CPMEv: A Tool for Service-based Configurable Process Evolution

Hanae Sbai, Loubna El Faquih, Mounia Fredj and Laila Kjiri
AlQualsadi Research team, ENSIAS, Mohammed V University of Rabat, BP 713, Rabat 10000, Morocco

Abstract. Variability management in business processes domain is considered as a key of reuse. Research in this field focused mainly on variability modeling and resolution; neglecting the evolution aspect. The study of existing tools in this domain shows the lack of solutions integrating both the evolution management, and the change propagation with respect to the variability. The need for evolution emerges from the occurrences of new business requirements affecting the business processes. The evolution at business layer represented by configurable processes impacts the IT layer represented by services, and hence the alignment of configurable processes and configurable services must occur to maintain an integrated view of an organization. This can be reached by the concept of service-based configurable processes. This paper presents the CPMEv, a novel tool for evolution management of service-based configurable processes.

Keywords: Variability, Evolution, configurable services, Alignment, Change propagation.

1. Introduction

In the last few years, with the wide adoption of PAIS (Process Aware Information Systems) such as ERP (Enterprise Resource Planning), companies are left with directories containing several variants of the same business processes. These variants differ according to their application context. For instance, in the e-healthcare domain, 90 variants of “medical examination process” could be distinguished in a hospital [1]. Consequently, in order to choose or combine variants, the designer has to compare and adapt them manually, which could be a complex and an error prone operation. In this context, many research studies have focused on managing the variability of business processes by developing configurable processes [2] [3] [4] [5] [6] [7]. The variability management is about three main phases: modeling, resolution and evolution. In terms of modeling, many approaches for business process variability representation have been proposed [8] [2] [3]. Regarding the resolution, there are several solutions enabling the generation of variants [9] [3]. Whereas the evolution still presents many limitations, the main contribution in this domain is the work of [10] which is an extension of the change patterns of [11] to support changes of process families. However, this proposal is limited to the processes represented in structural blocks and does not assist the designer during the evolution. In this context, we have proposed a pattern based approach offering model and process solutions [12].

Along with the improvement of business process reuse by the introduction of the variability management, the proposed approaches lack of business-IT alignment support. Our study shows that these approaches do not allow the generation of configurable services emanating from configurable
processes, in this context a new concept related to business processes has been introduced, which is the “service based process model” [13]. This has led us to study the alignment between the configurable processes and the enterprise applications, in particular services, with the aim of building PAIS that support service orientation. The emergence of variability management in business processes and services conduct the PAIS today to adopt the configurable processes at the business layer and the configurable services at the IT layer. In this perspective, we proposed an MDA (Model Driven Architecture) approach for the configurable services generation [14].

In this paper, we focus on the implementation of the proposed approach in configurable processes’ evolution and their alignment with the IT layer via a web tool named CPMEv. In fact, the business process variability management tools focus on the resolution of EPC (Event Process Chain) and YAWL (Yet Another Workflow Language) models [15] and do not provide an evolution management nor an automatic generation of configurable services from these processes. Furthermore, existing tools on business-IT alignment, and in particular on service generation, focus on mapping BPMN (Business Process Model and Notation) process models to SOAML (Service Oriented Architecture Modeling Language) Service models [16], without any support of variability concept. We argue that it’s necessary to dispose of a tool which combines processes evolution and their alignment with services regarding to variability management.

The paper is structured as follows: Section 2 reviews related work about the business process change management tools. Section 3 elaborates the requirements of our tool. In Section 4, the tool is described using an applicability study. Finally, Section 5 presents some discussion about what has been achieved, and Section 6 concludes the paper with some pointers for further work.

2. State of the art of business process management tools

Current business process management tools can be put under three main categories: business process change, business process variability management and service generation tools. In this section, we analyze existing solutions in each category, evaluating how suitable for our purposes they are. This analysis also provides valuable input regarding the requirements of our proposal.

2.1 Business process change tools

The evolution is the ability of an implemented process (a process instance) in a PAIS to change when its correspondent business process (also called process type) changes. The most common classification of the business process evolution is based on the process life cycle [17]. Two changes types can be identified:

- Process Type Level: concerns the change during the business process design.
- Process Instance Level: concerns the change during the business process execution.

In this context, too main frameworks have been proposed: ADEPT2 and AristaFlow. ADEPT2 [18] offers powerful concepts for modeling, analyzing, and verifying process schemes. It ensures
schema correctness, like the absence of deadlock-causing cycles or erroneous data flows. ADEPT2 supports both ad-hoc changes of single process instances and the propagation of process type changes to running instances. In order to enable robust and flexible PAIS, the ADEPT2 tool has been transferred to an industrial tool called ARISTAFLOW BPM suite [19]. This tool supports “correctness by construction” during process composition and guarantees correctness in the context of dynamic process changes.

As shown above, business process evolution tools are interested in managing business process evolution. However, they are not sufficient when it comes to evolve configurable process models, since they do not support the variability concept introduced by configurable process models.

2.2 Business process variability management tools

The variability represents the ability to change or customize a software system. It can be initially identified based on the concept of feature, which is a behavioral logical unit that corresponds to a set of functional requirements [20]. Variability lies on two main concepts: variation point (variability object part) and variant (variation point realization). A variation point may be optional, alternative, or a set of alternatives [21].

For managing the variability of eEPC models including their resolution, a module has been developed and integrated with the ARIS platform [22]. In the same vein as [8], authors developed an Eclipse plugin that only eliminates the non-required variants of Variant-Rich BPMN models. To provide a support for modeling and resolving C-EPC and C-IEPC models, "SYNERGIA" tool has been developed by La Rosa et al. [15]. Thus, a module C-YAWL Synergia has been integrated to support the configuration of the configurable workflow models [9]. Regarding the evolution of configurable process models, an implementation of ARIS under PROVOP framework is developed in [3]. However, this tool does not provide neither information on how to change the variation points, nor a process (method) for guiding this change. In order to manage a broad set of process variants, authors in [23] provide the APROMORE tool which brings together a rich set of features for the analysis, management and use of sets of configurable process models. Moreover, an integrated plugin for BPMN2 models was developed by [1] to enable the resolution of the variability of process models based on the BVR (Base Variation Resolution) approach which represents a configurable process model into three layers: base model, variation model and resolution model. Recently, the “modelery” tool has been developed for a collaborative web repository [24]. It allows multiple researchers to archive, search and catalogue configurable processes.

Despite the fact that these tools are well suited for modeling, configuring and storing processes, they cannot face the challenges raised by configurable process evolution. Indeed, we found weak support for the stage of configurable process evolution, especially at multiple perspectives.
2.3 Service generation tools

Based on our finding, the most existing tools for service generation adopt the MDA approach (Model Driven Architecture) which proposes a model driven development based on three main models such as CIM (Computation Independent Model), PIM (Platform Independent Model) and PSM (Platform Specific Model) [25].

With regard to the service based process model domain, authors in [26] developed a framework called MINERVA (Model drIveN&sErviceoRiented Framework for the continuous business process improVement&relAted tools). MINERVA is a combination of Eclipse plugins (BPMN modeling, Medini QVT, Magic Draw and Model Pro). It allows the generation of SOAML service models (CIMtoPIM transformation) from BPMN models. The SOAML models are then transformed into execution models represented in WSBPEL or XPDL (PIMtoPSM transformation).

As part of SHAPE project [27], the Modelio tool is developed in order to model and generate the web services including WSDL and XSD contracts. Thus, ATL (Atlas Transforming Language) modules for transforming BPMN models to SOAML models have been proposed in [28] [29] and also for the generation of variable web services [30].

These tools fall behind in supporting the variability of process models associated to services. None of the above cited tools covers the CIM2PIM transformation level including the variability support. Indeed, it is necessary to have a configurable services generation tool from configurable processes which covers both CIM2PIM and PIM2PSM between the MDA levels.

We define the following criteria for comparing the existing solutions (cf. Table 1):

- **Functionalities**: lists all the supported functionalities supported by the tool.
- **Formalism**: mentions the supported business process formalism.
- **Multi-perspective**: determines if the tool provides multi perspective variability management (activity, data, resource…).
- **Variability**: checks whether the tool supports the business processes variability.
- **Web**: indicates the use of a web environment.

**Table 1: Business process management tools comparison**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Functionalities</th>
<th>Formalism</th>
<th>Multi-perspective</th>
<th>Variability</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>[18]</td>
<td>Dynamic change</td>
<td>EPC</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[19]</td>
<td>Dynamic change</td>
<td>EPC</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[22]</td>
<td>Variability resolution</td>
<td>EPC</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>[8]</td>
<td>Variability resolution</td>
<td>BPMN</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Discussion. None of the analyzed tools was found suitable to fulfill the needs of “a configurable process evolution and service generation tool”, which can fully support the process evolution, the services generation, the storage and the change propagation. To summarize, it is possible to say that the tools are generally for EPC models [15] [18] [19] [22] [23], are closed process configuration [1] [15] [8] [22] or too limited in a single process/service without any support of variability concept [18] [19] [26] [27] [28] [29].

Our goal is to have a set of modules for configurable processes evolution, plus their alignment with the service layer through the configurable services generation module. Moreover, the tool should allow saving the configurable processes and allowing the versioning. The integration of these functionalities in a web tool will provide a collaborative environment to evolve the PAIS and will also provide a model driven development framework, where the configurable process models conduct the development of configurable services.
3 Requirements of a configurable process evolution and service generation tool

The main functionalities that we look forward to in our tool are: configurable process evolution, configurable service generation and change propagation. In what follows, we establish a set of requirements for each of the three main functionalities.

3.1 Configurable process evolution

In order to overcome the limitations presented in section 2.1, we propose a new change management tool that can generate not only the solution model, but also endows the PAIS designers of a complete framework for configurable process change, while ensuring the correctness of these changes. Thus, the design of our tool requires the identification of the different changes that could be applied to processes, and also the necessary patterns to realize these changes.

Understanding changes in configurable processes. The evolution types are distinguished according to the configurable process’s perspectives (functional, organizational and informational) and the elements on which these changes are applied. In fact, we define the perspectives supported by a configurable process as follows [12]:

- Functional variability: represents the variability of the activities (atomic/composite).
- Behavioral variability: represents the variability of sequence flows and control flows.
- Organizational variability: concerns the variability of resources which can be organizational unit (eg. customer service), a human role (eg. patient) or system (eg. payment system).
- Informational Variability: represents the variability of data and events.

Changes that may be made on a configurable process can affect configurable or non-configurable elements. Change on non-configurable elements concerns resources, data, events and activities. The last one has been addressed in Weber [11]. As for configurable elements’ changes, it can affect a variation point (object of variation), variant (realization of the variation point) or variable element (set of variation points).

Applying changes with patterns. To apply the different changes identified above, we proposed a set of patterns that are represented by P-Sigma formalism [31] which simplifies the expression of both model and process solutions. The “model solution” represents the solution to a given problem and the “process solution” is a guide that assists the designer in the evolution of the model. The P-Sigma formalism is composed of three parts, namely: Interface (allows the pattern selection), Realization (defines the solutions provided by the pattern) and Relation (organizes the relations between patterns).

We present in Table 2 an example of functional variability evolution pattern that concerns an insertion of a variation point activity [12].
Table 2: Variation point activity insertion pattern

<table>
<thead>
<tr>
<th>Parts</th>
<th>Fields</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identification</td>
<td>Variation Point Activity Insertion (VPAI)</td>
</tr>
<tr>
<td></td>
<td>Classification</td>
<td>Configurable process, Variation Point Activity</td>
</tr>
<tr>
<td></td>
<td>Issue</td>
<td>New business requirement: the need to insert a new activity of type variation point in a process</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>In an online sales system, we intend to offer several items delivery types which are not supported by the existing configurable process. In order to perform this, a new activity variation point &quot;deliver item&quot; can be inserted with a default variant such as &quot;express delivery&quot;.</td>
</tr>
</tbody>
</table>

| Solution Approach | Design choice: (a) insert a new activity variation point (b) transform an existing activity into a variation point |
|                  | If (a) Then |
|                  | Enter the activity name |
|                  | Choose the variation point type |
|                  | For each variant activity to insert Do |
|                  | Apply the pattern of variant activity insertion |
|                  | End for |
|                  | Enter the required capabilities R_CVPA for the variation point activity |
|                  | Define the insertion position of the variation point activity in the sequence flows //position between two activities, between an event and an activity or between two events |
|                  | Apply the sequence flows’ pattern deletion (Weber et al., 2008) |
|                  | Define the resource to affect |
|                  | If R_CVPA is included in Capability Then |
|                  | Affect the resource |
|                  | Apply the sequence flows’ insertion pattern (Weber et al., 2008) |
|                  | Else |
|                  | Apply the pattern of variation point resource insertion |
|                  | End If |
|                  | Else if (b) // Transform an existing activity into a variation point |
|                  | End If |
Example of the solution approach

For example, we suppose that the designer chooses to insert a new variation point activity B.

- The designer enters the variation point activity’s name: B
- The designer chooses the type “Alternative” for B. Thus, the activity is annotated by the stereotype “VarPoint”. (2)
- The designer has to insert two variants for B: the pattern of variant activity insertion is applied twice in order to insert B1 and B2. (3)
- The designer enters the required capabilities R.CVPA of the activity B: {RC1, RC2, RC3} (4)
- The designer defines the insertion position in the sequence flow. He/she has to insert the activity B between the activities A and D: Apply the sequence flows’ deletion pattern in order to delete the sequence flows {A → D} (5)

3.2 Configurable service generation

We have proposed a module for the generation of VarSOAML [30] configurable service models from a configurable process model by applying MDA approach. We note that the concept of configurable service represents a set of services that share common aspects, and also have predefined variables aspects. There exists several approaches for configurable services modeling; in this work we opt for the VARSOAML language since it is the richest language in terms of representation. Our MDA-based proposal is related to the CIM2PIM level, and provides rules for configurable services generation [14].
Example of generation rule. We present the rule for the transformation of the VariableCompositeActivity element to ProviderInterface or ConsumerInterface elements, as well as VariableCompositeActivity element is shown in VarSOAML by two interfaces: Provider Interface and Consumer Interface.

**Rule name:** VariableCompositeActivity2ConsumerInterface&ProviderInterface

**Input element:** VariableCompositeActivity

**Output element:** ConsumerInterface or ProviderInterface

For each VariableCompositeActivity element Do

- Create an element of ConsumerInterfaceOrProviderInterface types
- The name of the ConsumerInterfaceOrProviderInterface element is the name of the VariableCompositeActivity element
- If (the IncomingMattribute is Null) of the first SimpleActivity or VariationPointActivity or Event elements contained in the VariableCompositeActivity Then
  - Create ConsumerInterface element
- Else Create ProviderInterface element
- Apply VariableCompositeActivity2VariableInterface/Create aVariableInterface which represents the service interface which provides the ProviderInterface

End If

For each element SimpleActivity element in VariableCompositeActivity Do

- Apply SimpleActivity2Operation

End For

For each VariationPointActivity element Do

- Apply VariationPointActivity2VariationOperation

End For

3.3 Change propagation

Alongside the services’ generation, if the configurable process evolves, configurable services generated from it will necessarily be impacted by this change. To optimize this functionality, the change propagation is based on impact patterns [32] that provide a simplified way to identify the impacted parts of the configurable services. The definition of the change impact patterns involves the definition of all kinds of impact. Hence, in order to consider the changes related to all configurable and non-configurable elements, we define two types of change impact: change impact of non-configurable elements of a configurable process and impact of the configurable elements. Examples of the impact of configurable elements are published in [32].

4. CPMEv: a configurable process management evolution tool

The CPMEv (Configurable Process Management Evolution tool) is a tool for configurable process evolution and configurable service generation. It is developed according to the MDA approach. We
start by describing the technical architecture of our tool, followed by a use case for configurable process evolution, service generation and change propagation.

4.1 CPMEv architecture

The CPMEv tool is developed using the JEE technology in the NetBeans environment. We have adopted the MVC [33] pattern (Model View Controller) that allows organizing JEE applications in three layers, namely business layers (also called Model), Controller and presentation (also called View). For the implementation of the MVC pattern, we used the JSF framework 2.0 (JavaServer Faces) which is an Oracle standard [34]. The architecture is depicted in Figure 1. The CPMEv tool is intended to evolve configurable processes that are already stored as XML files.

![Figure 1. CPMEv architecture.](image)

The View layer brings together the different graphical interfaces through which the designer applies the desired changes to evolve the configurable process. The view layer is based on Primefaces technology [35] to build the graphical interfaces.

The Controller layer contains the controller classes CPMController.java, DBController.java and ServController.java. CPMController.java is used to launch the evolution algorithms described in the patterns, while the DBController ensures the connection between the CPMEv tool and the XML database eXistdb [36]. Finally, the ServController.java is responsible for the configurable services generation rules (cf. Section 4.3). The controller layer uses the Java API Jdom for XML files manipulation (browsing, modification, items removal and adding).

The Model layer contains the representation of configurable process elements, described in Variant-Rich BPMN as java classes. Moreover, it includes the storage module represented by the XML database eXistdb used to store XML files corresponding to configurable processes.
4.2 Configurable process evolution use case

To illustrate the evolution scenarios, we use a simplified version of the E-healthcare configurable process presented in [8]. The process (cf. Figure 2) is initiated when the patient sends a request for examination. Once received, the hospital system (composed of « general doctor » and « cardiologist doctor ») handles the request. After performing a clinical examination, two medical tests could be considered: "MRT Test" and "X-Ray Test", and a medical report is then generated. Based on this report, the treatment type ("Non Operative Treatment" or "Operative Treatment") is decided. At last, the patient makes the payment depending on the type of treatment.

![Diagram of E-healthcare configurable process](image)

**Figure 2.** E-healthcare configurable process [8]

An extract from the above described process is represented as XML file. The used XML tags and attributes correspond to notation of VR-BPMN:

```xml
<VariableCollaboration id="1c" name="Examination">
<MessageFlow id="1m" sourceRef="12" targetRef="26"/>
...
<Participant id="R1" name="Patient">
<CompositeActivity id="1ca" name="Examination">
...
(Activity id="12" name="Requestforexamination" nature="Simple" ParticipantRef="R1">
<DataObjectInput id="patientof12" name="patient">
```

27
In the following, we show through several interfaces, the interaction flows of the user with the tool in order to perform the variation point activity insertion.

The application menu is composed of four main options: Registration, CPM search, CPM evolution and service generation.

**Registration.** It is the first interface of the tool. It allows the CPM designer to register in the tool by typing his/her first name, last name, organization, login and password.

**CPM search.** After registration, the designer searches for the configurable process candidate to evolve. The search is lexical and is performed by entering the full name of the process or a part of it. The list of XML files corresponding to processes is then displayed as showed in Figure 3.

![Figure 3. Configurable process search.](image)

The research results is a list of files displayed as depicted in figure 3.

**CPM Evolution.** Once the configurable process is loaded, the designer is invited to choose the change type (insertion, substitution or deletion). Afterwards, he selects the configurable element: object of change (activity, resource, data or event), and its type (variation point, variant or variable). (cf. Figure 4). For instance, if the designer wants to insert a new variation point activity, he has to affect the resource which will realize it. Thereby, there are two cases: insert a new resource (cf. Figure 5) or affect the activity to an existing resource. The first scenario is described below:
Figure. 4. Configurable process evolution.

Variation point activity insertion with new resource. In this case, a screen asks the designer to insert the fields ID, variation point activity name, and to select the type (alternative, optional or optional alternative). Then the designer inserts the sequence flows of the activity, the variant activities and the variation point resources. We recall that the sequence flows insertion [17] is out of the scope of this paper.

Figure. 5. Variation point activity insertion with new resource.

New variation point resource insertion. This step consists of entering the ID, the variation point resource name, selecting the variation point type (Alternative, Optional or Alternative Optional) and the resource type (Role, System or Organizational Unit). The designer should also insert the associated variant resources (cf. Figure 6).
Through the illustration of this scenario, we have shown the possible cases of inserting a variation point activity, one of the evolution types supported by the tool. After the validation, a new XML file is generated and stored in the database under another name which is a combination of the origin file name and the current date. The new version of the CPM contains a new variation point activity «Payment processing». This variation point has two variants: “operative” and “non operative payment”. Each variant has its input and output data.

### 4.3 Service generation use case

This service generation functionality is integrated in the CPMEv tool and called: CS Generator. This tool is created to enable the automatic generation of configurable services from a configurable process (original or evolved version). CSGenerator is a Java module that can be launched when the user desires generating the services, by choosing the menu "Service generation" in the interface CPMSearch (cf. Figure 7). The controller responsible of this module is the ServController object.

We describe in Figure 7, the service generation interface.

The designer could select to generate one or more configurable service models. In this section, we present an example of configurable services generation related to the E-healthcare configurable process. From the original E-Healthcare.xml file we obtain the variable service contract model, the message type), the participant model and the service interface which we present in the following:
From the service interface model, we generated the Examination web service Examination represented by the WSDL file and the variability specification.

4.4 Change propagation use case

If the CPM designer selects an evolved version of the E-healthcare CPM, we obtain, in addition to the contract model «Examination», a new service «Payment» having its own service contract «Payment». This contract is related to the evolved version of the E-healthcare.xml file:

```xml
<VariableInterface name="Payment">
  <InterfaceProvider name="paymentTreatment">
    <operation name="Perform payment" />
    <VariationOperation name="paymentProcessing" …>
      <VariantOperation name="noneOperativePayment" RequiredVariant="" mutexVariant="" />
    </VariationOperation>
  </InterfaceProvider>
</VariableInterface>
```
Thereby, the tool allows propagating changes from a configurable to process the associated configurable services, in addition to the generation of configurable services.

5 Discussion

The wide adoption of PAIS, BPM and service-oriented architectures justifies the interest of researchers in these fields. Our goal, at first, was to enhance the reuse of business processes at the business layer of PAIS. Then to provide a solution for the alignment of the IT layer/services and the business layer/business processes.

The advantage of this tool is that it implements all directives of the proposed patterns. In addition to the evolution management, the tool provides a module for aligning configurable services from configurable process. It also offers an automatic generation, from a configurable process, of service contracts as well as service interfaces, message type and participants of configurable services. From these models, web services are generated and published in a configurable services register. By comparing our solution to existing tools, we point some similarities between our tool and Synergia, as our objectives are somehow similar. However, we provide some improvements with the CPMEv. First, our tool provides the evolution functionality which is not fully supported by the presented tools.

A registered user is invited to search a given CPM, choose the type of evolution and then apply changes to the CPM. Synergia is considerably more restricted in process variability resolving. The only formalism supported by Synergia tool is EPC or YAWL languages. Another possibility offered by our tool is an interesting online configurable services generation. Our tool offers the CSGenerator which is an MDA based module. Existing tools on service generation [26] [28] focus only on service models without supporting variability and do not provide other functionalities than service generation. CPMEv provides more usable options as the change propagation from configurable processes to associated services.

6. Conclusion and future work

Many solutions for business process management have been proposed. However, we can underline some limitations such as the weak support of business process variability and evolution. The solution we are proposing (CPMEv) is an interactive tool, with simplified user interfaces aiming at assisting the designers in configurable process development. It allows keeping the traceability of various modified versions of a configurable process via the storage module. The use of XML to represent configurable processes facilitates their manipulation. The proposed tool goes beyond the generation of services, and allows the synchronization of changes emanating from a configurable process and impacting the associated configurable services. One feature to mention is that our tool supports the change of variable as well as non-variable elements.

The CPMEv tool is designed to support the evolution of configurable processes represented in VR BPMN. As configurable processes in a PAIS might be represented using another notation, we need to work on an extension of CPMEv to carry out other formalisms. Another possible improvement is the
support of graphical modeling of processes. It is in this perspective that we are currently working on the development of a graphical editor for BPMN configurable process modeling (by use of GMF framework -Graphical Modeling Framework- under Eclipse), also incorporating the semantic aspect to design intelligent configurable processes [37].

References


2. La Rosa M. Managing variability in process-aware information systems (Doctoral dissertation, Queensland University of Technology Brisbane, Australia 25), 2009.


