A Software Process Line for Service-Oriented Applications

Cleiton Garcia
Pontifícia Universidade Católica do Paraná (PUCPR)
Rua Imaculada Conceição, 1155, Curitiba, Brazil
+55 47 9609-1191
cleitonsg@gmail.com

Marco Paludo
Pontifícia Universidade Católica do Paraná (PUCPR)
Rua Imaculada Conceição, 1155, Curitiba, Brazil
+55 41 8815-6853
marco.paludo@pucpr.br

Andreia Malucelli
Pontifícia Universidade Católica do Paraná (PUCPR)
Rua Imaculada Conceição, 1155, Curitiba, Brazil
+55 41 9994-2492
malu@ppgia.pucpr.br

Sheila Reinehr
Pontifícia Universidade Católica do Paraná (PUCPR)
Rua Imaculada Conceição, 1155, Curitiba, Brazil
+55 41 9997-4083
sheila.reinehr@pucpr.br

ABSTRACT

The management of processes and systems is a complex and time-consuming activity for organizations and also an ongoing Information Technology (IT) challenge. Among the different approaches for bringing flexibility to the business processes and systems are Service-Oriented Architecture (SOA) and Business Process Management (BPM). The SOA approach has become popular providing services and interfaces, enabling integration of heterogeneous and distributed platforms and BPM leverages the cycles of improvements, control and evaluation of business processes. BPM and SOA should work together aiming at improving business processes and evolving systems architecture. One main problem to apply BPM and SOA is the lack of established processes and this work proposes a software process line in order to simplify variability control and enable the instantiation of new development process applying BPM and SOA. It also aims at developing an environment to support the proposed software process line in order to automate the process, integrating industrial tools with one specifically developed to perform the transformation of UMA models into BPM notation. The main contribution of this work is the definition of the software process line for engineering service-oriented products. It is highly relevant to software industry since software process lines lacks experiments, practices and tools.

Categories and Subject Descriptors

• Software and its engineering, Software creation and management, and Software development process management

General Terms

Management, Measurement, Documentation, and Design.

SAC'15, April 13-17 2015, Salamanca, Spain
Copyright 2015 ACM 978-1-4503-3196-8/15/04…$15.00
http://dx.doi.org/10.1145/2695664.2695743

Keywords

Software Process Lines, SPL, SOA, BPM, Web-based services.

1. INTRODUCTION

A software reuse program requires an organizational structure and tools to promote, manage and maintain the practice of reuse [1]. In the late 1990s, the Software Engineering Institute (SEI) introduced the Software Product Line (SPL) approach. Since then, the SPL has been used to establish systematic reuse. An SPL can be defined as a set of systems that share a set of common features and functionality and are managed to meet a particular market segment based on a common set of assets [2].

From 1990 on, mergers and acquisitions began to occur more quickly in several sectors increasing the problems related to integration of incompatible IT systems [3]. Standardized system packages such as Enterprise Resource Planning have been proposed as a solution to this problem however, other systems were also necessary such as: Customer Relationship Manager (CRM), Supplier Relationship Manager (SRM), Supply Chain Manager (SCM), Manufacturing Execution System (MES) and Business Intelligence (BI).

In this context, the Service-Oriented Architecture (SOA) emerged and organizations are adopting this approach in order to establish the reuse of services level mapping business processes. The SOA goal is to connect the business world to the Information Technology world to make both more efficient [4]. SOA encapsulates the functionalities improving the system integrations and offering business process transparency. To coordinate the execution of the SOA system, the Business Process Management (BPM) system plays an important role. BPM supports the use of management methods, techniques and software to design, establish, control, and analyze operational execution processes involving people, organizations, applications, documents, and other sources of information [5]. SOA organizes the business process tasks as services and BPM coordinate the workflow of human and automated tasks. BPM supports the analysis and optimization of processes, supporting parallel operations, collaboration and distribution of business processes in activities inside and outside of the organization [6].

In order to apply reuse concepts to the software process area, the SPL was adapted to be used with software process. In 1996 the first work concerning software process families or software process lines (SPrL) was published, presenting how to analyze and create software process elements in families [7], in order to reduce efforts in definition and implementation. After a decade, the SPrL was leveraged incorporating techniques, mechanisms and tools for managing variability in software process. Reduction of tailoring efforts and learning time of the software process for each project are the main benefits of this approach [8].

The software process management and tailoring is also a problem regarding SOA and the industry lacks for standardized processes for service-oriented tasks. In order to succeed, SOA demands some activities, such as:

- Service query is the most important activity and requires a search mechanism that identifies services from the user's and business requirements [9];
- Analysis of existing processes and services and, whenever necessary, establish a new systematic process to create new versions or new composed services;
- Leverage the business processes view, measuring and orchestrating; and,
- Apply an iterative process to refine the requirement specification [10].

The objective of this work is to define a software process line for Service-Oriented Modeling and BPM, addressing variability to support the development of applications and composed services from existing services.

Section 1 presented an overview of the proposed work. Section 2 presents some concepts and related works, also presenting the adopted notation. Chapter 3 describes de process line conception with some application results. Chapter 4 evaluates the proposed SPrL and tools, following the method proposed by [11]. The last section presents the conclusions and future works.

2. RELATED WORKS

Service Oriented Architecture (SOA) is the evolution of distributed computing based on the high level messaging exchange designed according to business process. SOA can be defined as a set of tools to integrate business process and support to the IT infrastructure with security, standardized components as services, that can be reused and combined to meet business changes [3]. A case study conducted at Brazilian financial institutions identified the emphasis on projects for implementing and promoting reuse through SOA, where 60% of the participants indicated their intention to adopt service-orientation [12].

Service-orientation is highly suited for reuse and has facilitated and motivated acquisitions and sales of services. According to [13], in "SOA Is Evolving Beyond Its Traditional Roots", it is thanks to SOA's success that initiatives such as Software as a Service (SAAS), BPM, Mobility Computing and Cloud Computing are viable. Nowadays services are being supported through service-orientation in domains such as knowledge of electronic transport, credit card validations and payments, post office enquiry, electronic invoices, and several other everyday services.

2.1 SOA Layers

The solutions project based on SOA suggests a layer division with well-defined responsibilities. These layers aim at isolating the different levels of complexity, such as software construction parts and activities. Reuse is also promoted through different layers, identifying components in accordance with the granularity of each layer. The main layers of Service Oriented Architecture as presented by [14] [15] [3], from the consumers' perspective toward the providers' perspective are presentation, business process, service, business components and backend.

2.2 SOA Maturity Models

The Service Integration Maturity Model (SIMM) was conceived buy IBM considering the SOAMM model, incorporating the Open Group Service Integration Maturity Model [16]. The OSIMM defines a script for the incremental adoption of SOA, aiming at maximizing benefits to the business and IT through each of the seven stages of its evolution. Like the SOAMM levels, OSIMM divides the initial level into three levels, facilitating the initial evolution in some shorter maturity steps. In this model, different dimensions of IT are analyzed in each line and the columns correspond to maturity levels.

From the merge of SOAMM and SIMM, a new model called CSOAMM – Combined SOA Maturity Model was proposed. This model presents a more detailed level subdivided in smaller steps. The proposal presents ten levels. However, three stages are initial and are not related to SOA, which results in seven levels of SOA, such as SIMM [17]. Another important contribution was the definition of a maturity evaluation model based on the different perspectives of IT departments in the organization [18].

2.3 Service-Oriented Software Engineering Methods

Due to the variety of software engineering methods applied to service-orientation, a method selection guide was proposed according to the organization's needs [19]. This work had analyzed and evaluated the characteristics of each of the twelve methods: CBDI-SAE-TM (Component Based Development and Integration for Service Architecture & Engineering), SOA SDLC (Systems Development Life Cycle), SeCSE (Service Centric Systems Engineering), SOMA (Service-Oriented Modeling and Architecture), OASIS SOA-RM (Reference Model), SOAD (Service-Oriented Analysis and Design), SOSE (Service-Oriented Software Engineering), TRUE, CHANG, SODIUM (Service-Oriented Development In a Unified fraMework), SOAF (Service-oriented architecture framework) and SOUP (Service-Oriented Unified Process). This evaluation defined common criteria for evaluating different software engineering methods of SOA.

3. SOPLA – Service Oriented Process Line Approach

This section defines the characteristics and elements of the proposed process line called SOPLA – Service-Oriented Process Line Approach. The systematic delivery of service-oriented and BPM methods requires procedures, training and organizational governance, in the same way that they continuously require analysis and decisions for the project and development of this architecture of business processes, systems and infrastructure. The SOPLA process line aims at accelerating the definition of a process to support solutions based on this architecture.

3.1 Core Model Definitions

The SOPLA proposal is a software process line that reduces the work of the process engineer. A new process can be generated by selecting variability. The steps that the process engineer will have to take are: 1 - When beginning a new project, analyze its characteristic; 2 - Select the variability from the process line; 3 - Generate a process instance from the line and selected variability. This step applies the EPF tool [20] and GenArch-P [21]; 4 - Convert the process instance to BPMN (Business Process Modeling Notation) and configure the flow to execute the process. This conversion is made using a developed tool that makes the conversion from the EPF (UMA - Unified Method Architecture) to BPMN 2.0; 5 – Execute BPMN 2.0 with BPM Activiti platform, but due to the standardization many different BPM compatible platforms can be used.

To conceive the process line, the emphasis relied on execution, detailing the design, construction and product test activities. The process line documentation addressed higher granularity SPEM elements (phases and activities), detailing of lower granularity elements (tasks and steps) in the most relevant cases in the SOA and BPM context. The activities related to planning, change management, measurement, training, operating and monitoring were not part of the scope of this work. The process line design was based on the standard variability of SPEM 2.0 and some extra elements were added to improve the existing elements in the line. The standard variability was extends and substitutes, where the information omitted in the variability is inherited from the variation point.

The approach used to document the process line was based on vSPEM [22] using the concept of variation point, which allows instantiating a new process line which will be substituted for one of its variants. When the definition is omitted, each element represented in the process line is mandatory. Each variation point, also called an abstract component [23], has a classification type of its variability, such as:

- Alternative type, also called "XOR", where only one of the variants can be instanced;
- "OR" allowing the instance of more than one variant.

To document the elements that may or may not exist in a process line instance, the variability was specified as optional, represented by the symbol "?". The optional variability can be applied both to variation points and to elements with no variants. Figure 1 shows a summary of the variability types used in the line, similar to vSPEM.

	Mandatory minimum 1	Optional minimum 0	Variant
Process element with optional variability		?	-
Alternative variation point maximum 1		?	V
Or variation point maximum N		?	V

Figure 1: Variability elements and restrictions

3.2 SOPLA method content

The content of a method is the conceptual part used to define organization's knowledge base, where the independent elements of processes and development projects are persisted [24]. This part of the process is easily reused in any process, even out of the proposed line, because these elements are not temporal linked.

In the proposed line, some traditional parts of classic methods like Unified Process were used, enhanced by additional roles commonly present in medium or large organizations. The following list presents roles as responsible and participants within the process:

- Sponsor; Key user; Business analyst; Solution architect;
- System Analyst with 3 specifics profiles: SOA, User Experience and Information;
- Infrastructure Architect; Data Administrator;
- Security Information Architect that supports the Chief Security Officer (CSO) and standard compliance, such as: ISO 27001, Sarbanes-Oxley, etc.
- Programmers responsible for the code, with profiles specialized in SOA, User Interface, and Testing.

The definition of input and output of each activity is made by the work products. For the SOPLA line the following artifacts were proposed: Vision document; Business Requirements and Business process mapping (defining the as is and to be business scenarios for manual and automated: process, activities, tasks, KPIs with the used applications); Domain model (data types, services, class, database, glossary); Service list and adherence; Software architecture document; Use case specification and diagram; Supplementary specification; Navigation maps; User interfaces prototypes; Acceptance report; Test plan, Test case; Service mocks; and Test results.

3.3 SOPLA process content

When conceiving the process, the first step was to establish the life cycle, followed by the activities and variability specification. The basis for the process resulted from the analysis of 12 software engineering methods applied to service-orientation [19], being more influenced by the models that integrated strongly with BPM, RUP, OOAD and CBD. The life cycle was represented by phases that also have variability. For example, it may not be necessary to execute a certain phase, depending on the project's size classification. The proposed phases are presented in Figure 2 and each of the phases are presented in the next subsections.

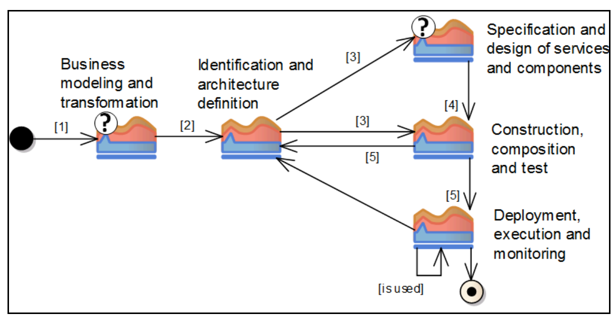

Figure 2: SOPLA phases

The life-cycle model used in the process line is iterative and incremental with great emphasis on the Identification phase, where the services and domain data should be searched and analyzed. This is one of the key steps toward reuse, aiming at finding artifacts to be reused, nevertheless reuse should also occur in other phases. The flow follow numbered arrows and the Identification phase is optional, so the process can bypass it and go directly to Specification or Construction phase. After the Construction phase, it is possible to execute the Deployment or again the Identification phase when a proof of concept is needed. These model phases were influence by the SOMA approach [14]. At the end of each interaction or phase, some of the project characteristics can be changed, generating the need for process adaptation, and when this occurs, steps 1 to 4 should be executed again to select the proper variability, generate a new process instance and update the process version in the BPM platform.

For each phase the activities have a detailed table with the activity name, descriptions and type. There is also an option to inform the variability and selection criteria, responsible; participants, inputs; outputs, and tools used.

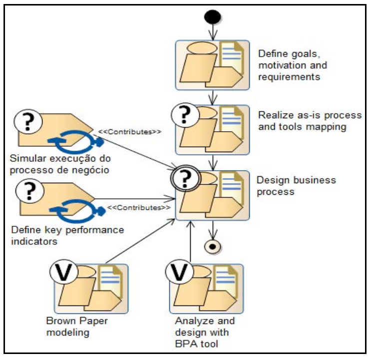

Figure 3: Business modeling and transition phases.

3.3.1 *Business modeling and transformation phase*

The Business modeling and transformation phase establishes the business objectives of the project, changing motivation and the main requirements. These business objectives should be clear, comprehensible to the sponsor (and management committees). For the business process transformation, the key elements are: the proper treatment of information, communication at the right time and the organization's conviction of the business process change. After simulating some future scenarios, the chosen one have to be chosen, as well as the performance indicators. Figure 3 shows the business modeling and transformation phase activities.

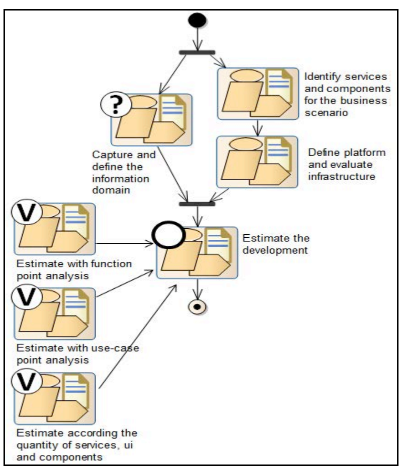

Figure 4: Identification and Architecture Definition phase

3.3.2 *Architecture identification and definition phase*

The goals of the phase for architecture identifying and defining the architecture are to strengthen reuse and select the right architecture standard, as well as estimating the costs and term of the proposed solution. The estimated costs should be evaluated and approved by the project sponsor or management committee (planning activity outside the scope of the line).

This phase happens after the business modeling, since the process cycle is iterative. At the end of an iteration the Identification phase can be performed again. In this case, it is possible to make a new specification and construction iteration without product release and at the end of this phase business services candidates are listed and classified according to the requirement adherence level. The main components of architecture and standard are selected. Figure 4 shows the activities for the identification and architecture definition phase.

3.3.3 *Specification and design of services and components phase*

In this phase the solution architecture is fully designed and described for each component. The main deliverables of the phase are: requirement documentation, service specifications, business rules, user interface design, technical restrictions and the specification of each component.

The adaptation of service specification is an activity that values maintaining compatibility with its previous versions, in such a way as to avoid regressive tests in several use scenarios. For this, there is a need to use a repository service that keeps this documentation updated and binds the customers to different use scenarios. A service changing with a high amount of customers generates needs for testing that may even wreck the Project financially. In some cases, an option is made keep more than one active version of the same service to reduce testing efforts. This phase is summarized in the diagram in Figure 5.

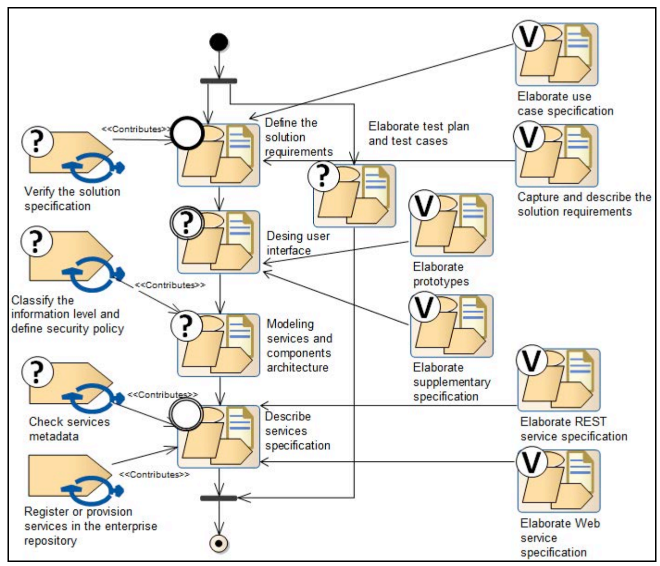

Figure 5: Service Specification and Project phase

3.3.4 *Construction, composition and testing phase*

The Construction, composition and testing phase is where the product is created. The previous phases identify and specify the components so that in this phase it is possible to manufacture the software. The main goals at this phase are: the construction of the products with agile tested components, assembly and integration, delivery of a quality product with low defect percentage and the user acceptance tests. At the end of this phase, as it is an iterative incremental model, the deployment phase can be executed to release part of the software under construction, as soon as possible. By making partial deliveries in each iteration, project risks are reduced and the return of investment is anticipated. For contract management, payments can be attached to delivery and acceptance of functional products [25]. Figure 6 shows the activity flow describing this phase.

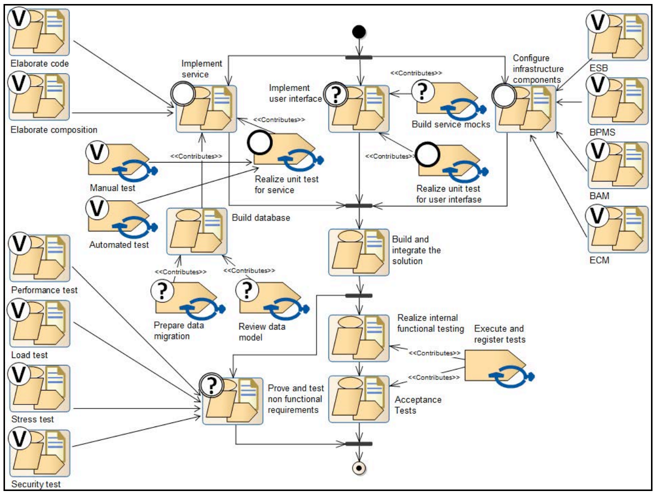

Figure 6: Construction, composition and testing phase

3.3.5 *Deployment, execution and monitoring phase*

After the Construction, composition and testing phase comes the deployment, execution and monitoring phase, responsible for releasing a new solution or version to the final user. A transport package or application copy to the production environment is made, configurations are updated by CMDB (Configuration Management Database) base and the user is enabled to use the software. The SLA indicators are established and monitored to keep the level of services as specified. The inclusion of new customers in a service causes an impact on the execution environment, which can compromise the achievement of SLA of other solutions. Therefore, constant monitoring is necessary, as well as business Key Performance Indicators (KPIs), such as technical services KPIs, as proposed by [14] and here adopted.

Within SOA and BPM, the infrastructure component Business Activity Monitoring (BAM) plays an important role in the automation of indicators, operating (feedback and errors) and business process. The control of the KPIs needs to be constantly and actively reviewed considering the compliance of the actual scenario, as well as possible improvements to meet new demands.

This phase was not fully described in the full work. The implementation of this phase should follow service models such as ITIL [26], CMMI-SVC [27], and ISO-20000 [28].

3.4 Automated support for the process line

The tools for documentation and management of engineering software processes, like the Rational Composer and EPF, have limits on process edition, on process element reuse, which is often done by copies, on variability selection that is not very effective, and there is also a lack of setting skills, execution and monitoring of processes [29]. Considering this problem, part of the tools processed by [29] is also applied, although the choice was made to develop a new layer for convention and execution of the software process.

To execute and monitor the process, a BPM platform was used, carrying to the construction of a new transformation tool between UMA and BPMN models. After the transformation to BPMN, a BPM Suite can be used for software process execution and monitoring. The activities and tools involved in the computational support for the process line are shown in Figure 7.

This approach begins with **Step 1**, which performs processes modeling and definition using the EPF tool to document the processes elements using its UMA metamodel. Every process line proposed in this work was documented in EPF, generating all elements and their variability, and all the information in EPF is defined based on SOPLA diagrams (Figure 3 to Figure 6). The UMA uses the same concept as its predecessor tools for project management, where the activities are related to their predecessors. This definition can be exported to a Gantt chart, to control project with tools such as the Microsoft Project.

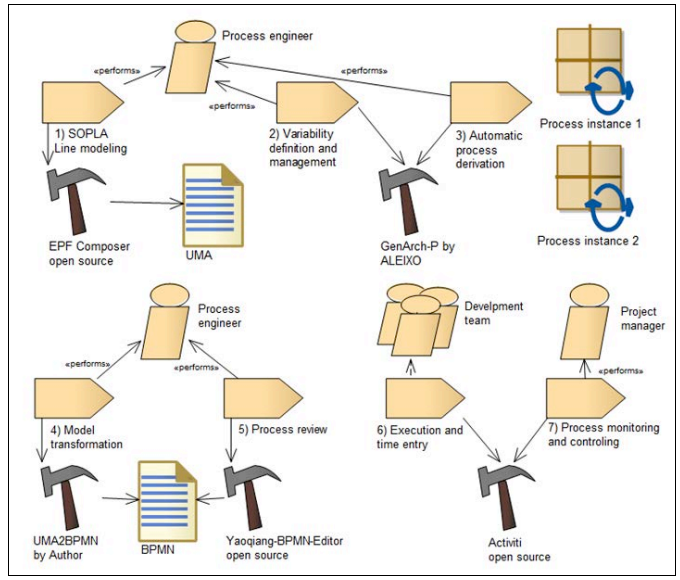

Figure 7: Activities and tools in computational support to process line, adapted from [29]

Following, **Step 2** uses the GenArch-P tool to process the definition and management of variability and **Step 3** analyses the characteristics needed for the main Project, followed by the selection of the alternative and optional variability to generate a new process instance [29].

Step 4 transforms a UMA model to BPMN, where a new plugin is developed for the eclipse that transforms a process instance derivate from UMA to BPMN version 2.0. This was the main contribution made to the set of tools used, where the Extensible Style Sheet Language Transformation (XLST) was used. This eclipse plugin extends the export options from EPF, and was called UMA2BPMN.

To generate a UMA file, the functionality of exporting the whole library from the method instance is used. After the creation of a UMA file from the process instance, it is made a transformation from UMA to BPMN models. The main difficulty of Step 4 is converting of information from the predecessors to generate all the existing transaction flows with definition of origin and destiny. The rules that make the transformation from UMA to BPMN are summarized in the following steps:

- UMA activities are transformed to userTask in BPMN;
- Using the predecessors between UMA activities the flows between the activities (sequenceFlow on BPMN) are created; The UMA activities without predecessors in BPMN are initiated directly for the process initializing event (startEvent);

- The UMA activities without predecessors in BPMN are initiated directly for the process initializing event (startEvent);
- The UMA activities without successor, in BPMN, have a flow to process the final event (endEvent);
- The activities that are related to more than one successor in UMA, including the initializing event, in the transforming to BPMN originate the inclusion of a parallel Gateway which is related to the other flows that occur separately;
- In UMA the activities with more than one predecessor, upon transformation to BPMN also generate a parallel Gateway to bind the activities, and this need may occur before the final event.

At first, it was decided to transform all the process tasks, but this rule was simplified to control only the activities. This was necessary due to the difficulty to take notes and monitor the flow of execution with all tasks, which often occurs jointly and simultaneously. In addition to the SOPLA, the classic methods such as RUP, define only the flow of activity execution, without defining the execution details and task sequencing. The result, after the transformation, is the generation of a BPMN file.

Step 5 of the approach is the review of the process, where an open source modeling tool for BPMN (GPL license) was used, called Yaoqiang BPMN. In this step visual adjustments can be visualized and made on the diagram generated for the process. These adjustments help the user to monitor and visualize the execution process flow, which can be seen in Figure 8.

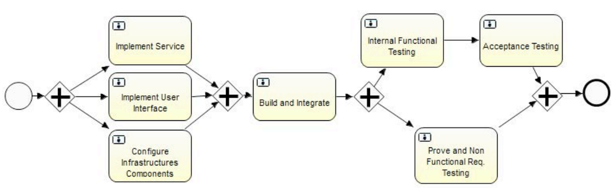

Figure 8: Visual adjustment in the BPMN editor

Step 6 loads the BPMN model to the Activiti Platform, which is a very lean BPM Platform that allows the definition of forms to perform the activities in a very easy and quick way. This tool was used to take notes of each of the process activities and build a standard form for information storage. A ".bar" file type is generally used to load the process into the tool. The generation of this file can be done using the Activiti plug-ins or using the Yaoqiang BPMN tool previously used.The last step is the control and monitoring of the process execution, where the project manager uses the database of the BPM Activity platform. This base has all the events selected at the process execution, helping monitoring and taking corrective actions to projects deviations. Commercial BPM platforms offer additional resources as analytic database and dashboards tools that facilitate the construction and graphic visualizing of information.

4. EVALUATION

The evaluation of the SOPLA process line was performed by 12 experts. The following criteria were used to choose the experts: more the 5000 hours in projects and at least 2 years of experience in software engineering processes, service oriented product engineering or business processes modeling. Among 30 professionals initially considered, just 12 have met the criteria and were eligible to participate in this research. The large majority of them (10) were professionals and just 2 have predominant experience in academia in software engineering processes. One was a project manager, two were researchers, four were system analyst or software engineering and five were enterprise or solution architects. The companies the participant worked for are six multinational, the largest Brazilian government IT company, the highest revenue Brazilian company, one IT state government company, two universities and one IT services enterprise.

The summary of the evaluation performed by 12 experts is presented in Table 1, considering questions about the phases and some questions about each phase. The summary table presents five groups, each of them with four questions, resulting in 240 answers (8 td, 17 pd, 48 ne, 47 pa and 120 ta). Considering the first item evaluated, overall evaluation of the phases, 4 experts selected the option for total or partial disagreement, producing 6 suggestions for improvement including temporal relations to the occurrence of the phase; and the optional variability of business modeling phase. However, the process line already has a tendency to be accepted considering the possible phases transitions with 7 agreements, 3 neutral positions and 2 disagreements. So, suggestions for improvement did not generate new SOPLA versions.

Two experts were unsatisfied with the flow transition and selected the totally disagree option. In this question it was suggested that the Construction, Composition and Testing phase should be split, separating the testing phase and generating a new phase. With this new phase it is possible to improve the interaction of geographically distributed testing teams. This suggestion appeared in two answers as well as in the "general vision of the phases", as in "Construction, composition and testing phase", where a suggestion was also made in the situations of "process modifying" and "transaction flow", totaling 6 negative results. However, many initiates for process improvements have stated an opposite view to this splitting, where the tests are integrated and driving the construction. The principle of doing it right at first demands a greater effort to comprehend and define tests in several development stages, which results in the preparation, construction and delivery of a highly effective and quality product. Therefore, this suggestion was not included in the line's review.

Table 1: Summary of the results of the experts analysis

Answers	td	pd	ne	pa	ta	Result
Overall evaluation of the phases	3	3	13	15	14	Tend to accept
Business modeling and transformation **phase**	0	4	12	7	25	Tend to accept
Identif. and architecture definition **phase**	0	4	9	10	25	Tend to accept
Specification and design of services and components **phase**	1	3	9	7	28	Tend to accept
Construction, composition and testing **phase**	4	3	5	8	28	Tend to accept
Total	8	17	48	47	120	

TD: Totally disagree; PD: Partly disagree; NE: Neutral; PA: Partly agree; TA: Totally agree

There was also a suggestion for including a review task for business process artifacts and KPI which generated another “partially disagree” response. In this task there is a validation step, but the suggestion was accepted. The definition of an explicit validation activity, as of the future model, KPI and comparatives should be reviewed and also validated by the sponsor or process management group. However, this activity permeates the planning activities that were avoided in the processes line conception.

Improvement 1 – Include of a new activity: Improvement validation in business process. The participants would be a business target process improvement council and the sponsor. In material preparation for this validation techniques like optional variability to support the proposal can be used, such as benchmarking and value engineering.The absence of an explicit activity for non-functional requirement definition and the building of an architectural prototype to evaluate technical viability also generated one “totally disagree” and one “partially disagree” response. However, the non-functional requirements are referred to in the line through a task in an activity called “identify services and components”. The suggestion for handling the model through an optional activity for architectural validation was accepted, even when this practice can be done with a short interaction of “identification and definition of Architecture”, “Services and Project Specification” and “Construction, Composition and Testing” phases.

Improvement 2 – Including a new activity: Proposed architecture validation. The participants would be the architecture board and the sponsor. Another point that generated a “totally disagree” response, was the “design user interface” activity, in which it was suggested that a contribute activity should be included to map and validate the user data entry. Another expert also suggested including in this activity a review task that generated a “partially disagree” answer. The “design user interface” activity had a task that refers to the validation with the user, the suggestions of to make explicit the information mapping task and review task was accepted and included in the line. The specification phase also received a “partially disagree” response due to the suggestion for joining the activities of “modeling services…” and “describe services”. In this matter the suggestion was not included in the review due to the “modeling services and components architecture” activity for the solution architecture that consumes and provides a number of services, while the goal of the “describe service specification” activity is to generate the specification that describes each of the reusable services for many applications. To the “describe service specification” activity there was also a “Partially Disagree” response, where improvements were suggested and acceptd for contemplating the governance of services inventory.

Improvement 3 – In "design user interface” activity the following task should be included: Mapping information and validation of data entries.

Improvement 4 – With the activity “describe service specification“, include name reviewing and governance of service inventory tasks. One expert selected 3 “partially disagree” answers due to the lack of low level granularities in the process, where the expectation was to define variability for the several tasks in the process line. The analysis of variability of all tasks was a restriction of this paperwork scope, which is why the suggestion did not generate a new review point in line. The “realize internal functional testing” and “acceptance test” activities also originated two “partially disagree” responses, with one in the transition and the other in the variability. The suggestion concerning transition did not generate new reviews, since the sequencing would reduce the average performance delivery, which is a common conductor for SOA and BPM processes to reduce the time to market. The suggestions, due to variability, require the detailing of each task and those responsible for them, which restricts the scope of work.

5. CONCLUSIONS

The adoption of software development for reuse is a theme that has been approached in academia, although little has been done to address questions related to service-oriented development, where there is a lack of well-defined processes. The SOPLA proposal explored the existing models to define of a process line that could improve the development process of products of this nature and integrate the paradigms, tools and practices involved.As previously stated, SOA has become essential to organizations and its technologies are used daily in several business processes. The development of service-oriented applications is demanded to leverage 3 out of 5 most important issues to the IT market, according to the priority research of CIOs conducted by [30], as following presented: 1-Delivery of business solutions for mobile devices; 3- Establishment of flexible infrastructures and with reduced cost using cloud computing (SAAS, PAAS and IAAS); and, 5-Building of collaborative solutions (workflow management- BPMS).

The manufacturing industry has applied the control of product variability for years, providing a favorable scenario to extend this approach for software. Although the application of software product variability is an approach that has been largely studied in academia, a great part of the software industry remains unaware of it. Therefore, the application of this approach to software process has not been widely explored, making the present work even more relevant. The definition of software process lines with a systematic management of their variability has little attention in both academia and industry. However, the software process line is a relevant approach that can help organizations to achieve quality and productivity improvements.

The main contribution of this research was the conception of a software process line for service-oriented product engineering. Few practical models can support software engineering definition processes according to the characteristics of organizations. Another contribution was the application and tailoring of the approach of [29] to define a process line, where the UMA2BPMN tool was developed and executed, allowing to get BPMN notation from UMA models. This transformation led to the control, execution and monitoring of processes through BPMN platforms, using the Activiti tool. The last contribution was the integration of business processes management models with the software development process. The composition of BPM models and solution development processes is an important contribution, aiming at improving the quality and synergy of the various roles and activities involved.

6. FUTURE WORK

The proposed process line could be adapted to include the activities related to the deployment, execution and monitoring phases and could cover several processes to automate the process execution control (during execution time) making change activities easier. Also, rather than working with only one tool for software process control using BPM, the computational support could be provided by specific tools for each process, improving the integration of activities between these tools, where the Open Services for Lifecycle Collaboration (OSLC) plays an important role. This would allow easy integration based on XML, ReST and Resource Description Framework (RDF).

7. ACKNOWLEDGMENTS

The authors would like to thank Pontifícia Universidade Católica do Paraná, Católica de Santa Catarina, WEG and all colleagues that have contributed with this work.

8. REFERENCES

[1] Prieto-Diaz, R. *Making Software Reuse Work: An Implementation Model.* ACM SIGSOFT Software Engineering Notes, 16,3 (1991) 61-68.

[2] Clements, P. Cl; Northrop, L. *Software Product Lines: Practices and Patterns.* Addison-Wesley, Reading, 2002.

[3] Bieberstein, N.; Laird, R.; Jones, K.; Mitra, *T. Executing SOA: A Practical Guide for the Service-Oriented Architect*. IBM Press, Indianapolis, 2008.

[4] Brown, W. A.; Laird, R. G.; Gee, C.; Mitra, T. *SOA Governance: Achieving and Sustaining Business and IT Agility.* IBM Press, Indianapolis, 2009.

[5] Van Der Aalst, W.M.P.; Ter Hofstede, A.H.M.; Weske, M. 2003. Business process management: a survey. In *International Conference on Business Process Management (BPM03).* (Berlin, 2003). Springer-Verlag, Heidelberg, 2003.

[6] Zhang, L.; Ge, M.; Bi, X.; Chen, S. A SOA-BPM-Based Architecture for Intelligent Power Dispatching System. In *Asia-Pacific Power and Energy Engineering Conference* (*APPEEC, 2010*), 2010.

[7] Sutton, S.M.; Osterweil, L.J. Product families and process families. In *Software Process Workshop. Process Support of Software Product Lines*. 1996, 109-111.

[8] Rombach, Dieter. Integrated Software Process and Product Lines. Unifying the Software Process Spectrum. In *International Software, Process Workshop (ISPW 2005)*. Springer, Beijing, 2005, 83-90.

[9] Spanoudakis, G.; Zisman, A.; Kozlenkov, A. A service discovery framework for service centric systems. In *Services Computing, Ieee International Conference,* Orlando, 2005.

[10] Zachos, K.; Maiden, N.; Zhu, X.; Jones, S. Discovering Web Services To Specify More Complete System Requirements. In *Advanced Information Systems Engineering*, v4495, Springer-Verlag, 142-147, 2007.

[11] Li, M.; Smidts, C. A ranking of software engineering measures based on expert opinion. In *IEEE Transactions on Software Engineering*, 29, 9 (sept. 2003), 811-824

[12] Reinehr, S. S. *Reuso Sistematizado de Software e Linhas de Produto de Software no Setor Financeiro: Estudos de Caso no Brasil*. Ph.D. Thesis, USP, São Paulo, Brazil, 2008.

[13] Gartner Inc., *Plateau of Productivity*, 2011. http://www.gartner.com/.

[14] Arsanjani, A.; Ghosh, S.; Allam, A.; Abdollah, T.; Ganapathy, S.; Holley, K. Soma: a method for developing service-oriented solutions. *IBM Systems Journal*, 47, 3, 2008

[15] Bianco, P.; Lewis, G. A.; Merson, P.; Simanta, S. *Architecting Service-Oriented Systems* (CMU/SEI-2011-TN-008). Software Engineering Institute, Pittsburgh, 2011.

[16] OSIMM – Open Group Service Integration Maturity Model. *Service Integration Maturity Model Technical Standard: The SOA Source Book*. <http://www. opengroup.org/soa/source-book/osimm/model.htm>.

[17] Söderström, E.; Meier, F. Combined SOA Maturity Model : Towards a Guide for SOA Adoption. In *Enterprise Interoperability* (II 2007). Springer, London, 2007, 389-399.

[18] Rathfelder C.; Groenda, H. iSOAMM: an independent SOA maturity model. In *International Conference On Distributed Applications And Interoperable Systems (IFIP WG 6.1)*, 8., 2008, Berlin, Germany. Springer-Verlag , Heidelberg, 2008.

[19] Gu, Q.; Lago, P. Guiding the selection of service-oriented software engineering methodologies. *Service Oriented Computing and Applications* (2011), 2011, 1-21.

[20] EPF - *Eclipse Process Framework Project*. http://epf.eclipse.org/.

[21] Aleixo, F.A.; Freire, M.A.; Alencar, D.; Campos, E.; Kulesza, U.: A Comparative Study of Compositional and Annotational Modelling Approaches for Software Process Lines. In *Simpósio Brasileiro de Engenharia de Software* (SBES 2012), 14, 2012, Natal, Brasil. 1-10.

[22] Martiinez-Ruiz, T.; Garcia, F.; Piattini, M.; MüNch, J. Modelling software process variability: an empirical study. In *Institution of Engineering and Technology*, IET Software, 5, 2, (Apr. 2011), 172-187.

[23] Barreto, A.; Nunes, E.; Rocha, A.R.; Murta, L. Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines. In *International Conference on the Quality of Information and Communic. Technology*, 7, Porto, Portugal, 2010, 15-24.

[24] OMG - Object Management Group. *Software & Systems Process Engineering Metamodel specification* (SPEM) Version 2.0. http://www.omg.org/spec/SPEM/2.0/.

[25] Cockburn, A. *Agile Software Development: The Coope-rative Game*.2ed.Addison-Wesley, Boston, MA:, 2006.

[26] OGC - Office of Government Commerce. *Introduction to the ITIL Service Lifecycle.* The Stationary Office, Belfast, United Kingdom, 2011.

[27] Software Engineering Institute. *CMMI® for Services* (CMU/SEI-2010-TR-034), Version 1.3. Carnegie Mellon University, Pittsburg, 2010.

[28] ISO/IEC 20000-1:2011. *Information technology -- Service management -- Part 1: Service management system requirements.* 2011.

[29] Aleixo, F.A., Freire, M.A., Santos, W.C., Kulesza, U.: Automating the Variability Management, Customization and Deployment of Software Processes: A Model-Driven Approach. In *International Conference Enterprise Information Systems* (ICEIS, 2011), 13, Beijing, 2011.

[30] Gartner Inc., 2012 CIO Agenda Survey, http://www.gartner.com/, 2012.