Business Process Extensions as First-Class Entities — A Model-Driven and Aspect-Oriented Approach

Heiko Witteborg, Anis Charfi, Mohamed Aly, and Ta'id Holmes

SAP Research Darmstadt
Bleichstr. 8, Darmstadt, Germany
firstname.lastname@sap.com

Abstract. To facilitate customer adaptation, extensibility constitutes an attractive design choice for providers of business software. However, most works in the context of business application extensibility are focusing on the code level. Lacking a conceptual foundation, such extensibility has shortcomings with respect to the understandability and the development of extensions. Addressing these issues, this paper presents the novel concept of business process extensions as first-class entities. We apply a model-driven approach which focuses on the business process layer and uses aspect-oriented modeling implicitly.

Keywords: process extensions, extensibility, business process modeling, model-driven

1 Introduction

Enterprise resource planning (ERP) and customer relationship management (CRM) are examples of business applications that support a wide range of standard business processes. However, the reality shows that companies in the same activity domain (e.g., retail or service) may have their own variants of these processes. When looking at diverse domains one observes that business processes are also performed in different ways. In fact, the standard processes delivered and implemented by the business software provider mostly require to be adapted to the particular needs of a certain company running it or to those of a certain domain. The adaptation of the business applications and their underlying business processes is typically done by implementing extensions. These extensions are either built by the customer or by a third party such as an independent software vendor (ISV). In fact, there is an ecosystem of ISVs and consulting organizations around leading business software providers, which specialize in configuring base software and building extensions to it. Extensions are owned by the ISVs whereas the base software is owned by the provider of the business software.

Extending business applications is a complex task for both the application provider and the extension developer (cf. [1]). Therefore, there is a need to raise the level of abstraction in order to ease understanding, developing, and managing extensions and systems that result from base software and extensions. The lack of appropriate models for extensibility leads to further problems, which are hard to tackle on the code layer. First, it is challenging to understand a business process that results after several extensions have been applied to the base software. This is because no appropriate models of the extensions

exist in code level approaches. Second, understanding which extensions perform which activities at which points in the base business process is complicated. Therefore, it is also difficult to identify potential conflicts between extensions without looking deep into the code. The same applies to the activation and deactivation of extensions, which, in current approaches, is a topic for expert technical users through complex low-level configuration and management tools. In this paper we address some of these problems using a model-driven and aspect-oriented approach for business process extensibility.

Our contributions are three-fold: First, we present a novel approach for defining business process extensions as first-class entities. This approach focuses on the business process layer and also includes a mechanism for composing (resp. decomposing) the extension models defined by the ISVs with (resp. from) the base business processes delivered by the business software provider. Second, we present a toolset supporting the proposed approach, which includes an extension editor and model repositories. Starting from a base process, the editor supports the in-place modeling of extensions as if the end user was directly modifying the business process. Internally, the toolset extracts the deltas modeled by the extender into business process extensions. This is transparent to the modeler (implicit aspect-oriented modeling). Third, we discuss the benefits of our approach and its application areas to further address the problems mentioned above.

2 Concepts and Tools for Business Process Extensibility

We propose a model-driven and aspect-oriented approach to business process extensibility. Focusing on the process layer, it enables participants in the ecosystem of the base software provider to extend the (read-only) process models of the business application, e.g., to add new activities to the process, to match the needs of a specific domain, or to integrate with other applications. The extended process is decomposed automatically and stored in a modular way: the resulting extension encapsulates the modifications that are associated with a certain concern, ready to be reapplied, or reused in another context.

We exemplarily illustrate our approach using a common CRM process for creating and approving sales quotations, as shown in Figure 2. In our extension scenario, this process is extended to integrate customer rating data delivered through an external credit reporting agency. Based on this additional information, an online shop could lower the risk of non-payment, e.g., by appropriately setting the payment terms.

In the following, we shortly present the conceptual basis of our approach using a business process extension metamodel. After that, we describe the main steps in the extension development approach. Finally, we present the toolset supporting our approach.

2.1 Business Process Extension Metamodel

Figure 1 shows a business process extension metamodel that formalizes the concepts that enable an ISV to model process extensions as first-class entities. It consists of two packages: The package `processmodel` contains the metamodel of the process definition language used to define the base process. We present a simplistic metamodel consisting of processes, edges, and vertices to exemplarily show the generic interrelation between

our extensibility concepts and the underlying process definition language [1]. The second part of Figure 1 is the package `extensionmodel`, which defines the concepts that can be contained in an extension.

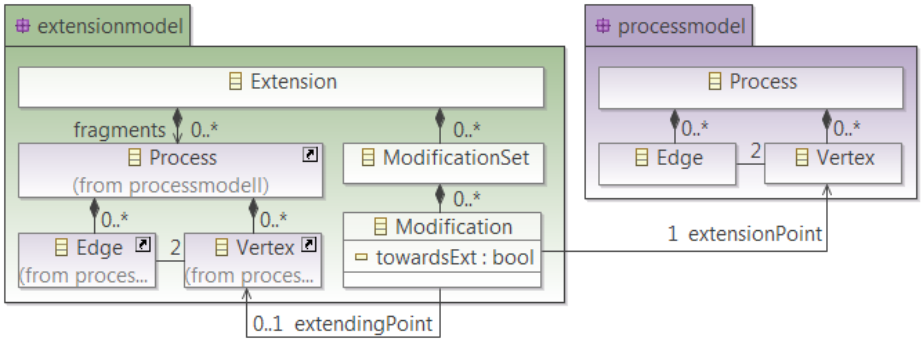

Fig. 1: Process Extension Metamodel

The *fragments* contain the new process elements introduced by the extension modeler. It uses the same constructs as the base process modeling language, following a symmetric approach to composition (cf. [10]), i.e., there is no conceptual distinction between base process and the extension part. This facilitates the intuitive graphical definition of extensions, and supports extension of extensions. Figure 2 shows the extended process of our motivating example, with a fragment consisting of a single task "Get external customer rating".

A *Modification* describes a single extension operation that is performed on a certain extension point. If `extendingPoint` is not specified, the modification describes a deletion. A deletion can be represented by hiding the targeted element, or by showing it grayed out. In the other case, the modification inserts and connects an extending element contained in one of the extension's fragments with the extension point contained in the base process (hereinafter called binding). Both extension point and extending point reference single identifiable nodes of the underlying process language, in our generic case, a single vertex. The binding can easily be represented as an edge in an extended process, with the `towardsExt` attribute specifying the direction of the edge.

Grouped using the *ModificationSet* concept, several bindings can be defined for the same combination of base process and fragment. In our running example, the additional task is integrated into the base process as an optional task after "Receive customer request"; thus, there is a forking edge between these two vertices and a merging edge targeting the successor vertex, as shown in Figure 2.

[1] However, our approach can be applied to any process definition language that formalizes a process as a flow of tasks, modeled as vertices interconnected via directed edges such as Business Process Modeling Notation [12] (BPMN) for example.

2.2 Extension Development Approach

The extension development is simplified by the fact that we use the same language for both the extended process and the base process. However, visualizing the extension part in the extended process, as well as extracting the extension when the process is saved requires the extended process model to hold additional information [2]; in particular, for each element, `modelId` as a reference to the base process or extension the element is assigned to, and `modificationOperation` indicating the modification operation.

Creation of Extended Processes Using a special editor, the ISV can specify modifications directly by drawing them in the diagram of the base process. The newly added or deleted elements are automatically annotated accordingly. Depending on the graphical notation of the underlying process language, these annotations can be used to enhance the visualization of the extended process, e.g., using different background and line colors. This enables the ISV to intuitively distinguish the base process from the extension parts. Figure 2 depicts the resulting extended process of our customer rating extension example.

Decomposition of Extended Processes When an extended process is saved, the extension is extracted automatically and the base process is left unmodified. Our decomposition algorithm uses the extension annotations to retrieve the list of modified elements and derive the fragments and modification sets. Listing 1 shows the extension artifact derived by decomposing our extended example process. It consists of the extension process with a single fragment and a modification set with two modifications.

Listing 1: Customer Rating Extension

```
<?xml version="1.0" encoding="UTF-8"?>
<extensionmodel:Extension xmi:version="2.0"˜xmlns:xmi="http://www.omg.org/XMI"
        xmlns:extensionmodel="http://extensionmodel/1.0"
        name="Customer Rating Extension">
  <fragments name="Customer Rating">
    <vertices name="Get external customer rating"/>
  </fragments>
  <modificationSets id="Quote2OrderModifications">
    <modifications towardsExt="true" extendingPoint="//@fragments.0/@vertices.0"
        baseProcessName="SalesQuotationCreation"
        extensionPointName="Receive customer request"/>
    <modifications extendingPoint="//@fragments.0/@vertices.0"
        baseProcessName="SalesQuotationCreation"
        extensionPointName="Check customer credit limit"/>
  </modificationSets>
</extensionmodel:Extension>
```

Composition of a Base Process and Extensions Base processes can be (re)composed with extensions: First, the modification sets relevant for the selected base process are retrieved. The corresponding fragment is inserted and annotated as `added`, and edges are created to represent the bindings. For a deletion, the extension point is annotated accordingly. Reapplying the extracted customer rating extension (cf. Listing 1) to the "Sales Quotation Creation" process restores the extended process shown in Figure 2.

[2] To store this information we annotate the extended process using lightweight language-specific extension constructs. We abstract from concrete annotation implementation in the following, assuming that the additional extension information is available.

2.3 Tooling

For the accomplishment of an end-to-end solution we implemented a tool for modeling business process extensions. This tool is supported by one or more model repositories for storing the base processes and the extensions. While managing extension models as first-class artifacts in business software systems, these repositories with the tools help to bridge the gaps between various phases of the engineering lifecycle. We refer to such a setting as a Model-Aware Service Environment [11] (MORSE). Moreover, as the repositories store conceptual, platform-independent models (PIMs) they can be utilized across different products of the business software provider's portfolio. Our tooling consists of the following components:

The base business process models of one or more business applications are stored in the central *Process Model Repository*. For reference and traceability purposes each model and model element in the MORSE is identifiable by a Universally Unique Identifier. The *Extension Repository* stores business process extension models separately from the process models although it might utilize the same model repository with the access rights appropriately set. One reason for this separation is the separate ownership of the respective models: the base process models are owned by the base software vendor whereas the process extension models are owned by the ISVs.

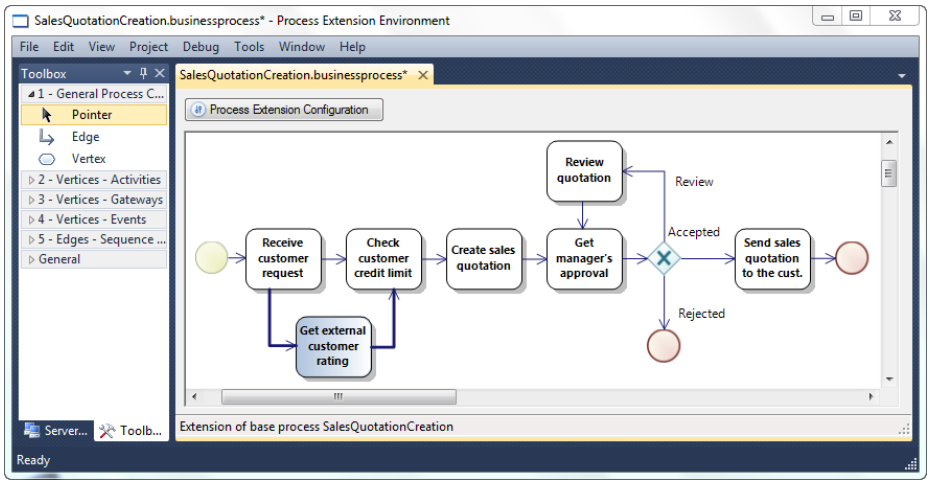

Fig. 2: Sales Quotation Creation Process with the Customer Rating Extension

The *Process Editor* enables the ISV to retrieve and view the base process models from the Process Model Repository, as well as extensions to be stored in the Extension Repository. Although having only read access, the editor allows to directly draw a new extension into the base process diagram or loading and editing an existing extension – in accordance with the modification restrictions defined by the base process owner. Moreover, the user can load and reuse an extension in another process context. Saving

the extended process automatically triggers the extraction of the extension as a first-class entity, stored in the Extension Repository. A screenshot of the tool is shown in Figure 2.

3 Discussion

In this section, we report on relevant design decisions and give some requirements from practice. We also report on several application areas of our approach.

3.1 Design Decisions and Requirements

In-Place Extension Modeling and Implicit Aspect-Oriented Support One requirement was to find and leverage a simple approach towards end users. To achieve that we introduced the technique of in-place extension modeling, which means that an ISV models the extension in our editor after loading the base business process as if he was directly modifying that business process. As a result, the extension modeler does not have to know or understand the metamodel and concepts presented in Section 2.1, unlike in aspect-oriented approaches such as [8]. This reduces the complexity for end users.

Process Extensions as Modular First-Class Entities An important requirement was to have a strict separation of the base process model and the extension models. The base processes are owned by the provider of the business software and extension models are owned by the ISVs. Further, we had the requirement of being able to compose multiple extensions from different ISVs with the same business process model. From this requirement, which is supported through the separation of ownership, we derived the design decision to have process extensions as modular first-class entities.

3.2 Application Areas

Code and Test Generation The design and development process of business software extensions may start now from the process extension models. These models – possibly in conjunction with other models – can serve for generating implementation artifacts for the extension. Furthermore, the process extension models can also serve as basis for generating tests to ensure the correct interplay of the extension with the base application.

Documentation The process extension models also serve as a documentation of an extension. In fact, every potential customer of an extension can now understand its effects and its interplay with the base business processes (e.g., in case of an extended process containing multiple extensions).

Extension Governance and Management The proposed approach enables a better understanding of the business process resulting after the deployment of several extensions. For example, a customer organization may buy the basis business application and gets the respective process models. When each ISV extending that software provides a process extension model as proposed in our approach a view with the complete process including the base process and extensions can be generated for the customer organization. A sample process view created by the composition mechanism is presented in Section 2. This view not only serves the understandability but also can ease the management of extensions and their dependencies and allows the detection of potential conflicts.

4 Related Work

Reference Process Models Balko et al. [2,3] propose a conceptual framework using provider defined reference workflow processes with extension points. Reference processes are standard business processes that are realized by the software provider following a workflow approach. The provider defines extension points where an extender can plug in his extensions in the form of process fragments. Extensions should then be managed and called at runtime by means of an extensibility framework (similar to an extension aware BPEL engine or an aspect-aware BPEL engine as in AO4BPEL [7]). That papers remain on the conceptual level providing only a high level description of the framework without the implementation of the presented concepts.

Adaptation Techniques Although the variety of business service types and natures is very broad (cf. [6]), a common characteristic among all services is the process of value creation between multiple parties in a flow of interaction/communication [9]. An extension of a business service therefore adds flow elements to the underlying process model, with the goal of increasing the service's business value in a specific scenario. The idea of extending, reusing, and adapting a base model to construct models for a concrete application domain is subject of the research area of reference modeling [4]. Concerning the adaptation techniques of reference models, vom Brocke [5] identifies five mechanisms such as configuration, aggregation, instantiation, specialization, and analogy. To ensure the applicability for our approach in a realistic business service network, the adaptation technique should allow (a) an automated adaptation process and (b) a maximum of flexibility for an ISV, while ensuring (c) the process owner's business need of having compliant and consistent core processes. The first requirement, an analogy-based extension is too vague to be automated. Requirement (b), both configuration and instantiation lack flexibility, because the modeler of the base process would have to foresee all potential extensions (or, at least, the extension points). For specialization, problems may arise in regard to requirement (c) as this technique empowers an ISV to fundamentally change the base process.

Aspect-Oriented Process Modeling In a previous work [8], we defined an aspect-oriented extension to BPMN called AO4BPMN. This extension allows to model aspects in business processes. Several differences exist between that work and the work presented in this paper. In AO4BPMN aspects are modeled explicitly by an expert user whereas here process extensions are extracted automatically by the toolset behind the scenes. Moreover, AO4BPMN extends the metamodel of BPMN with aspect-oriented concepts whereas in this work the same process modeling language is used for both the base process and the extensions. AO4BPMN is a powerful language with a rich join point model and pointcut language. In the current work the extension process elements and the base process elements are related through the binding. There is no explicit pointcut and no pointcut resolution (and thus no quantification). On the other hand AO4BPMN is complex for a business user and requires an understanding of aspect-oriented concepts.

5 Conclusion

Extensibility of business applications involves several stakeholders in the ecosystem of the business software vendor such as ISVs who build extensions on top of the base

software and customers who use the base software together with these extensions. All of these roles need to better understand the behavior of extensions, their effects on the base business process, and also potential conflicts between different extensions. In this paper we presented a model-driven and aspect-oriented approach and a toolset for defining business process extensions as first-class entities. Our approach addresses the limitations of state-of-the-art approaches and – through an implicit aspect-oriented modeling – makes extension development and management more accessible to business users. We also discussed several application areas that are enabled by this approach in our industrial context.

Acknowledgments The authors would like to thank Batbold Bilegsaikhan for his implementation of the customer rating extension and Wei Wei (危巍) for his contributions to the modeling environment prototype. The work presented in this paper was performed in the context of the Software-Cluster project Emergent. It was partially funded by the German Federal Ministry of Education and Research under grant no. 01IC10S01.

References

1. Aly, M., Charfi, A., Mezini, M.: On the extensibility requirements of business applications. In: International Workshop on Next Generation Modularity Approaches for Requirements and Architecture, NEMARA. ACM, Postdam, Germany (March 2012)
2. Balko, S., ter Hofstede, A.H., Barros, A.P., Rosa, M.L., Adams, M.J.: Controlled flexibility and lifecycle management of business processes through extensibility. In: Enterprise Modelling and Information Systems Architectures (EMISA), Proceedings of the Workshop. GI (2009)
3. Balko, S., ter Hofstede, A.H., Barros, A.P., Rosa, M.L., Adams, M.J.: Business process extensibility. Enterprise Modelling and Information Systems Architectures Journal (July 2010)
4. Becker, J., Beverungen, D., Knackstedt, R.: The challenge of conceptual modeling for product-service systems: status-quo and perspectives for reference models and modeling languages. Inf. Syst. E-Business Management 8(1), 33–66 (2010)
5. Brocke, J.V.: Referenzmodellierung - Gestaltung und Verteilung von Konstruktionsprozessen, Advances in Information Systems and Management Science, vol. 4. Logos (2003)
6. Cardoso, J., Barros, A.P., May, N., Kylau, U.: Towards a unified service description language for the internet of services: Requirements and first developments. In: IEEE SCC. pp. 602–609. IEEE Computer Society (2010)
7. Charfi, A., Mezini, M.: AO4BPEL: An aspect-oriented extension to BPEL. World Wide Web Journal: Special Issue: Recent Advances in Web Services 10(3) (March 2007)
8. Charfi, A., Müller, H., Mezini, M.: Aspect-oriented business process modeling with AO4BPMN. In: Modelling Foundations and Applications, 6th European Conference. LNCS, vol. 6138, pp. 48–61. Springer (2010)
9. Chesbrough, H., Spohrer, J.: A research manifesto for services science. Commun. ACM 49(7), 35–40 (2006)
10. Harrison, W.H., Ossher, H.L., Tarr, P.L.: Asymmetrically vs. symmetrically organized paradigms for software composition. Tech. Rep. RC22685 (W0212-147), IBM (Dec 2002)
11. Holmes, T., Zdun, U., Dustdar, S.: MORSE: A model-aware service environment. In: Proceedings of the 4th IEEE Asia-Pacific Services Computing Conference (APSCC). pp. 470–477. IEEE (Dec 2009)
12. Object Management Group, Inc.: Business Process Model and Notation (BPMN), Version 2.0 (Jan 2011), http://www.omg.org/spec/BPMN/2.0, [accessed in August 2012]