Process Lines for Automatic Workflow Development

Mario L. Bernardi[1], Marta Cimitile[2] and Fabrizio M. Maggi[3]
[1] *University of Sannio, Benevento, Italy*
[2] *Unitelma Sapienza University, Roma, Italy*
[3] *University of Tartu, Tartu, Estonia*

Keywords: Workflow Development, SOA, Business Process Lines.

Abstract: In some business environments, processes of different organizations are very similar to each other. This produces families of processes with common characteristics but also portions that vary according to the specific organization. Two emerging approaches can be adopted and combined to easily model, implement and update families of business processes: Software Product Line (SPL) and Service-Oriented Architecture (SOA). Our work suggests a framework to transfer the main peculiarities of the SPL to the SOA system development, in order to realize a SOA system line. Starting from the SPL concept, we introduce process lines, i.e., families of process models suitable for different customers or market segments. Moreover, we present an approach for the automatic generation of a SOA system starting from a process model. The combination of these approaches, can be used to easily develop a family of SOA systems each one appropriate for different context characteristics. In this work, an application of the proposed approach in a real project is also proposed.

1 INTRODUCTION

Often, organizations working in the same business environment execute their business processes in similar but slightly different manners (Mohammadi and Mukhtar, 2011). To provide them with appropriate workflow systems, it is necessary to tailor these systems according to different context factors (objectives, technologies, industrial standards, quality programs, budget, workers, tools, cultural factors). For this reason, it is necessary to define new approaches aiming at reducing the effort in modeling, updating and implementing a family of similar processes each one suitable to be used in a specific context. We propose an approach based on process lines and Service Oriented Architectures (SOA) supporting the reuse of process model components, the generation of process models based on combinations of such components (suitable for different contexts) and their rapid implementation into workflow systems. This proposal comes up from the following considerations:

- the concept of Software Product Line (SPL) is based on the idea that enterprises can implement software systems for different customers reusing software resources rather than developing the same software capabilities again. This concept can be extended to processes for modeling them in the perspective of process reuse and flexibility (Bernardi et al., 2012a).
- SOA offers a mature platform to rapidly implement a process model in pace with specific business needs (Welke et al., 2011).

The proposed approach consists of 2 phases: Process Variant Definition (PVD) and Automatic Workflow Development (AWD). The PVD phase is based on the concept of process line, which is, in turn, derived from the one of SPL. Through a process line, it is possible, starting from a basic process model, to define a process variant based on the customization and the adaptation of the basic process model according to different customer needs. We use BPMN (Business Process Modeling Notation) (Dijkman et al., 2011) to define the process models in a process line.

In the AWD phase, we transform a process variant into an executable SOA system through successive transformations aimed at making the process model understandable by an execution engine. In particular, we introduce an approach for transforming BPMN process models to BPEL (Business Process Execution Language) (Zhao et al., 2010).

Starting from a process line, it is possible, therefore, to automatically generate a SOA system line that automates the underlying process models. So, a process model can be customized using a process line and

Bernardi M., Cimitile M. and Maggi F..
Process Lines for Automatic Workflow Development.
DOI: 10.5220/0004492400410049
In *Proceedings of the 8th International Joint Conference on Software Technologies* (ICSOFT-EA-2013), pages 41-49
ISBN: 978-989-8565-68-6

the SOA system generated from it will be, in turn, customized. Also, if the underlying process line changes, it becomes straightforward to adapt the SOA system line accordingly. Finally, since a SOA system will result from the automatic translation of a process model, it will be in line with the underlying model without misalignment between models and their implementation.

The remainder of the paper is structured as follows: section 2 presents some related work; section 3 gives the overview of the proposed approach; section 4 illustrates the application of the approach in a research project; section 5 completes the paper providing some conclusive insights and final remarks.

2 RELATED WORK

The proposed approach is mainly based on the application of SPL principles to the business process domain and their integration with a workflow development method for business process implementation. The adoption of SPL in the context of process models was introduced in (Bernardi et al., 2012a) and can also be found in (Gimenes et al., 2008; Rolland and Nurcan, 2010). For example, in (Rolland and Nurcan, 2010) the authors propose to organize business processes as business process families and to manage variability and commonalities within the family in order to promote reuse and adaptability of business process models. To capture variability across the business processes of a family in an intentional manner, a modeling formalism is used.

A method that enables business managers to create and edit business rules is presented in (Auechaikul and Vatanawood, 2007). This approach allows the user to construct decision tables as a guideline and to expose them as web services. Similarly, in (Costello and Molloy, 2004), an approach decoupling business logic encoded as business rules from the applications in an organization is proposed.

Our approach to workflow development is based on a refinement of the algorithm first presented in (Ouyang et al., 2006; Ouyang et al., 2007). Here a mapping of BPMN process models to BPEL is introduced. Based on this algorithm, a process variant derived from a process line (according to a given context profile) can be translated into an executable workflow. We define a mapping between BPMN artifacts and BPEL variables and between BPMN activities and the web services to be orchestrated (including user interfaces to execute human tasks). Then, based on the algorithm presented in (Ouyang et al., 2006; Ouyang et al., 2007), we translate well-structured BPMN components into standard BPEL constructs and BPMN components that are not well-structured into control links and BPEL handlers.

3 PROPOSED APPROACH

The proposed approach consists of two main phases: Process Variant Definition (PVD) and Automatic Workflow Development (AWD). In the first phase, starting from a process line and from specific customer needs, a process variant is identified. In the second phase, the process variant is automated and transformed into a SOA system. These phases, shown in Figure 1, are detailed in the following subsections.

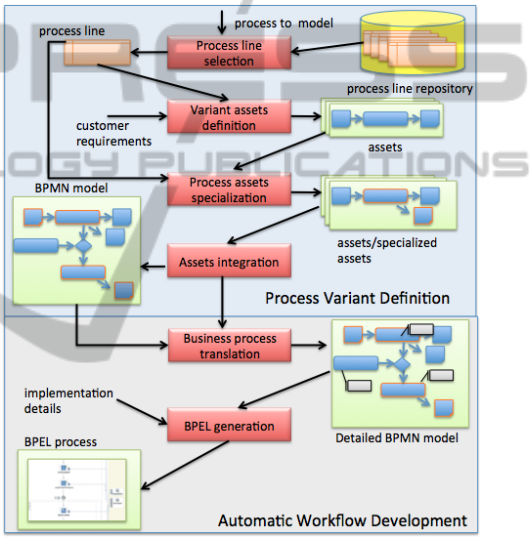


Figure 1: Proposed Approach.

3.1 Process Variant Definition

The PVD phase is focused on the concept of process line. Process lines realize the idea that Software Product Line principles can be transferred to the business process field. In (Schnieders and Puhlmann, 2006), (Bayer et al., 2006), a process line is defined as a set of similar process models sharing invariant assets and variant assets. The first formalization of a process line through a decision table is proposed in (Boffoli et al., 2009). In this work, the authors defined a *process asset* as an atomic reusable part of a process model including one or more activities with their inputs/outputs. A *process variant* is composed of invariant and variant assets integrated to obtain a process model suitable for a given context. Variant assets are selected to be included in a process variant depending on the specific context where the process

will be executed whereas invariant assets are always included in every process variant independently of the specific context profile.

The Process Variant Definition can be done through the following steps:

***Process Line Selection*:** A process line is selected basing on the type of the process to model. Some examples of process lines can be: Selling, Procurement, Marketing, Human Resource Management, Financial Management, Shipping. For each one of these processes, some invariant and variant assets can be defined. For example, an invariant asset for *Selling* is activity *Obtain order* and all the inputs/outputs related to it.

Variant Assets Definition: When a process line is selected, invariant assets and all candidate variant assets are specified. To extract a process variant from the process line, the suitable variant assets must be selected on the basis of the specific customer needs. For example, if we are interested in the process variant *Sell Product in electronic store with auction* we will use the process line *Selling* and we will select (among the others) the variant asset *Auction* and its related inputs/outputs.

Process Assets Specialization: Process assets (variant or invariant) are also specialized basing on the context. The specialization aims at specifying the behavior of each process asset through specialization actions. A specialization action can consist of the addition of input/output artifacts to an activity, the specialization of an artifact name, the specialization of an activity name, the addition of an attribute to an artifact (e.g., size, compilation guidelines, quality standards etc.), the addition of an attribute to an activity (e.g., required skills, tools, hardware or software resources etc.). For example, if we want to specialize *Selling* into *Sell Product in electronic store* we specialize activity *Obtain order* in *Obtain order in electronic store*.

Assets Integration: The process assets selected and specialized must be integrated into a process variant. This integration is obtained relating the activities through their inputs and outputs.

More formally, the Process Variant Definition can be described as a function associating to a specific context profile the corresponding process variant in the process line:

$$f : CP \rightarrow S \tag{1}$$

CP is the set of all the possible context profiles. **cp** is an element of CP and is represented as a vector of diversity factors in DF_i with $i = 1,\ldots,r$. Each DF_i is a factor characterizing a particular aspect of the environment and has a definition domain $[DF_i] = df_{i1}, df_{i2}, \ldots, df_{iq}$, where each df_{ij} with $j = 1,\ldots,q$ is an instance of DF_i. Therefore, $CP = [DF_1] \times [DF_2] \times \ldots \times [DF_r]$. **S** is the set of all the possible process variants of the considered process line. f can be detailed in this way:

$$f(cp) = \phi(\rho(cp,K), \rho(cp, \chi(cp))) \tag{2}$$

If A is the set of all the process assets associated to the process line, in (2):

- $K = \{ia_1, ia_2, \ldots, ia_n\} \subseteq A$ is the set of invariant assets;
- $\chi : CP \rightarrow A - K$ is the function associating to each fixed cp in CP the set of variant assets $\{va_1, va_2, \ldots, va_m\} \subseteq A - K$ according to the fixed context cp. $A - K$ is the set of the candidate variant assets associated to the process line;
- ρ is the function associating to a set of assets another set of assets specialized according to the fixed context profile through specializing actions. In particular, $\rho(cp, \{asset_1, asset_2 ... asset_p\}) = \{\rho_1(cp, asset_1), \rho_2(cp, asset_2), \ldots, \rho_n(cp, asset_p)\}$, where $\rho_1, \rho_2, \ldots, \rho_p$ are transformations specializing respectively assets $asset_1, asset_2, \ldots, asset_p$ according to the context profile cp.
- ϕ includes the integration rules useful to compose the assets.

3.1.1 Asset Selection and Specialization

The process line logic model can be implemented through a decision table system (Hong et al., 1999). A decision table is a tabular representation of a procedural decision situation, where the state of a number of conditions determines the execution of a set of actions. In general, a decision table is a table divided in four quadrants: conditions, conditional states, actions and rules. The table is defined so that each combination of conditions and conditional states corresponds to a set of actions to be executed. We need two types of tables for defining the process line logic model, one for Variant Asset Selection and another for Asset Specialization.

A **Variant Asset Selection Table** is structured as follows:

CONDITIONS. The quadrant contains the diversity factors DF_i with $i = 1,\ldots,r$ driving the variant assets selection. They are all the factors characterizing a given context.

CONDITIONAL STATES. The quadrant contains the possible values for each diversity factor: $[DF_i] = df_{i1}, df_{i2}, \ldots, df_{iq}$.

ACTIONS. The quadrant contains all the candidate variant assets (va) that can be selected to define a process variant.

RULES. The quadrant identifies the relationships between each context profile and the variant assets to realize the corresponding process variant.

An **Asset Specialization Table** allows us to specialize a process asset (variant or invariant) on the basis of a specified context profile, by executing a set of specializing actions. This decision table is different from the Variant Asset Selection table only for the actions and the rules quadrants. The actions quadrant contains the specialization actions (*sa*) used to specialize the asset according to the specified context profile. The rules quadrant identifies the relationships between each context profile and the specialization actions to be applied. A Variant Asset Selection Table and an Asset Specialization Table are shown in Figure 2.

3.2 Automatic Workflow Development

The AWD phase allows a process model defined in BPMN to be translated into a BPEL workflow. BPMN models are well versed for communication between domain experts. These models can be understood by both business users and developers and they are often given as requirement specifications for software development projects. To implement a BPMN model it is necessary to translate it in a language for process execution like BPEL. BPEL derives from the extension of imperative programming languages with constructs characteristic of service orientation and also defines a concrete engine for executing business processes. Even if BPMN and BPEL share several constructs there is no simple one-to-one mapping between them. Indeed, there are several constructs and components in BPMN diagrams that could not be easily translated to BPEL. Here, we adopt an extension of the automatic approach for mapping BPMN process graphs to BPEL processes described in (Ouyang et al., 2006).

The final outcome of the AWD phase is a BPEL application realizing the integration of the web services provided by all the parties involved. The Automatic Workflow Development can be done through the following steps:

- ***Process model specification:*** Due to the gap existing between a process modeling language and a process execution language, the original process model must be translated into a detailed process model including all the information needed for its implementation.
- ***BPEL generation:*** The detailed process model is automatically translated into an executable BPEL process.

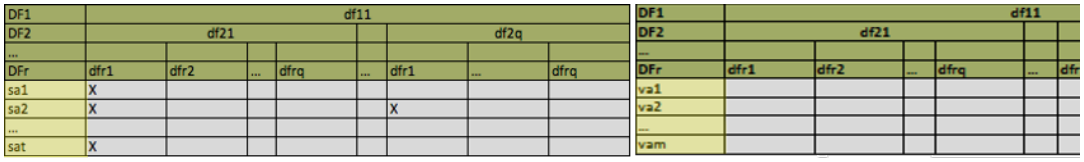
3.2.1 Process Model Enrichment

The translation of a BPMN process model into an executable BPEL process is possible by defining a mapping between BPMN and BPEL entities. In particular, we enrich the abstract model with few implementation details, thus generating a detailed BPMN model. BPMN activities can be translated to web services or human tasks depending on their properties. In particular, system-side activities that do not need interactions with human actors can be translated to web services. If a web service implementing an activity is already available, the detailed BPMN model will associate the activity to the corresponding WSDL URL. Conversely, if for some activities a web service is not available, the detailed BPMN model will associate them to a skeleton java class. Activities requiring data and the interaction with at least a (human) system user will be mapped to adequate human tasks. For the implementation of the user interfaces we use the BPEL extension BPEL4People (Kloppmann et al., 2005).

We map a BPMN artifact to a BPEL variable whose name corresponds to the artifact label. Moreover, the variable type should match the structure of the class which represents the artifact. However, information contained in the class definition is not always sufficient to automatically translate the class into a BPEL type. Consequently the detailed BPMN model must also provide a mapping between each class in the abstract process model and the corresponding BPEL types in the executable process. A BPEL type can be simple, or complex (equivalent to Java data structures defined as a combination of one or more simple or complex types). The detailed model will include the (simple or complex) BPEL type corresponding to each (simple or complex) artifact in the abstract model. For instance, in the Selling process the artifact Order can correspond to the Java data structure:

```
public class Order {
     public int sellerID;
     public int buyerID;
     public int productID;
     public int quantity;
     public float price;
}
```

Note that, when an activity is implemented using a web service, the corresponding WSDL file is already provided with the BPEL types associated to the web service input/output variables. In this case, the detailed BPMN model should only contain the mapping between the input/output artifacts of the BPMN activity and the (existing) BPEL types extracted from the specified WSDL file.



DF1	df11							
DF2	df21				df2q			
...								
DFr	dfr1	dfr2	...	dfrq	...	dfr1	...	dfrq
sa1	X							
sa2	X					X		
...								
sat	X							

DF1	df11				
DF2	df21				
...					
DFr	dfr1	dfr2	...	dfrq	...
va1					
va2					
...					
vam					

Figure 2: Asset Selection (left) and Specialization (right) Table.

```
<complexType name="Order">
<sequence>
<element name="sellerID" type="xsd:int"/>
<element name="buyerID" type="xsd:int"/>
<element name="productID" type="xsd:int"/>
<element name="quantity" type="xsd:int"/>
<element name="price" type="xsd:float"/>
</sequence>
</complexType>
```

Figure 3: XML Code Translation.

3.2.2 BPEL Generation

If a detailed BPMN model is built including the information mentioned above, an automatic translation algorithm is able to automatically generate a BPEL application implementing the underlying model. We first execute a pre-processing phase to import into the generated BPEL workflow the complex types needed for its execution. Complex types are imported from the existing WSDL files implementing the different activities of the business process. Complex types which cannot be extracted from the available WSDL files must be created using the information contained in the detailed BPMN model. For instance, the above Order type is translated into the code of Figure 3.

A recursive algorithm has been implemented to deal with nested complex types. After the pre-processing phase, we start the translation of the entire BPMN (detailed) model. The adopted algorithm is introduced and largely discussed by (Ouyang et al., 2006; Ouyang et al., 2007). The approach distinguishes between well-structured BPMN components (that can be directly mapped to BPEL structured activities) and components that are not well structured. A formal definition of well-structured component is given in (Ouyang et al., 2006). The algorithm tries to map to control link-based BPEL code components that are not well structured but acyclic. Components that are not well-structured and, also, cannot be translated using control links (e.g., if they contain unstructured cycles) are mapped to BPEL code based on event handlers.

4 CASE STUDY

In order to validate the feasibility of our approach, we applied it for the definition and implementation of an online selling process. In particular, we used our approach to model the selling process of a typical online shopping auction store.

At this aim, first, we created a process line for selling processes. Then, we used this process line to model a selling process variant. Finally, the obtained process variant was implemented in BPEL with our automatic workflow development approach.

4.1 Selling Process Line Definition

Starting from the analysis of the literature (Malone et al., 2003; Bernardi et al., 2012b; Bernardi et al., 2012c), we realized a process line for selling processes. The characteristics of this process line are synthesized in Tables 1, 2 and 3. Table 1 includes the invariant assets, i.e., the assets that are common to all the selling process variants.

For example, each process variant contains an asset corresponding to activity *Obtain Order* with input *Order request* and output *Order*.

Table 2 is obtained taking into consideration the possible application contexts where a selling process can be applied. In this table, we list a number of diversity factors and their possible values. For example, we introduce the diversity factor *Sell How* and its possible values: *Sell via physical store*, *Sell via electronic store*, *Sell via face to face*, *Sell via direct mail*, *Sell via email/fax*, *Sell via television*. This means that the selling process will be executed differently based on how the items are sold. For example, we will select different variant assets (and we will specialize the assets differently) if the items are sold in a physical store or in an electronic store. Finally, in Table 3, we indicate the candidate variant assets, which are the assets that can be included in one or more process variants. For example, we can have, in some process variants, the asset *Share out goods* with input *Goods* and *Shared out equipments* and output *Shared out goods*.

Starting from the tables just described, we create the Variant Asset Selection Table. An extract of this

Table 1: Invariant Assets.

	Activity	IN	OUT
ia_1	Obtain order	Order request	Order
ia_2	Deliver	Order	Delivered goods
ia_3	Receive payment	Order	Receipt

Table 2: Diversity Factors.

Diversity Factors	Values
Sell How	Sell via physical store Sell via electronic store Sell via face to face Sell via direct mail Sell via email/fax Sell via television Sell via telemarketing
Sell What	Services Product Store
Auction	Y , N
Advance Payment	Y , N
Quality control	Y , N
Selling Suggestions	Y , N
Seller	Direct , Indirect
Advertising	Y , N
Customer Assistance	Y , N

table is shown in the top of the Figure 5.[1] It specifies the rules for associating to each possible context profile the related variant assets. These have to be composed with the invariant assets to obtain the process variant specific for the given context profile.

For instance, we have that for the context profile cp*= (Sell via electronic store, Product, Y, Y, Y, N, Indirect, Y, Y) the variant assets to be selected are (see column 101 of the table): $va_1 =$ *Register Seller*, $va_2 =$ *Arrange store displays*, $va_3 =$ *Auction*, $va_4 =$*Check Quality*, $va_5 =$*Register Auction Result*, $va_6 =$*Identify potential customer needs*, $va_7 =$*Identify potential customers*, $va_8 =$ *Manage customer relationship*.

To define the Asset Specialization Table of the process line we have to identify the specializing actions needed for each possible context profile. Figure 4 (bottom) includes some of the rules for associating to each context profile, the actions to be executed to specialize the behavior of the (variant and invariant) assets of the process line. For instance, for the context profile cp* considered before (see column 52), we need to specialize the activities: *Deliver* in *Deliver product* and *Obtain order* in *Obtain order in electronic store*. Moreover, we have to add inputs *Receipt* and *Shipping paper* to *Deliver Product*. To obtain the process variant for a given context profile, we have to integrate variant and invariant assets. This integration is obtained based on the inputs and the outputs of each asset.

[1]In this case study, we use Prologa (Vanthienen and Wets, 1995), a tool supporting the decision table modeling.

4.2 Process Variant

The process line introduced in the previous section can be used to model different process variants of a selling process. In particular, here, we describe the application of the process line to automate a selling process with online auction. The considered process owner sells *products* in an *electronic store* through *auctions*. The organization is an *indirect seller*, i.e., it offers a virtual market place and some additional services to support the interaction between the direct seller and the customers. In particular, a registered seller can send an auction proposal. The process owner *certifies the quality* of the products and starts the auction. When the auction is over, the organization offers an online payment service. After the payment, the (direct) seller sends the products to the process owner that, in turn, sends them to the customers. The process owner also offers *customer assistance* and *advertising services*. The process owner needs to model the selling process as just described and to automate it into a SOA system. The system must integrate the services of the indirect seller with the ones provided by the direct sellers and by their shipping partners. Starting from the process owner requirements, we specify the context profile cp* = (Sell via electronic store, Product, Y, Y, Y, N, Indirect, Y, Y). Using the Variant Asset Selection table, we identify the list of variant assets needed for the given context profile. Through the Asset Specialization table, we identify the list of specialization actions to be applied to the process assets. The specialized assets are integrated in the process variant shown in Figure 5. In the figure, invariant assets are colored in blue, variant assets are colored in yellow, while the specialized artifacts/activities are highlighted with a red border.

4.3 BPEL Workflow Generation

Using our workflow development approach, we have generated the detailed process model starting from the process variant identified through the process line.

Table 3: Variant Assets.

	Activity	IN	OUT
va_1	Share out goods	Goods, Shared out equipments	Shared out goods
va_2	Register Seller	Entry document	Registered entry document
va_3	Register Alternative Product	Registered entry document	Suggestions
va_4	Arrange store displays	Registered entry document	Arranged goods
va_5	Auction	Arranged goods	Customer ID, Order request
va_6	Check quality	Arranged goods	Quality certification
va_7	Register Auction Result	Customer ID, Order, Registered entry document	Registered Order
va_8	Identify customers needs	Registered Order	Marketing report
va_9	Identify potential customers	Customer request	Marketing report
va_{10}	Inform potential customers	Marketing report	Advertising initiative
va_{11}	Manage customer relationships	Marketing report, Customer ID	Customer support

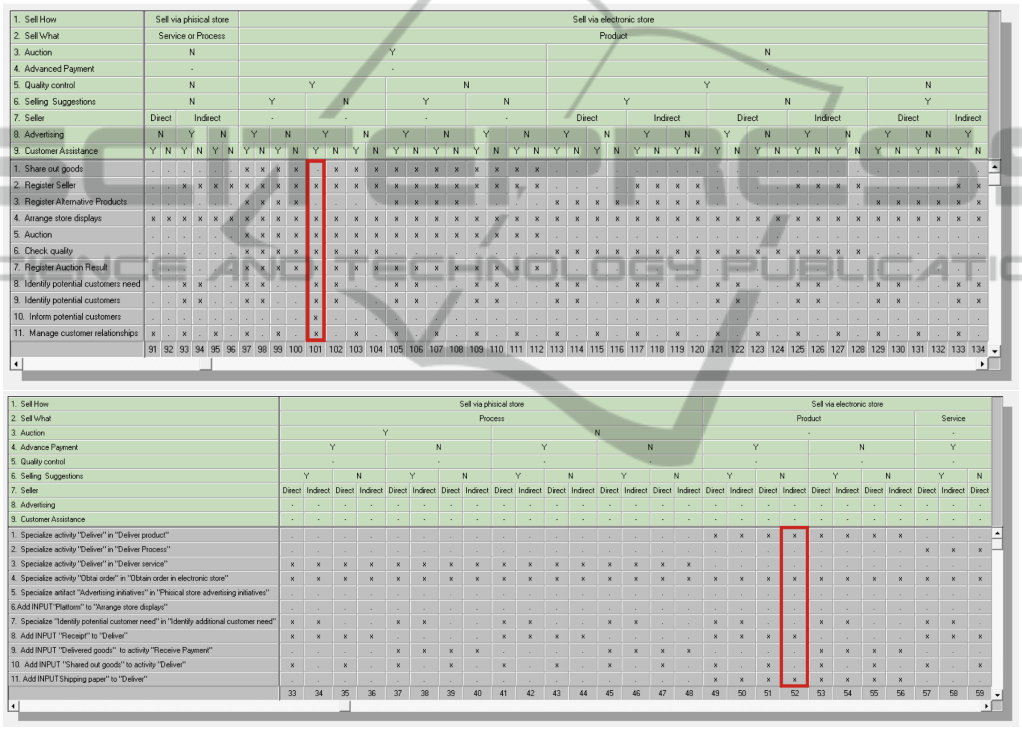


Figure 4: An example of Variant Asset Selection Table (top) and Asset Specialization Table (bottom).

This new model has been obtained enriching the process model with implementation details. We have implemented a prototype tool to support the detailed process model generation. This tool has been realized as an Eclipse add-inn, a graphical UML design and business analysis tool for modeling, documenting, building and maintaining object-oriented software systems. Starting from the detailed process model previously created, through the algorithm described in (Ouyang et al., 2006), we generated the workflow BPEL implementing the underlying process variant. The process structured composition tree, obtained by applying the algorithm to the portion of the BPMN model highlighted with a red dashed line in Figure 5, is shown in figure Figure 6. Using such a model as a guide, the corresponding BPEL code is generated as shown in Figure 7.

5 CONCLUSIONS

This paper extends the best practices of SPL to the SOA systems development. In particular, we introduce here the concept of process line aiming at combining a set of reusable process assets into a BPMN process variant suitable to be used in a specific context. In addition, we show an approach for implementing a SOA system starting from the obtained BPMN process model. This system will be, in turn, suitable to be used in the context previously speci-

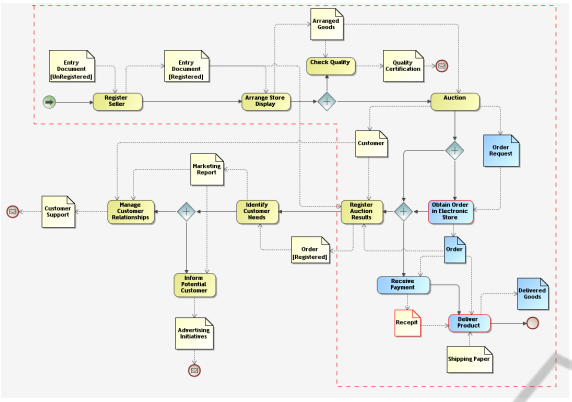


Figure 5: Online Selling Process Variant.

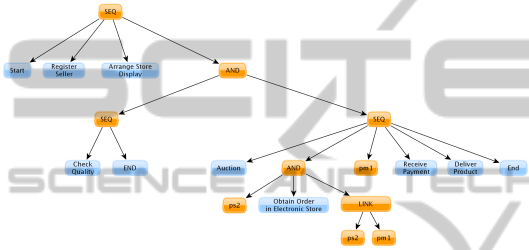


Figure 6: Excerpt of BPEL Process composite Structure.

```
<!-- ================================================================= -->
<!-- ORCHESTRATION LOGIC                                               -->
<!-- Set of activities coordinating the flow of messages across the    -->
<!-- services integrated within this business process                  -->
<!-- ================================================================= -->
<bpel:sequence name="main">
    <!-- Receive input from requester.
         Note: This maps to operation defined in Auction.wsdl
         --><!-- Generate reply to synchronous request -->


    <bpel:sequence name="Sequence">
        <bpel:invoke name="Start"></bpel:invoke>
        <bpel:invoke name="Register Seller"></bpel:invoke>
        <bpel:invoke name="Arrange Store Display"></bpel:invoke>

        <bpel:flow name="Flow"><bpel:sequence name="chkquality">
                <bpel:invoke name="Check Quality"></bpel:invoke>
                <bpel:exit name="Exit"></bpel:exit>
            </bpel:sequence><bpel:sequence name="auction">
                <bpel:invoke name="Auction"></bpel:invoke>
                ...
            </bpel:sequence></bpel:flow>
    </bpel:sequence>

</bpel:sequence>
</bpel:process>
```

Figure 7: Small Excerpt of BPEL Generated Code.

fied. We show an application of our approach to a case study for the automation of an online auction selling process. In this scenario, we have evaluated the feasibility of our approach. In particular, we carried out a qualitative evaluation in which two developers were asked to model and implement the same process. The first developer was asked to use our approach whereas the second developer was asked to use a only standard CASE tools. According to the data gathered, we found out that the proposed approach reduces the time required for modeling and workflow development of around 65%. A limit of the proposed approach is related to the effort needed to model the process lines (200h were required for the Selling process line described in this work). However, when the decision table system has been created no much effort is needed to maintain it, due to the availability of mature tool support for decision table modeling.

REFERENCES

Auechaikul, T. and Vatanawood, W. (2007). A development of business rules with decision tables for business processes. In *TENCON 2007 - 2007 IEEE Region 10 Conference*, pages 1 –4.

Bayer, J., Kose, M., and Ocampo, A. (2006). Improving the development of e-business systems by introducing process-based software product lines. In *Proceedings of the 7th international conference on Product-Focused Software Process Improvement*, PROFES'06, pages 348–361, Berlin, Heidelberg. Springer-Verlag.

Bernardi, M. L., Cimitile, M., Lucca, G. A. D., and Maggi, F. M. (2012a). M3d: a tool for the model driven development of web applications. In *WIDM*, pages 73–80.

Bernardi, M. L., Cimitile, M., Lucca, G. A. D., and Maggi, F. M. (2012b). Using declarative workflow languages to develop process-centric web applications. In *EDOC Workshops*, pages 56–65.

Bernardi, M. L., Cimitile, M., and Maggi, F. M. (2012c). Model driven development of process-centric web applications. In *ICSOFT*, pages 340–346.

Boffoli, N., Cimitile, M., and Maggi, F. M. (2009). Managing business process flexibility and reuse through business process lines. In *ICSOFT (2)*, pages 61–68.

Costello, C. and Molloy, O. (2004). Orchestrating supply chain interactions using emerging process description languages and business rules. In *Proceedings of the 6th international conference on Electronic commerce*, ICEC '04, pages 21–30, New York, NY, USA. ACM.

Dijkman, R. M., Hofstetter, J., and Koehler, J., editors (2011). *Business Process Model and Notation - Third International Workshop, BPMN 2011, Lucerne, Switzerland, November 21-22, 2011. Proceedings*, volume 95 of *Lecture Notes in Business Information Processing*. Springer.

Gimenes, I., Fantinato, M., and Toledo, M. (2008). A product line for business process management. In *Software Product Line Conference, 2008. SPLC '08. 12th International*, pages 265 –274.

Hong, T.-P., Wang, S.-L., and Tsai, J.-S. (1999). Integrating multiple knowledge sources using decision tables. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, volume 2, pages 142 –147 vol.2.

Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., and Trickovic, I. (2005). Ws-bpel extension for people–bpel4people. *Joint white paper, IBM and SAP*, 183:184.

Malone, T. W., Crowston, K., and Herman, G. A., editors

(2003). *Organizing Business Knowledge: The MIT Process Handbook*. The MIT Press, 1st edition.

Mohammadi, M. and Mukhtar, M. (2011). Soa-based business process for supply chain management. In *Software Engineering (MySEC), 2011 5th Malaysian Conference in*, pages 213 –216.

Ouyang, C., Dumas, M., ter Hofstede, A. H. M., and van der Aalst, W. M. P. (2006). From bpmn process models to bpel web services. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '06, pages 285–292, Washington, DC, USA. IEEE Computer Society.

Ouyang, C., Verbeek, E., van der Aalst, W. M. P., Breutel, S., Dumas, M., and ter Hofstede, A. H. M. (2007). Formal semantics and analysis of control flow in ws-bpel. *Sci. Comput. Program.*, 67(2-3):162–198.

Rolland, C. and Nurcan, S. (2010). Business process lines to deal with the variability. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1 –10.

Schnieders, A. and Puhlmann, F. (2006). Variability mechanisms in e-business process families. In *Proc. International Conference on Business Information Systems (BIS 2006*, pages 583–601.

Vanthienen, J. and Wets, G. (1995). Integration of the decision table formalism with a relational database environment. *Inf. Syst.*, 20(7):595–616.

Welke, R., Hirschheim, R., and Schwarz, A. (2011). Service-oriented architecture maturity. *Computer*, 44(2):61 –67.

Zhao, W., Huang, Y., Yuan, C., and Wang, L. (2010). Formalizing business process execution language based on petri nets. In *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, pages 1 –8.