspect, and the Archive Information is part of the logging aspect. These aspects describe different concerns related to the Transfer activity.

Figure 2 depicts the AO4BPMN model of the Transfer Money Process. The concerns are removed from the main process and modelled separately through aspects. Hence, the main process contains only the Fill Information and Transfer activities. An aspect is the realisation of a concern through one or a number of processes. Each aspect is modelled in a separate model, i.e. the Logging Aspect and the Security Aspect. Each aspect model is annotated

\(^4\) Note that for simplicity we omit pools/lanes in this process model.
with an Aspect label. An aspect may have one or several advices, each annotated with an Advice label. An advice captures a specific part of a concern that should be considered if a condition is fulfilled. This condition is called Pointcut. The pointcut shows when and where the advice should be integrated with the process model. The possible points of integrations are called Join Points. In AO4BPMN, these are activities. For instance, Fill Information and Transfer activities are both examples of join points. A join point can be related to an aspect by defining a pointcut. In such case, a join point is called advised join point. In the example, the advice in the Logging Aspect is related to a pointcut, named Archive. This pointcut is related to the Transfer activity through an annotation, which shows that the Logging Aspect is relevant to the Transfer activity. The Transfer activity is an advised join point.

In AO4BPMN, the PROCEED activity acts as a “placeholder” indicating when the advised join point activity should be performed. This activity is optional [12] to be included in an advice. We call an advice which contains the PROCEED activity an explicit advice and an advice which does not have any PROCEED activity implicit advice. In Figure 2 the Archive Information will be completed after the Transfer activity. There is also another pointcut which is related to the Transfer activity. It shows that the Transfer activity should be signed when a customer wants to transfer money to an account owned by someone else. This concern is modelled through the Security Aspect. This aspect contains Sign Transaction activity, which is performed before the Transfer activity. It should be noted that it is also possible to intersperse a PROCEED activity with the activities of an advice. This implies that the advice will be considered around the advised join point.

AO4BPMN modelling increases reusability because an aspect can be related to different activities and even different processes. It also facilitates the maintenance of processes, because if a concern changes, the change would only be reflected in one place. Finally, it reduces the complexity of a process model as it reduces the number of activities inside the process.

3 Overview of the Solution

Existing aspect oriented modularisation approaches such as AO4BPMN only support process modeling, whereas the enactment of aspect oriented business processes is still an open issue. In this section, we propose a solution in form of a so-called Aspect Service, which complements current WfMSs with support of aspect oriented modularization.

We define this service as a sub-service of the Worklet Service [5]. The Worklet Service is defined to support flexibility for business processes [4]. It is utilized because: (i) it provides a foundation for extending the behaviour of business process execution in a WfMS; (ii) it is based on Service Oriented Architecture (SOA), hence applicable for any WfMS; (iii) it is open-source and currently proven as an implementation for YAWL; and (iv) it is formalized through CPN which enables reuse of relevant sub-nets [7].

For explaining the Aspect Service, we use an abstract example (see Figure 3). The example contains a main process with four aspects, which are defined for one
of its activities, activity B. The enactment of the business processes is managed through a WfMS. It results in five different process instances. The Aspect Service takes care of the weaving of the aspects to the main process. This means that the advices are executed in parallel, and a synchronisation towards the main process is made at the Proceed placeholder as well as at the end of the execution of the advices. In this particular example, the execution sequence of the activities will be A, followed by D, G and H in parallel, then B, followed by E and F in parallel, and finally C. This is written as regular expression $A(D||G||H)B(E||F)C$ (see also the Woven Result in Figure 8).

Figure 4 shows the architecture of a typical WfMS where the Aspect Service is a part of the Worklet Service [7]. The Worklet Service is designed to support flexibility in business process management. It has certain built-in functions, such as those for suspension of an work item, which is useful for Aspect Service. The Worklet Specs repository is used to store the advices. These are defined in a process editor. The Rules repository is used to store pointcut definitions. The communication between the Worklet service and the Workflow engine happens through a number of message exchanges. The messages that we utilize are shown in Figure 6.

The Aspect Service gets enabled upon receiving a constraint. In [6], four types of constraints are defined, i.e., CasePreConstraint, ItemPreConstraint, ItemPostConstraint and CasePostConstraint. ItemPreConstraint and ItemPostConstraint represent the beginning and ending of a work item while CasePreConstraint and CasePostConstraint represent the beginning and ending of a case. Each time a work item gets enabled, a constraint of the type ItemPreConstraint is raised by the WfMS. The Aspect Service receives this constraint and performs two checks: a check on whether the workitem has a pointcut associated to it and if so, if the pointcut is met (recall that a pointcut is a condition). If the workitem is not related to a pointcut, or if a pointcut condition is not met, the execution of the workitem is proceed as usual. Otherwise, the Aspect Service starts the weaving of the corresponding aspect(s). We call this the Initiating step. This step
represents the initiating requirements of the service. I.e. the service shall check if: a received workitem constraint is related to a pointcut; if a received workitem constraint is related to a Proceed placeholder; if a received case constraint is related to the end of an advice.

In fact, the same checks are also performed when an ItemPostConstraint is raised (which occurs at a work item completion). This is a way to indicate the beginning and the end of an advised join point and to distinguish it from normal workitems.

The weaving orchestrates the enactment of the main business process module and its related aspects. Both the main process and related aspects are enacted in the WfMS as separate business processes. Therefore, the weaving is performed by the Aspect Service through receiving and sending sets of messages from/to the WfMS. This orchestration is performed in four steps representing four requirements. These steps are visualized in Figure 5 (left hand side). Each step contains several message exchanges between the Aspect Service and the Workflow engine. The messages and the steps they are used in, are illustrated in Figure 6. The steps are detailed below.

1 **Launching**: When the Aspect Service is activated, it sends a message to the WfMS to suspend the main process. When the suspension is confirmed

---

Fig. 4. Aspect Service (as a part of the Worklet Service) in a typical WfMS Environment

Fig. 5. Weaving steps
by the WfMS, the Aspect Service sends messages to the WfMS for launching the relevant advices. Hence, the launching requirement is defined as the Aspect Service shall support the WfMS to suspend the execution of an advised join point (R1.1) and launch its related aspects (R1.2).

2 Pausing: When an advice reaches the Proceed placeholder, the Aspect Service sends a message to the WfMS to suspend the advice. However, advices that do not have a Proceed placeholder will reach to their End. Once the relevant advices have been suspended or ended, the Aspect Service orders un-suspension of the advised join point. Hence, the pausing requirement is defined as the Aspect Service shall support the WfMS to suspend a launched advice upon reaching its Proceed placeholder (R2.1) and un-suspend an advised join point once all its advices have been suspended or ended (R2.2).

3 Resuming: After the advised join point has been completed, the WfMS raises an ItemPostconstraint. The Aspect Service sends messages for suspension of the advised join point and un-suspension of the corresponding advices. Then, it sends messages to force complete the Proceed placeholders, so the advices can be continued. Hence, the resuming requirement is defined as the Aspect Service shall support the WfMS to suspend the execution of an advised join point (R3.1) and un-suspend the execution of its paused aspects (R3.2).

4 Finalizing: When all advices are ended (i.e. their CasePostconstraints have been raised), the Aspect Service sends message to the WfMS to unsuspend the advised join point. Hence, the weaving is completed, and the control of the main process is handed back to the WfMS. Hence, the finalizing requirement is defined as the Aspect Service shall support the WfMS to
un-suspend the advised join point when its related aspects are finished (R4).

During all this steps, the business data is synchronised between the main process and its aspects. In case several advices operate on the same data simultaneously (see for example activities D, G, and H in Figure 5) the last workitem to complete will overwrite the data stored by the workitems completed earlier. In the next section, we describe the CPN model for the Aspect Service.

4 Formal Semantics

The formalisation of the Aspect Service is specified using hierarchical Coloured Petri Nets. The solution is a three-level model. The top-level module captures the behaviour of the Initiation of the service (see Figure 7). The second level captures the weaving behaviour (see Figure 8). This model contains four modules capturing the requirements related to weaving steps 1 to 4 that are described in the previous section. It also contains a module for communicating with the WfMS and performing actions for data persistence, which is needed for the weaving of the aspects to the main process. These five modules constitute the third level of the CPN Model.

The model defines 57 colour sets and 33 functions. Full details of the model with definition of the colour sets, variables and functions can be downloaded from [1]. We re-used some of the colour sets, variables and functions from the Worklet Service CPN model [7]. In the following subsections, we explain selected parts of the Aspect Service CPN model.

4.1 Level 1: the AspectService module

As depicted in Figure 6, the interaction of the Aspect Service and a WfMS is realized through passing a number of messages. These messages are named constraints and commands. Constraints are the messages raised by the WfMS, and Commands are the messages invoked by the Aspect Service. All these messages should be supported by the WfMS. We used YAWL [13] as an example system and the name of the messages as defined in YAWL. However, it should be noted that although the names may differ from one system to another, the messages are generic. Hence, the solution is general and can be adapted to any WfMS. The constraint messages are WorkitemConstraint and CaseConstraint. The raising of one of these messages is signified in the CPN model in Figure 7 as a token arriving in the workitemConstraint or caseConstraint places correspondingly. We have highlighted this by shading these places in the model. In other words, these places are the starting points of the net.

The first level of the net addresses the initiating requirement by processing the received constraints. The constraints are related to workitems or cases. Workitem constraints represent the enabling and ending of a work item. A work item is an instance of a task. Each task can be associated with a pointcut, except the Proceed placeholders. Therefore, these work items are checked if they meet the conditions of their pointcuts. The net performs this check using the
matchPointcut transition. If the pointcut is met, the bold part of the net is executed. The result of this execution is a token in AdvisedJP or AspectInfo places. If the constraint is an ItemPreConstraint, a token is produced in AspectInfo place; otherwise, a token is produced in AdvisedJP place. Finally, if the token in the workItemConstraint place represents a constraint for a Proceed place-holder, it will be consumed by the isProceedCmd transition. This transition produces a token in Proceed place as a result.

In order to fulfil the initiating requirement, the case constraints should also be addressed. Therefore, the endAdvice transition checks the tokens in the caseConstraint place. This transition produces a token in completeAdvice place if the case constraint is related to an advice.

In conclusion, this net controls how the weaveAspect net is enabled. The weaveAspect net is responsible for addressing the weaving requirements and is described in the next subsection.

### 4.2 Level 2: the weaveAspect and selectPointcut modules

There are two sub-nets at the second level, i.e. the selectPointcut and the weaveAspect modules. The selectPointcut module is taken from the Worklet Service [7] and used to check whether a pointcut condition is met.

The weaveAspect sub-net (see Figure 8) contains five modules. These are: Launching, Pausing, Resuming and Finalizing modules which handle the four weaving steps, and Core module which is mainly responsible for persisting the data along these four steps. Furthermore, the Core module is also responsible for sending messages to the WfMS. All these five sub-nets have a common place, ICore, which is both input and output place for these sub-nets.

\[5\] In [7] this net is called ‘CPN: Evaluating the Rule Tree’.
Fig. 8. CPN: weaveAspect

The tokens in this place symbolize different messages passed through the nets. The color set of the ICore place is COREMSG. COREMSG is the product of a command name and a parameter which is a list. Each of the nets produces and consumes specific sets of tokens with different commands. The ICore place is the foundation for the communication between the nets. A brief description of one of these nets is given below.

4.3 Level 3: the weaving steps modules

The Launching sub-net, depicted in Figure 9, is designed to address the launching requirements. One consideration when developing it was the capability of launching several advices. The launchAspect transition produces four tokens in the ICore place. One of these tokens, with suspendWorkItem command, is used by the Core module to send a message to the WfMS to suspend the advised join point (fulfilling R1.1). One of the tokens, with initAdvice command, is consumed by the Core module to persist the number of advices related to the advised join point. The other two tokens, both containing getAdviceNumber command, are used to retrieve the number of advices for the advised join point. The launchAspect transition also produces a token in the Aspect place. The getnextAdvice transition extracts all advices of the pointcut, and produces individual tokens for them in advice place. After all advices are extracted, the advised join point is suspended, and the number of advices related to the advised join point are retrieved, the enableLauncher transition will produce some tokens in AdvisedJP place. The number of tokens is equal to the number of advices, so initAdvice transition fires once per advice.

As mentioned earlier, each advice is enacted as a separate business process. The Aspect Service does not know the ids of the advices before launching but needs to keep track of them once they become known. To distinguish the id for each advice, the Aspect Service must launch them individually. This is captured by the bold parts in the net where place ExclFlag, containing most one token at the time, enforces a sequential launching.
Basically, the launching is realised in three steps. First the input parameters are retrieved, second an advice is launched, and third the relation between the advised join point and the launched advice is stored. The first step is realized through the initAdvice transition. This transition produces three tokens. The first token, in place ICore, is used by the Core sub-net to send a message to the WfMS to retrieve the parameters of the advice. The second token, in place joinPointID, keeps the advised join point id. This information is later used to persist the relation with the advice. The third token, in AdviceInfo place, keeps the advice and the advised join point data. This token is consumed, in the second step, by the launchAdvice transition to launch the advice (fulfilling requirement R1.2). This transition is enabled when the data regarding the input parameters of the advice is retrieved, which is captured through a token with rspGetInputParams command in the ICore place. The third step is realized through the setAdviceState transition. This transition consumes the token in the joinPointID place. It also consumes a token from the ICore, which contains the ackLaunchCase command and the id of the launched advice. As a result, this transition produces a token in the ICore place with setAdviceInfo command.

The Pausing sub-net, depicted in Figure 10, is designed to address the pausing requirements. The net is enabled when the execution of a process in the WfMS reaches a Proceed placeholder. In this case, the WfMS raises a constraint, and the top-level module produces a token in the Proceed place, as described Section 4.1. The Pausing net should do three steps: i) pausing the advice (R2.1); ii) updating the advised join point data using the advice data; and iii) saving the
The relation between advice and Proceed placeholder, which is needed for resuming it later.

The first and third steps are started when the prepare transition produces tokens in the ICore place with a suspendWorkItem and saveJoinpointID commands correspondingly. However, the second step cannot be done until the advised join point id is known. Therefore, the prepare transition produces a token in the ICore place with getAdviceInfoByCase command to retrieve this information. When this is done, the updateData transition consumes the token containing the result and starts the update by producing a token in the ICore place with updateWorkItemData command.

![CPN: Pausing](image)

Fig. 10. CPN: Pausing

When all advices related to the advised join point are handled, and all data are synchronised, the service un-suspends the advised join point. Therefore, the setAdviceNum transition consumes all tokens which represent the advices of an advised join point and produces a token in the adviceNum place which contains the number of advices. The setAdviceInfo transition reduces the number of advices when the advice is suspended and the advised join point data is synchronised with the advice data. When the number of advices for an advised join point in the adviceNum place is zero, the proceedAdvisedJoinPoint transition gets enabled. This transition un-suspends the advised join point (R2.2). It also produces two tokens in the ICore place containing resumeFlag and getAdviceNumber commands. These tokens are used for resuming the advices.

The Resuming, Finalizing and Core subnets fulfil the rest of requirements, but due to space limitation these are not described here.

5 Analysis

The CPN model presented in the previous section allows us to verify the design of the Aspect Service using state space analysis. The Aspect Service is designed
to deal with any number of advices. The verification is complicated due to the numerous aspects/advices and their possible combinations that may be triggered for an activity. It is not realistic to go through each of the possible situations. Instead, we divide all the possible situations into four groups, and define four abstract types to characterise them. These are:

- **Type 1**: no advices are triggered for an activity;
- **Type 2**: at least one implicit advice and no explicit advices are triggered for an activity;
- **Type 3**: at least one explicit advice and no implicit advices are triggered for an activity;
- **Type 4**: at least one implicit advice and at least one explicit advice are triggered for an activity.

Note that as to Type 3 and Type 4 above, we do not distinguish between the variations ‘before’, ‘after’, and ‘around’ that apply to an explicit advice. The reason is that the sequence of message exchanges between the Aspect Service and a WFMS is the same independently of where the Proceed placeholder (determining the ‘before’, ‘after’, ‘around’ behaviour) is positioned in an advice.

We assume that all advices are independent of each other, i.e. there is neither interference nor data dependency between any two advices. Hence, it is valid to scope the verification within a set of four scenarios which are simple but representative examples of the above four types of situations. These are:

- **Scenario A**: no advice is triggered for an activity (i.e. an advised join point);
- **Scenario B**: one implicit but no explicit advice is triggered for an activity;
- **Scenario C**: one explicit but no implicit advice is triggered for an activity;
- **Scenario D**: one implicit and one explicit advices are triggered for an activity.

Figure 11 shows graphical descriptions of these four scenarios and the necessary steps performed for aspect weaving.
Table 1. The state space statistics and the number of deadlocks and cycles of the four scenarios in Figure 10

<table>
<thead>
<tr>
<th>Scenario</th>
<th>State space statistics</th>
<th>SCC graph statistics</th>
<th>Deadlocks</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>No. of nodes: 11</td>
<td>No. of nodes: 11</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>No. of arcs: 10</td>
<td>No. of arcs: 10</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>No. of dead markings: 1</td>
<td></td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>B</td>
<td>No. of nodes: 78</td>
<td>No. of nodes: 78</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>No. of arcs: 118</td>
<td>No. of arcs: 118</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>No. of dead markings: 1</td>
<td></td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>C</td>
<td>No. of nodes: 270</td>
<td>No. of nodes: 270</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>No. of arcs: 550</td>
<td>No. of arcs: 550</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>No. of dead markings: 1</td>
<td></td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>D</td>
<td>No. of nodes: 1863</td>
<td>No. of nodes: 1863</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>No. of arcs: 5993</td>
<td>No. of arcs: 5993</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>No. of dead markings: 1</td>
<td></td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

In the Aspect Service CPN model, each scenario\(^6\) is captured via configuration of the initial marking of the model which is specified by the marking of place \textit{workitemConstraint} and the marking of place \textit{caseConstraint}. As a result, four state spaces are generated corresponding to each of these four scenarios. Table 1 lists the state space statistics of each scenario and the number of deadlocks and cycles derived from the analysis of each scenario. The result indicates that all the scenarios are free of deadlocks. Also, for each scenario its Strongly Connected Component (SCC) graph has the same number of nodes and arcs as the corresponding state space. This indicates that there is no cycle in the CPN model, which is expected since the weaving process does not contain any cycles.

Next, we investigated if the CPN model for each scenario fulfils the specified requirements (see Section 3) by model checking the corresponding state space. For example, one verification result shows that the advised join point is always suspended regardless of which execution path is taken thus demonstrating the fulfilment of Requirement R1.1.

Finally, we also examine two more scenarios, one of Type 2 by considering two implicit advices and one of Type 3 by considering two explicit advices, for an extra check of the synchronisation between main process and the associated advices. The analysis results indicate that this synchronisation works as expected. Hence, we argue that increasing the number of advices would not influence its behaviour and are confident that the design of the Aspect Service is sound in dealing with any number of advices.

6 Related Work

To support the Aspect Oriented paradigm two components are needed: decomposition for capturing separation of concerns, and integration i.e. the weaving of aspects with processes.

In the process modelling area, there are some attempts for process decomposition, e.g. [12, 20, 19, 23, 10]. Despite these numerous attempts for the process

\(^6\) The scenarios can be downloaded from [1]
modelling area, we could not find any work which shows how *weaving should be performed when executing process models*. Therefore, in the work presented here, we elaborate on the *weaving* for business process management. The work was inspired by the work on weaving in Programming e.g. [9, 18, 17, 16].

Finally, it should be mentioned that there is one implementation of weaving for service orchestration, namely AO4BPEL [11]. AO4BPEL is an extension to the Business Process Execution Language (BPEL) to support aspect orientation. This extension is, language specific and does not show how the weaving should be performed at a general level. Furthermore, it does not specify the requirements which are necessary to implement weaving in WfMSs. Finally, BPEL neither supports graphical representation of the business processes, nor the involvement of human resources in them, both of them are fundamental for the BPM area. Thus, these limitations are inherited to the AO4BPEL.

### 7 Conclusions and Future Work

In this paper, we presented a generic solution to address how the weaving of aspects to business processes can be done. The solution is designed in form of a service, namely Aspect Service, which extends a WfMS to support enactment of aspect oriented business process models. We provided a formalisation of the Aspect Service using CPNs and verified the soundness of the design of this service using state space analysis.

The solution is currently limited to weaving advices in which the Proceed placeholder is only enabled once. This means that the Proceed placeholder cannot be included in loops and that in case several Proceed placeholders are defined within the same advice, care must be taken that only one of them can be enabled during the execution of the advice (e.g. as a result of an XOR split). The impact of these limitations, i.e. how frequent such scenarios occur in real life, needs to be studied further.

Other directions for future work include: (i) an implementation of the Aspect service in a state-of-the-art WfMS such as YAWL [13]; (ii) a comparison of Aspect Orientation in the programming and BPM areas. Such comparison would fortify Aspect Oriented BPM, as the Aspect Orientation is more mature in the programming area; (iii) applications of Aspect Oriented business process management in different areas, e.g., health care, bank and finance, to study benefits of the work in a real life setting; (iv) definition of a pointcut language which captures other business process perspectives such as the resource perspective; and (v) investigation on how the resource patterns [22], e.g., separation of duties and retain familiar, should be captured in orthogonal modularization.

### Acknowledgement

We would like to thank Prof. Paul Johannesson for his valuable comments on earlier versions of the paper.
References


