Aspect Oriented Business Process Modelling with Precedence

Amin Jalali[1], Petia Wohed[1], and Chun Ouyang[2,3]

[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. Complexity is a major concern which is aimed to be overcome by people through modelling. One way of reducing complexity is separation of concerns, e.g. separation of business process from applications. One sort of concerns are cross-cutting concerns i.e. concerns which are scattered and tangled through one or several models. In business process management, examples of such concerns are security and privacy policies. To deal with these cross-cutting concerns, the aspect orientated approach was introduced in the software development area and recently also in the business process management area. The work presented in this paper elaborates on aspect oriented process modelling. It extends earlier work by defining a mechanism for capturing multiple concerns and specifying a precedence order according to which they should be handled in a process. A formal syntax of the notation is presented precisely capturing the extended concepts and mechanisms. Finally, the relevance of the approach is demonstrated through a case study.

Keywords: Business Process Modelling, BPMN, Aspect Oriented, Separation of concerns

1 Introduction

The interest to business process management has increased considerably during the last decade. BPMN is one of the most widely spread notation of business process modelling. Business processes are associated with a set of requirements some of which also reflect different concerns. Examples of concerns are security and logging. Concerns are typically *cross-cutting*, i.e. they are relevant for several business processes. For example, Figure 1 shows four typical concerns from the banking domain that spans across four processes. In addition, concerns can also be reflected in several places in one same process, i.e. they are *scattered* through a process.

Traditionally, as can be seen from Figure 1, the concerns are modelled as an integral part of the processes. This often leads to complex, inflexible and less reusable solutions. The complexity is increased as the number of tasks in a process grows to cover both business logic and cross-cutting concerns. The solution is not flexible as changes in a concern have to be reflected in multiple places.

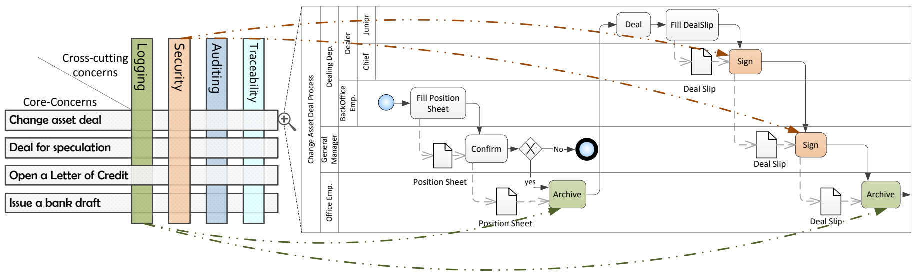

Fig. 1. Example of concerns in the context of a business process

Reusability is not supported due to the lack of placeholders for the concerns that can be refereed to when relevant.

To address these issues the aspect oriented principle has been proposed. In essence, this is a *separation of concerns*, advocating the separation of cross-cutting concerns from the core business process logic (which for short will be called core concerns). Within the programming paradigm, this principle is realised in Aspect Oriented Programming (AOP) (see for instance AspecJ [2]). In the business process management paradigm the aspect oriented principle has been introduced only recently. Charfi et al. [9] elaborate on how the separation of concerns can be handled in business process modelling by extending the Business Process Modelling Notation (BPMN) with notions for aspect oriented process modelling. They also extend BPEL [8] with features for aspect oriented Web service composition. Another existing effort is seen as the work by Cappelli et al. on proposing a different notation for aspect oriented business process modelling [1].

However, when applying the existing approaches, we recognised that not all the concerns could be separated from a business process model, due to the fact that none of these approaches can capture multiple concerns with sequential order of execution. In this paper we take the advances from [9, 8] and extend the approach presented in [9]. The contributions are three-fold. Firstly, we define a requirement which is necessary for capturing multiple concerns in a process with specific orders. We called it precedence requirement and extend the findings in [9] to fulfil this requirement. Secondly, we provide a rigorous formalisation of our approach to precisely capture the extended concepts and mechanism. Finally, we study and examine the relevance of the extended aspect oriented modelling mechanism using a case study.

The remainder of the paper is organized as follows. In Section 2 we present a conceptualisation of aspect oriented business process modelling. This includes a set of requirements for designing aspect oriented process modelling paradigm, the concepts for aspect orientation in business process modelling, and a formalisation of the correspondingly extended mechanism. Section 3 demonstrates the approach through a case study. Section 4 discusses the limitations of the current findings. Section 5 presents an overview of the related work in the area. Finally, Section 6 concludes the paper and presents directions for future work.

2 Approach

In order to provide support for aspect oriented business process modelling, some terminology need to be introduced. This terminology is influenced by the terminology in Aspect Oriented Programming. To exemplify it, we use the Business Process Modeling Notation (BPMN) [18]. We choose BPMN because: (i) it is a well known and widely spread out modelling notation and (ii) because our work initially targeted to extend the work by Charfi et al. [9] where BPMN was extended for the purposes of aspect oriented business process modelling. However, it should be noted that the conceptualization proposed here is general and could be adapted to extend other modelling notations such as UML Activity Diagrams, EPC, YAWL, etc. We start the presentation with a discussion of some basic requirements.

2.1 Requirements

When developing support for aspect oriented business process management there are some important requirements that need to be considered. These are compiled in [19] for the software engineering domain, but they are general and therefore applicable for the business process management domain as well. We summarize them in the list below and discuss their application in the business process management domain:

R1 It should be possible to identify and encapsulate concerns *simultaneously*. The concerns are *equal*, i.e. there is not a dominant concern that obstructs the extraction of other concerns. This means that *a notation supporting aspect oriented business process modelling shall allow for the presentation of multiple concerns relevant for a process.* In addition we identify the need for associating several concerns to one activity. This means that the notation *should be able to express the precedence order between multiple concerns* associated to an activity. I.e., it should be possible to specify both parallel and sequential order of execution of the concerns.

R2 It should be possible to identify and add concerns *incrementally* at any time during the development lifecycle. For business process management this means that *the addition of new concerns at a later stage of the development should be easy and without the need of invasive re-modelling.*

R3 Developers should not be required to know details of concerns that do not affect their particular activities. I.e. concerns should be "encapsulated" and business process analysts should be able to deal with one complexity at the time. In other words, *it should be possible to profile analysts*, i.e. business analysts of the core processes, business analysts of the security policies or archiving routines and etc.

R4 It must be possible to represent and manage *overlapping* and *interacting* concerns. For example, the Logging concern for a business entity may contain security elements. Therefore, in business process management *it should be possible to identify, model, execute and maintain processes which contain overlapping concerns.*

R5 “any separation of concerns mechanism must also include powerful integration mechanisms” [19]. In business process management context, it means that it is important to develop services (or software modules) that extend the behaviour of present workflow management systems in such a way that they can interpret and enact models that are produced with the aspect oriented principle.

While requirements R1-R4 are applicable in the design of an aspect oriented modelling notation, R5 is clearly related to the design of the underlying software. Hence for this paper, the first four requirements are of interest. In line with R5 we designed a service called the Aspect Service using Coloured Petri Nets (CPN) and present it in [13].

2.2 Concepts

We describe the concepts of aspect orientation with a fictitious `Transfer Money` process of a bank (see Figure 2). The process starts with a customer submitting a request of transferring money, i.e. `Fill form` activity. If the transfer is directed to an account owned by the customer, it is executed directly (i.e. activity `Transfer money`), if not the customer is asked to sign the transfer request (activity `Sign Transaction`), and then an automated `Detect fraud` activity is executed. After the money has been transferred, the transaction is archived (`Archive information` activity). If the transfer is made to an account with a different owner, the customer is also notified (`Notify Customer` activity), which is done before the archiving.

Looking closer into the `Transfer Money` process we can identify two concerns namely Security and Logging. The activities related to these concerns are coloured in two different ways to distinguish them from the core process. Figure 3 shows the same process modelled according to the aspect oriented principle. This implies that the Logging and the Security concerns are extracted from the core process and modelled as individual processes. We adopt the terminology introduced in aspect oriented programming and call the representation of concerns for *Aspects*. Although not shown in the example, an aspect can contain more than one process. These processes are called *Advices*. An advice contains a *PROCEED* activity, while a core process contains *Join Points*. The joint points show the possible places in a process where an aspect can be related to a process.

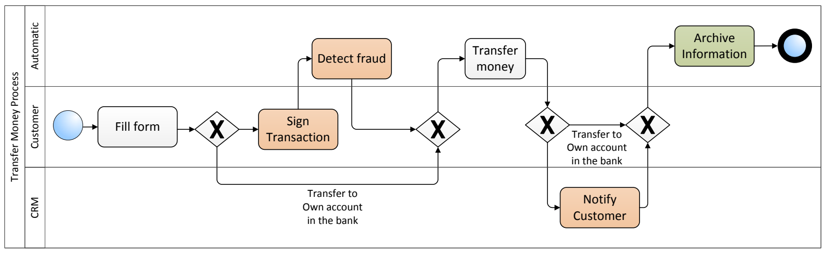

Fig. 2. Transfer Money Process - modelled in traditional way

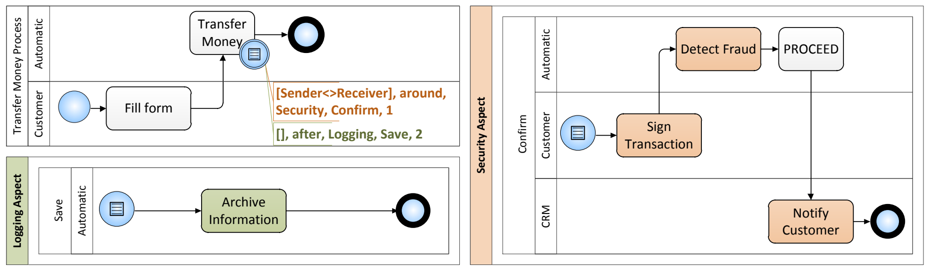

Fig. 3. Transfer Money Process - modelled according to the aspect oriented principle

For BPMN, these are all activities. When an activity is related to an advice, it is called *advised join point* activity. The `Transfer Money` activity in Figure 3 is an example of this. We propose the use of a *conditional event* on the border of an activity for indicating an advised join point.

In addition, each conditional event is annotated. The annotation shows the relevant advice for the joint point and the condition which needs to be fulfilled in order to trigger this advice. These conditions are called *pointcuts*. In the `Transfer Money` process, the `Confirm` advice is triggered only if a transfer request specifies a different account owner. Furthermore, the annotation shows when the advice process shall be executed in relation to the advised join point activity. The alternatives *before*, *after*, and *around* are possible. When the around alternative is specified, the corresponding advice also need to contain a *PROCEED* activity. The *PROCEED* activity is a placeholder which shows where in an advice process the corresponding advised join point activity should be executed. An advice process can have zero or one *PROCEED* activity: when zero the advice process is called *implicit*, otherwise *explicit*.

Encapsulation is achieved by modelling each concern as an advice process. Advices are grouped into Aspects based on their focus, e.g. Security and Logging. To support the execution of a process like the one in Figure 3, the functionality of a WfMS needs to be extended so that execution sequence specified in Figure 2 can be derived. The “integration” of an aspect to a process is called *weaving*. We also developed a service, called the Aspect Service, which specifies the semantics of the weaving [13].

It should be noted that the model in Figure 3 exposes the need for associating multiple advices to an activity. In the example, the advices shall be executed in a sequence, which means that it is important to be able to specify the execution order for them. This is done as part of the annotation. We call this order for the *precedence order*. The precedence requirement is recognized from the programming area, but have not been considered in previous work related to business process modelling. We recognised the relevance of this requirement through a case study.

2.3 Formalisation

We present a formalization of the syntax of BPMN covering the set of core elements depicted in Fig. 4. Based on that we then define the syntax of BPMN extended with the *Aspects* considerations and refer to it as Aspect-Oriented Business Process Modeling Notation (AOBPMN).

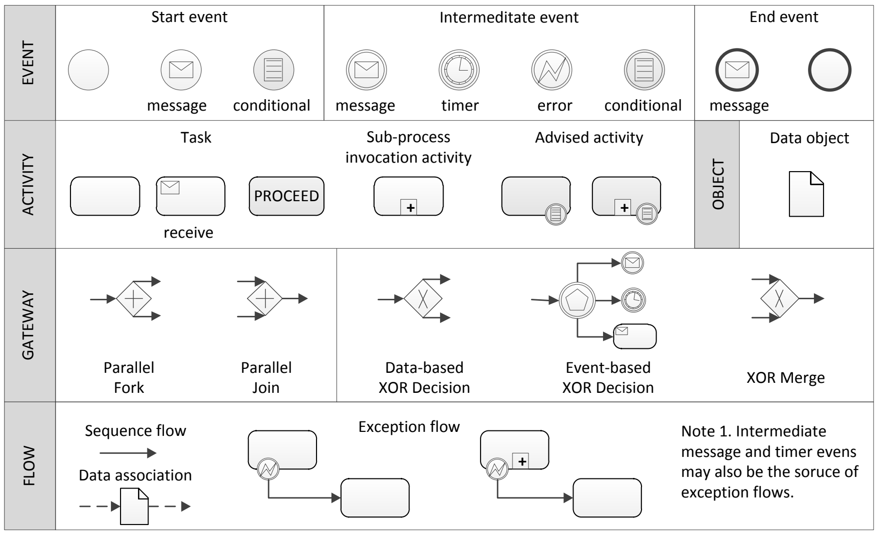

Fig. 4. A core subset of BPMN elements

The formalisation of the syntax of BPMN builds on the previous syntax definition (based on BPMN1.0) in [10] and extends it with the data and resource information and elaboration on events and exception constructs according to BPMN2.0 [18].

Definition 1 (Standalone BPMN Process). *A standalone BPMN process is a tuple* $\mathcal{M} = (\mathcal{O}, \mathcal{A}, \mathcal{A}^T, \mathcal{A}^S, \mathcal{E}, \mathcal{E}^S, \mathcal{E}^I, \mathcal{E}^E, \mathcal{G}, \mathcal{G}^A, \mathcal{G}^X, \mathcal{G}^E, \mathcal{G}^M, \mathcal{F}, \mathcal{D}, \mathcal{L}, \mathcal{R}, \mathcal{W}, \mathsf{EN}, Etype, Attch, Excp, Econd, Fcond, Dflw, Act, Belto, ADlab, Elab)$ *where:*

- $\mathcal{O}$ *is a set of objects which can be partitioned into disjoint sets of activities* $\mathcal{A}$*, events* $\mathcal{E}$*, and gateways* $\mathcal{G}$*,*
- $\mathcal{A}$ *can be partitioned into disjoint sets of atomic activities (i.e. tasks)* $\mathcal{A}^T$ *and compound activities (i.e. subprocesses)* $\mathcal{A}^S$*,*
- $\mathcal{E}$ *can be partitioned into disjoint sets of start event* $\mathcal{E}^S$*, intermediate events* $\mathcal{E}^I$*, and end event* $\mathcal{E}^E$*,*
- $\mathcal{G}$ *can be partitioned into disjoint sets of parallel gateways* $\mathcal{G}^A$*, data-based exclusive decision gateways* $\mathcal{G}^X$*, event-based decision gateways* $\mathcal{G}^E$*, and exclusive merge gateways* $\mathcal{G}^M$*,*
- $\mathcal{F} \subseteq O \times O$ *is the control flow relation, i.e. a set of sequence flows connecting objects,*
- $\mathcal{D}$ *is a set of data objects associated with the process,*
- $\mathcal{L}$ *is a set of data object names,*
- $\mathcal{R}$ *is a set of roles designated to perform tasks or events within the process,*
- $\mathcal{W}$ *is a set of organisation groups involved in carrying out the process,*
- $\mathsf{EN} = \{\mathsf{message}, \mathsf{timer}, \mathsf{error}, \mathsf{conditional}\}$ *is a set of basic event type names,*

- *Etype* : $\mathcal{E} \to \mathsf{EN}$ *is a function which assign to each event an event type,*
- *Attch* : $\mathcal{E}^I \nrightarrow \mathcal{A}$ *is a function*[4] *which attaches an intermediate event to an activity indicating that the event may occur during the activity execution,*
- *Excp* : $dom(Attch) \to \mathbb{B}$ *is a function*[5] *which specifies, for an intermediate event that is attached to an activity, whether or not its occurrence interrupts the (normal flow of) activity execution,*
- *Econd* : $\{e \in \mathcal{E} | Etype(e) = \mathsf{conditional}\} \to C$ *is a function*[6] *which assigns to each conditional event a condition specified as a boolean function,*
- *Fcond* : $\mathcal{F} \cap (\mathcal{G}^X \times O) \to C$ *is a function which maps sequence flows emanating from data-based exclusive decision gateways to conditions thus determining if the associated sequence flow is taken during the process execution,*
- *Dflw* : $\mathcal{D} \to (\mathcal{A} \cup \mathcal{E}^S \cup \mathcal{E}^I) \times (\mathcal{A} \cup \mathcal{E}^I \cup \mathcal{E}^E)$ *is a function which specifies the fact of each data object being transferred from one activity/event to another,*
- *Act* : $(\mathcal{A}^T \cup \mathcal{E}) \to 2^{\mathcal{R}}$ *is a function which designates one or multiple roles eligible to perform a task or event,*
- *Belto* : $\mathcal{D} \to \mathcal{W}$ *is a function which assigns a role to an organisation group.*
- *ADlab* : $\mathcal{A} \cup \mathcal{D} \to \mathcal{L}$ *is a function which labels each activity or data object,*
- *Elab* : $\mathcal{E} \nrightarrow \mathcal{L}$ *is a function which labels an event (without mandating that each event is labelled).*

For a standalone BPMN process $\mathcal{M}$, if ambiguity is possible, we use $\mathcal{M}$ as subscripts to each element defined in the tuple $\mathcal{M}$. For example, $\mathcal{A}^S_{\mathcal{M}}$ refers to the set of subprocess invocation activities in $\mathcal{M}$. Next, we define the syntax of a core BPMN process which supports a hierarchical structure comprising a set of standalone BPMN processes.

Definition 2 (Core BPMN Process). *A core BPMN process is a tuple* $\mathcal{P} = (\mathcal{Q}, \mathcal{M}^{top}, \mathcal{S}^\diamond, map, HR)$ *where:*

- $\mathcal{Q}$ *is a set of standalone BPMN processes,*
- $\mathcal{M}^{top} \in \mathcal{Q}$ *is the top level process,*
- $\mathcal{S}^\diamond = \bigcup_{\mathcal{M} \in \mathcal{Q}} \mathcal{A}^S_{\mathcal{M}}$ *is the set of all subprocess invocation activities in* $\mathcal{Q}$*,*
- $map : \mathcal{S}^\diamond \to \mathcal{Q} \backslash \{\mathcal{M}^{top}\}$ *is a function which maps each subprocess invocation activity to a standalone BPMN process, and*
- $HR = \{(\mathcal{M}, \mathcal{M}') \in \mathcal{Q} \times \mathcal{Q} \mid \exists_{s \in \mathcal{A}^S_{\mathcal{M}}} map(s) = \mathcal{M}'\}$ *is a connected graph,*

Next, an *advice* process is a BPMN process in which all the start events of the (top-level) process are conditional events and there may be one or more *PROCEED* activities, and an *aspect* process comprises a number of advice processes that belong to the same aspect.

Definition 3 (Advice Process). *An advice process* $\mathcal{P}^a = (\mathcal{Q}, \mathcal{M}^{top}, \mathcal{S}^\diamond, map, HR, \mathcal{A}^{Tp})$ *is a core BPMN process that satisfies the following conditions:*

[4] $\nrightarrow$ indicates a 'non total function', i.e. there are values in the domain that do not have a corresponding value in the range

[5] $\mathbb{B}$ is the boolean set {`true`, `false`}.

[6] C is the set of all possible conditions. A condition is a boolean function, operating over a set of propositional variables, which evaluates to true or false.

– $\forall e \in \mathcal{E}^S_{\mathcal{M}^{top}}, Etype(e) = \textsf{conditional}$, *i.e. all start events in the top level process are conditional events, and*
– $\mathcal{A}^{Tp} = \bigcup_{\mathcal{M}\in\mathcal{Q}}\{a \in \mathcal{A}^T_{\mathcal{M}} | ADlab_{\mathcal{M}}(a) = \textsf{PROCEED} \wedge a \notin ran(Attch_{\mathcal{M}})\}$ *where* $\textsf{PROCEED}$ *is a preserved label for PROCEED activities.*

Definition 4 (Aspect Process). *An aspect process is a tuple* $\mathcal{P}^A = (\{\mathcal{P}^a_1, \mathcal{P}^a_2, ..., \mathcal{P}^a_n\}, \mathcal{AN}, Advice)$ *where:*

– $\{\mathcal{P}^a_1, ..., \mathcal{P}^a_n\}$ *is a set of advice processes,*
– $\mathcal{AN}$ *is a set of advice names, and*
– $Advice : \{\mathcal{P}^a_1, ..., \mathcal{P}^a_n\} \to \mathcal{AN}$ *is a bijective function which assigns to each advice process a unique advice name.*

Finally, a AOBPMN process comprises a main BPMN process and a set of associated aspect processes. The interactions between the main process and the aspect processes, which are defined in the *pointcut* specifications, are carried out at the corresponding *advised join point activities*.

Definition 5 (Core AOBPMN Process). *A core AOBPMN process is a tuple* $\mathcal{AP} = (\mathcal{P}, \mathcal{E}_A, \mathcal{A}_{JP}, \{\mathcal{P}^A_1, \mathcal{P}^A_2, ..., \mathcal{P}^A_n\}, \mathcal{CN}, Aspect, Pointcut)$ *where*

– $\mathcal{P} = (\mathcal{Q}, \mathcal{M}^{top}, \mathcal{S}^{\diamond}, map, HR)$ *is a core BPMN process,*
– $\mathcal{E}_A = \bigcup_{\mathcal{M}\in\mathcal{Q}}\{e \in dom(Attch_{\mathcal{M}}) \mid Etype_{\mathcal{M}}(e) = \textsf{conditional} \wedge Excp_{\mathcal{M}}(e)\}$ *is the set of intermediate conditional events attached to an activity in* $\mathcal{P}$*,*
– $\mathcal{A}_{JP} = \bigcup_{\mathcal{M}\in\mathcal{Q}}\{a \in \mathcal{A}_{\mathcal{M}} \mid \exists_{e\in\mathcal{E}_A} Attch_{\mathcal{M}}(e) = a\}$ *is a set of advised join point activities in* $\mathcal{P}$*,*
– $\{\mathcal{P}^A_1, ..., \mathcal{P}^A_n\}$ *is a set of aspect processes,*
– $\mathcal{CN}$ *is a set of aspect names,*
– $Aspect : \{\mathcal{P}^A_1, ..., \mathcal{P}^A_n\} \to \mathcal{CN}$ *is a bijective function which assigns to each aspect process a unique aspect name.*
– $Pointcut : \mathcal{E}_A \to 2^{Expr}$*, where* $Expr = \{\langle cond, pos, cn, an, order\rangle | cond \in C \wedge pos \in \{\textsf{before}, \textsf{after}, \textsf{around}\} \wedge cn \in \mathcal{CN} \wedge an \in \mathcal{AN}_{Aspect^{-1}(cn)} \wedge order \in \mathbb{Z}^+\}$*, is a function which relates an event* $e \in \mathcal{E}_A$ *to a set of expressions, and each expression specifies for the corresponding advised join point* $a \in \mathcal{A}_{JP}$*:*
  ∗ *the condition capturing the constrains for triggering an advice (cond),*
  ∗ *when the advice should be triggered in relation to a (pos),*
  ∗ *the aspect name (cn) and the advice name (an), and*
  ∗ *the precedence order of triggering this advice among the multiple advices associated with a (order).*

3 Case Study

In this section, we apply the proposed approach on a real case study from the financial domain. The case study demonstrates how the approach can be applied for modularizing cross-cutting concerns in a banking process model. In particular, the case confirms the relevancy of the precedence requirement.

The banking case was selected due to previous knowledge in that domain. To choose appropriate processes, i.e. fairly simple yet representative processes with at least a couple of cross-cutting concerns, we conducted an interview with a domain expert from a bank. For confidentiality reason, the bank asked to be remained anonymous. Two processes were selected. Here, we present one of them namely the *Change asset deal* process[7]. Detailed information about the process was derived through a follow-up interview with the same domain expert.

Generally, the assets of the bank are in two forms, cash and non-cash. Cash assets are either in the form of the account balances of the bank or the marketable securities. The `Change asset deal` process (see Figure 5) handles deals for exchanging assets of the bank from one currency to another. The process starts with a *back office employee* filling in a position sheet (`Fill position sheet` activity). Then, the *general manager* either confirms or denies the deal. If the sheet is denied, the process ends. If the position sheet is approved, it is archived. Then, a *junior dealer* makes the deal and fills in a deal slip. Next, both a *chief dealer* and the *general manager* sign the deal slip, after which the deal slip is archived.

After the deal slip has been archived, two parallel sets of activities are performed. On the one hand, the dealt amount of money is sent to the external partner of the deal. For this, first an *employee of the Swift department* provides a swift draft for sending the money. Then, for security purposes, the *dealer*, *chief dealer* and *general manager* sign the swift draft. Finally, an *employee of the Swift department* sends out the swift. On the other hand, the dealt amount of money should be received. This part starts when an *employee of the Swift department* receives an NT300 swift message. The *employee* sends this message to the general manager. The *general manager* makes an order to the Back office department and to the dealer to control the swift message. These orders are issued separately. When each one of them has been controlled, the messages are archived (separately). When the deal is made, a *back office employee* registers a voucher in the accounting system. Finally, the deal is archived.

Figure 5 shows the models including both BPMN and AOBPMN versions of the change asset deal process. We distinguish the following results of applying the aspect oriented modularization approach:

– The aspect oriented solution *documents additional knowledge* of a business processes in the model. This knowledge specifies the relation between cross-cutting concerns and specific activities. For example, in Figure 5b, two security concerns are associated to the `Send Swift` activity; while, this knowledge is missed in the model in Figure 5a. I.e., it is not obvious to which of the two activities, `Provide Swift Draft` or `Send Swift`, the security concerns are related.
– The aspect oriented approach presented here enables the *separation of several concerns.* For instance in Figure 5b, two different aspects are associated to the `Fill DealSlip` activity. In this way, security policy makers could easily define and change their related policy without changing the main process or the archiving concerns.
– This approach enables separation of concerns which have different orders for consideration. For example, a dealslip should be first confirmed and then

[7] A detailed description and analysis of both process can be find in [12].

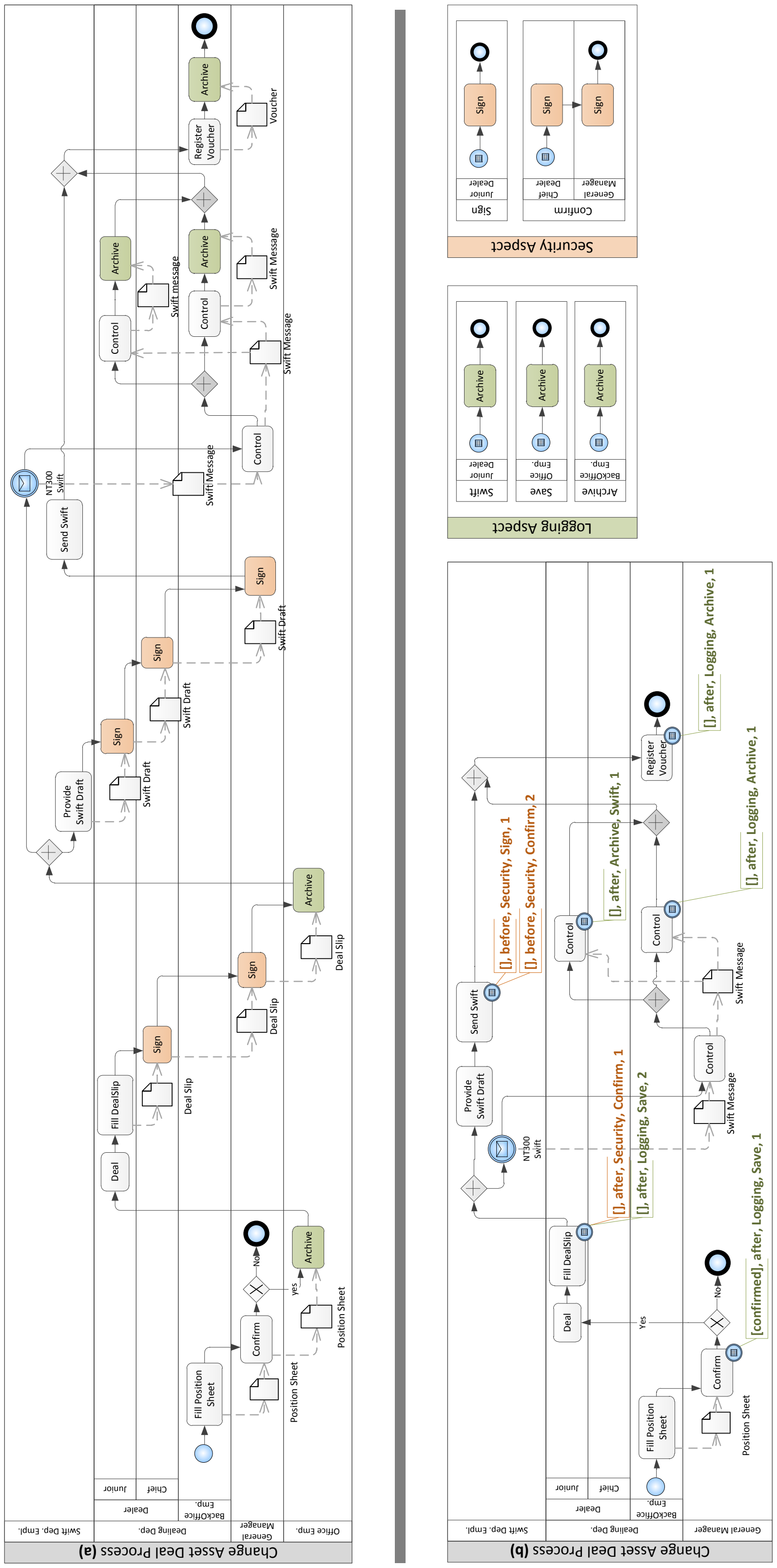

Fig. 5. The case study process: (a) traditional modelling; (b) AOBPMN modelling

archived (see `Fill DealSlip` activity). Other approaches [9, 1] are not able to separate archive concern in this example, because they do not capture precedence requirement in the definition of advices. Therefore, they could not separate all concerns from a business process. In contract, our approach supports full degree of separation.

- The aspect oriented model in Figure 5b is *less complex*, in terms of number of activities, than the model in Figure 5a. While the model in Figure 5a contains 20 activities, the model in Figure 5b contains 10 activities in the main process and 6 in the advice processes. Hence, communicating the aspect oriented model to business users is expected to be easier [17].
- Aspect orientation *increases the reusability* since policies are defined once and can be used many times. See for instance the use of `Confirm` advice in Figure 5b, where it is associated both to `Fill DealSlip` and `Send Swift` activities in the core business process.
- It also *facilitates the maintenance* of the system. If a policy is changed, it should be applied in one model rather through all involved business processes. E.g., if the `Confirm` concern is changed, the updates are reflected in the corresponding advice in Figure 5b rather than on a number of places in a process and even in several processes (Figure 5a).
- Last but not least, aspect oriented modelling *enables agile development* of business processes, due to faster response to changes, better adaptability and flexibility [4, 3]. This enables incremental development of business processes i.e. the ability to add or change aspects also sometime after the development of the main process.

4 Limitations

During the work we encountered two types of limitations: limitations on the approach and limitations of the case study. We report on these here.

The *limitations of the case study* are implied by the characteristics of the two processes that the case study was run on, i.e. small size processes containing approximately 20 activities. Because, this was our first case study, we aimed at studying small real processes deeply. For this reason we did not look at too large processes. Instead we selected processes that we could learn quickly and that were suitable for presenting to a less domain knowledgeable audience. A side effect from this was that the advice processes we separated out were too small, i.e. several of them containing one activity only. While this raises the question whether it is meaningful to model and maintain processes with single activities, we believe that this phenomenon needs to be studied further. We would need to study how frequent this occurs and whether the benefits of separating the concerns outweigh the disadvantages of separating and maintaining small advice processes. Naturally, these will be questions for a follow-up case study where the applicability of the approach on bigger processes should be explored.

A *limitation of the approach* follows from the fact that we do not transfer information about resources among the core process and advices. This implies that we may have to define one advice multiple times if a resource who should perform an activity is different. For example, in the case study, three almost identical advices were defined to capture the `Logging Aspect` (see Figure 5b). If this limitation is addressed, the three processes in the `Logging Aspect` would be reduced to one with the corresponding resource configurations. On the other hand, this resource limitation may not be a trivial thing when several resources are included in the execution of an advice.

Finally, while the formalisation of the syntax for aspect oriented business process modelling is presented in this paper, the semantics is formalised through Coloured Petri Nets and presented in [13]. As the work was carried out iteratively, the precedence requirement was confirmed through the case study and captured in the formal syntax of the approach. However, it also needs to be reflected in the formal semantics in future. The limitations outlined here present some lines for future work.

5 Related Work

The work on aspect oriented business process management is inspired from two fields, namely *software development* and *requirements engineering* that have been carried out primarily by the research groups of Charfi and Capelli, correspondingly. Within the software development, a lot of work has been done in the area of Aspect Oriented Programming(AOP) e.g., [15, 2, 16, 14, 11]. The ideas from AOP were initially transferred to the web services composition domain. Recently, they were also utilized in BPM area.

Charfi, et al. extended BPEL to support Aspect-Oriented Web Service Composition [6–8], and called the extension AO4BPEL. Although this work does not address people involvement in processes, it opened up further investigation of aspect orientation for process composition. For example, an extension of BPMN was proposed to support aspect oriented business process modelling, which is referred to as AO4BPMN [9]. Based on the terminology from AOP, AO4BPMN defines the notions for aspect, advice, pointcut, join point and proceed. AO4BPMN enables the modelling of aspects and advices through decomposition (i.e. in separate pools and swimlanes) and uses annotations on the join point activities, for relating the advices to the main process. We founded our work on the approach by Charfi et al. [9] and provide the following contributions. First, we fine-tune the application of aspect orientation terminology on BPMN. E.g. annotation (which were used to relate the advices with the main process) are replaced with intermediate conditional event, as these actually affect the flow of a Process [18]. By using the notion of intermediate conditional event, we ensure compliance with the BPMN specification [18]. Second, we provide a formalization of the syntax. Third, from the case study we identified the need for specifying Precedence between advices associated to the same joint point activity [12] and included mechanisms for expressing these in the formalization.

The other approach, by Cappelli, et al., introduces a notion for aspect oriented BPM [5, 1, 20, 21]. The work was initiated with a conceptualization of the terms used for aspect oriented process modelling language. It is based on BPMN and implemented in the CrossOryx editor. The approach [1] also outlines criteria for identifying cross-cutting concerns. This work was demonstrated with the application of aspect oriented business process modelling in a case. It also develops a conceptual model for Aspect Oriented Process Modelling Language (AOPML) [5]. It also describes criteria for identifying cross-cutting concerns. In addition, an extension of BPMN with aspect oriented notions and a representation language for pointcut specification [1] were defined and implemented. Later on the approach was adapted for relating aspect oriented business process models with goal [21] and service identification [20]. From this body of work, [1] is the most relevant to our work. In contrast to [1] we (i) do not add new notation to the BPMN standard, but use the existing elements for capturing aspect oriented models; (ii) we deal with precedence; and (iii) we remove ambiguity by providing formalization of the syntax. E.g., in [1] the application of "around" (called "during") is not clarified. In addition to the work by Charfi, et al. [9] and Cappelli, et al. [1], we also provide a formal semantics to our approach, which is presented in [13].

6 Conclusions

In this work, we elaborated on aspect oriented business process modelling and demonstrated its application for reducing the complexity of process models. We also outlined the need of precedence requirement, i.e. the need for specifying the execution order of multiple advices associated to the same activity. As an outcome, we extended the AO4BPMN approach [9] to support the definition of precedence. We also modified the approach to comply with the BPMN specification and presented a formal syntax. We validated the approach using a real banking case study, which illustrated how the approach separates concerns, prioritises the handling of concerns, reduces model complexity, increases reusability, documents additional domain knowledge, enables agile process development, increases flexibility and facilitates the maintenance of process models. In a sequel paper [13] we specify the formal semantics of the approach.

During our work, we reflected on a number of requirements presented for the software development domain. Our approach fulfils these requirements. It allows for the *modelling of multiple advices* associated to a process as well as multiple advices associated to a single activity. Aspects can be overlapping, i.e., *advices can be related to each other*. For example, an advice can contain activities associated to another advice. Finally, due to encapsulation, the definition of advices can be done *incrementally* and carried out by *different stakeholders*.

Some directions for future work include extending the semantics of the weaving mechanism to deal with precedence. Moreover, the approach shall be extended with a mechanism for passing resource information to advices. Finally, it is also desirable to carry out a more extensive case study to explore the benefits of aspect oriented business process modelling of larger processes and in other domains such as health care and the public sector.

Acknowledgements

This work is partially supported by National ICT Australia (NICTA). NICTA is funded by the Australian Government as represented by the Department of

Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. C. Cappelli at. all. Reflections on the modularity of business process models: The case for introducing the aspect-oriented paradigm. *Business Process Management Journal*, 16:662–687, 2010.
2. N. Belblidia and M. Debbabi. Formalizing AspectJ Weaving for Static Pointcuts. In *SEFM*, pages 50–59. IEEE Computer Society, 2006.
3. R. Booth. Agile manufacturing [management]. *Engineering Management Journal*, 6(2):105–112, april 1996.
4. T.F. Burgess. Making the Leap to Agility: Defining and Achieving Agile Manufacturing through Business Process Redesign and Business Network Redesign. *International Journal of Operations and Production Management*, 14(11):23–34, 1994.
5. C. Cappelli, J.C.S.P. Leite, T. Batista, and L. Silva. An aspect-oriented approach to business process modeling. In *Proceedings of the 15th workshop on Early aspects*, EA '09, pages 7–12, New York, NY, USA, 2009. ACM.
6. A. Charfi and M. Mezini. Aspect-Oriented Web Service Composition with AO4BPEL. In L-J. Zhang, editor, *ECOWS*, volume 3250 of *LNCS*, pages 168–182. Springer, 2004.
7. A. Charfi and M. Mezini. Hybrid web service composition: business processes meet business rules. In M. Aiello, M. Aoyama, F. Curbera, and M.P. Papazoglou, editors, *ICSOC*, pages 30–38. ACM, 2004.
8. A. Charfi and M. Mezini. AO4BPEL: An Aspect-oriented Extension to BPEL. In *World Wide Web*, pages 309–344, 2007.
9. A. Charfi, H. Müller, and M. Mezini. Aspect-Oriented Business Process Modeling with AO4BPMN. In T. Kühne, B. Selic, M.-P. Gervais, and F. Terrier, editors, *ECMFA*, volume 6138 of *Lecture Notes in Computer Science*, pages 48–61. Springer, 2010.
10. R.M. Dijkman, M. Dumas, and C. Ouyang. Semantics and analysis of business process models in BPMN. *Information & Software Technology*, 50(12):1281–1294, 2008.
11. W.-M. Ho, J.-M. Jézéquel, F. Pennaneac'h, and N. Plouzeau. A Toolkit for Weaving Aspect Oriented UML Designs. In *AOSD*, pages 99–105, 2002.
12. A. Jalali. Foundation of Aspect Oriented Business Process Management. Master's thesis, Stockholm University, 2011.
13. A. Jalali, P. Wohed, and C. Ouyang. Dynamic Weaving of Aspects for Business Process Management Systems. Technical report, Dept. of Computer and Systems Sciences, Stockholm University, March 2012.
14. J.-M. Jézéquel. Model Driven Design and Aspect Weaving. *Software and System Modeling*, 7(2):209–218, 2008.
15. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J-M. Loingtier, and J. Irwin. Aspect-oriented programming. In M. Aksit and S. Matsuoka, editors, *ECOOP'97 Object-Oriented Programming*, volume 1241 of *LNCS*, pages 220–242. Springer, 1997.
16. J. Klein, F. Fleurey, and J.-M. Jézéquel. Weaving Multiple Aspects in Sequence Diagrams. *Transactions on Aspect-Oriented Software Development*, 3:167–199, 2007.

17. J. Mendling, H.A. Reijers, and J. Cardoso. What Makes Process Models Understandable? In G. Alonso, P. Dadam, and M. Rosemann, editors, *BPM*, volume 4714 of *LNCS*, pages 48–63. Springer, 2007.
18. OMG. Business Process Model and Notation (BPMN), Version 2.0, 2011. availble on http://www.omg.org/spec/BPMN/2.0/PDF/, accessed Mar 2012.
19. H. Ossher and P. Tarr. Multi-Dimensional Separation of Concerns and the Hyperspace Approach. In Mehmet Aksit, editor, *Software Architectures and Component Technology*, volume 648, pages 293–323. Springer US, 2002.
20. A. Perin-Souza, C. Cappelli, F.M. Santoro, L.G. Azevedo, J.C.S. do Prado Leite, and T.V. Batista. Service identification in aspect-oriented business process models. In Jerry Zeyu Gao, Xiaodong Lu, Muhammad Younas, and Hong Zhu, editors, *SOSE*, pages 164–174. IEEE, 2011.
21. N. Santos, F. Jack, S. do Prado Leite, J. Cesar, C. Cappelli, T.V. Batista, and F.M. Santoro. Using Goals to Identify Aspects in Business Process Models. In *Proc. of the 2011 Int. Workshop on Early Aspects*, EA '11, pages 19–23, New York, NY, USA, 2011. ACM.