Variability Management in Business-IT Alignment: MDA based Approach

Hanae Sbai[1] and Mounia Fredj[2]
[1,2] AlQualsadi Research team, ENSIAS, Mohammed V University of Rabat,
BP 713, Rabat 10000, Morocco

***Abstract*—The expansion of PAIS (Process Aware Information Systems) has created the need for reuse in business processes. In fact, companies are left with directories containing several variants of the same business processes, which differ according to their application context. Consequently, the development of PAIS has become increasingly expensive. Therefore, research in business process management domain introduced the concept of configurable process, with the aim of managing the variability of business process. However, with the emergence of the services-based development paradigm, the alignment of services with business processes is highly required in PAIS. Thus, in this paper an MDA based method which allows for generating configurable services from configurable process is proposed.**

Keywords—alignment; variability; MDA; PAIS; configurable service, configurable process

I. INTRODUCTION

Due to the lack of process control and automation into information systems centered data, the process orientation was established by the introduction of a new generation of information system called Process Aware Information System (PAIS), where the main unit of these information systems is the business process models. Thus, workflow management systems (WFMS) and integrated systems known as Enterprise Resource Planning (ERP) represent an example of a PAIS [1]. In the literature, a PAIS is considered as a software system that manages and executes business processes involving people, applications and information sources, based on a process model, while advocating separation of business logic and application logic [1].

In the last few years, with the wide adoption of PAIS, companies are left with directories containing several variants of the same business processes, which 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 [2]. 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 [3] [4] [5] [6] [7] [8] [9].

Along with the improvement of business process reuse by the introduction of the variability management, the proposed approaches lack of business-IT alignment support. The study of existing works 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" [14]. 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, an MDA (Model Driven Architecture) approach for the configurable services generation is developed in [15].

Therefore, it is argued that the alignment with supporting variability could also be beneficial. This alignment will enable the traceability management of business needs expressed at the business layer and their realization at the IT layer [16]. Indeed, it also allows change synchronization between the two layers. Consequently, the alignment is not limited only to establish the mapping between the configurable processes and configurable services, but also to maintain this correspondence when companies business needs evolve.

The paper is structured as follow: the concept of alignment supporting variability is firstly introduced and secondly the comparative study of different approaches is given. An MDA based method for configurable service generation is described in the Section IV.

II. BUSINESS-IT ALIGNMENT SUPPORTING VARIABILITY

A. Alignment concept

This section focuses on defining and discussing the most existing definitions of the alignment concept.

Alignment can be defined as the «dependency management» between business processes and services [18], «connection» of services to processes [19] or «change synchronization» between business processes and services [16].

According to the authors [20], the alignment of services with business processes is the ability to realize business process as a set of services. In this sense, the alignment allows for ensuring coherence between the processes of the business layer and services of the IT layer. It is considered that the alignment consists of the design of service-oriented architectures in a way that allows for easily adaptable business processes. This requires not only defining the dependency relationships between the activities of a business process and

related services, but also managing changes of processes and their related services.

In this work, business IT alignment concerns the generation of configurable services from configurable process in order to maintain consistency between the two layers (IT and business) and facilitate change management, taking into account the variability of processes and services.

Existing works on business IT alignment can be put under two main categories:

- Generation of services from a business process that involves designing business processes and defining the different rules that generate services from configurable process. These services will be implemented using the web services technology.
- Change management by analyzing the impact of changing the business process and services.

In this article, the main work is about the first category by focusing on the management of variability when it comes to generate services from a business process.

B. Service generation from business process

In this section, the concept of service generation is detailed in order to define the main elements which are required when it comes to generate services.

In the literature, the generation consists of transforming a business process into a set of services. This describes how services can be automatically generated from a business process.

The study of service generation works [14] [16] [17] [21] has shown that these works use the BPMN language [22] to represent a business process and the SoaML (Service Oriented Architecture Modeling language) [23] to represent the services to generate. For the generation, it is carried out in the most of time under the MDA (Model Driven Architecture) [24].

Furthermore, despite of the diversity of these works, there is no generation approach supporting the variability of the business process and services. In this perspective, an MDA based approach for the generation of VARSOAml configurable services from a Variant-Rich BPMN configurable process is proposed.

Thus, service generation with supporting variability requires:

a) Language for modeling a configurable process

b) Language for representing configurable services

c) MDA approach for configurable services generation

2) Modeling of configurable process: Variant-Rich BPMN language

There are many approaches to represent the variability of BPMN process [7] [9] [10]. All these approaches are derived from the Variant Rich BPMN (VR-BPMN) language [5]. The VR-BPMN extends BPMN to support the variability of business processes using annotation technique. It allows representing three concepts of variability, a variation point (alternative or optional), a variant (default or simple variant) and the relationship between variation point and variants (encapsulation, extension, inheritance). It was used in several case studies within the automotive field [6] and E-healthcare [11]. All the variability representation stereotypes are described in the following table (cf. Table I):

TABLE I. VARIABILITY REPRESENTATION STEREOTYPES OF THE VR-BPMN

Configurable elements		***Variability representation Stereotypes***
Variation point activity	**Alternative**	-« VarPoint» defines an alternative variation point activity -« Abstract» defines an abstract variation point activity with several implementations.
	Optional	-«Optional »- defines an optional variation point activity that can be extended by several variants.
Variant		-«Default » represents the default realization of a variation point activity. -«Variant » represents the realization of a variation point activity
Association {variation point variant}		-« implementation » is used to associate a variant activity with an abstract variation point activity. -« inheritance » is used to signify that a variant activity is a type of a variation point activity -« extension » is used to associate a variant activity with an optional variation point activity.

For each variant activity, a feature is associated to define the selection condition of the alternative activity. In this work, the Variant-Rich BPMN is used to represent the variability of all perspectives of business processes (functional, behavioral, organizational and informational) [13], unlike the generation approaches that are limited to activity and data transformations. A complete representation of configurable process will allow us to treat the generation of services in a broader sense.

3) Representing configurable services: VarSOAML language

Recently, with the emergence of reuse in SOA, some approaches, yet few, became interested in modeling services supporting variability. The approaches that propose an extension of SoaML language to support the variability of services are thus examined. The VarSOAML language [25] is adopted. It represents the variability of all SoaML service elements, to cover four views of service, namely the business view, structural, functional and composition. To extend the elements of SoaML, VarSOAML uses UML stereotypes. Different representation stereotypes of service elements (contract participant, message, service and operation interface) are described below (cf. Table II).

TABLE II. REPRESENTATION STEREOTYPES OF VARSOAML

Representation stereotypes of variable service elements	Definition
« VariableContract «	Describes the variability of collaboration.
« VariableInterface »	Define a variable service interface
« VariationOperation »	Defines a variation required or consumer operation
« VariableMessage »	Defines a variable message type of service.
« VariantOperation »	Defines a variant operation.
« VariationType »	Defines a variation data.
« VariantType »	Define a variant data.
« VariableParticipant»	Represents a variable participant of service

It appears that the VARSOAML language is the richest language in terms of representation, because it covers four views of service, namely the Service View (contract variable), functional view (variable interface, variation / variant operation, variation / variant type), structural view (variable participant) and the composition view (UML activity diagram).

4) Generation approach: MDA

MDA can be defined as the achievement of the MDD approach (Model Driven Developement) around a set of OMG standards such as MOF (Model Object Facility), UML, XMI and OCL enabling a new model based development approach.MDA defines three types of models [24]:

- CIM (Computation Independent Model): It represents the business requirements of a system. This is a model of business requirements defining the business interactions and business tasks of a system, without describing its structure or its implementation. In object-oriented approaches, CIM is represented by the use case diagram, while in service-oriented approaches CIM is represented by BPMN business process models [14] [17] or UML activity diagram [25].
- PIM (Platform Independent Model): it represents a model describing the business logic of a system, independently of any technology. It allows describing the structure of the entities which constitute the system. In object-oriented approaches, the UML class diagram is often used at this level, while in the service-oriented approach; the PIM is represented by SoaML models [14].
- PSM (Platform Specific Model) is a model that represents an implementation of a system according to a particular technology. MDA offers UML profiles to create these models, such as EJB profile (Enterprise Java Beans).

In order to establish traceability between CIM, PIM and PSM levels, MDA proposes the model transformation concept. These models must conform to their meta models. Thus, a metamodel is a model of a modeling language. The model transformation can be of three types:

- Simple transformation (1 to 1): it combines every element of source model with at most one element of the target model. An example of this transformation is the transformation of a UML class in a Java class.
- Multiple transformations (M to N): it takes as input a set of elements of the source model and produces a set of elements of the target model. Sometimes this transformation can be a type of composing models (1 to N) or merging models (N to 1).
- Update transformation: it is dedicated for changing a model by adding, modifying or deleting some of its elements.

In this paper, the composing models category (1 to N) is adopted. In fact, the idea is to transform a VR-BPMN configurable process model to four VarSOAML models which represents configurable services. The paper aims to offer a service generation method to generate configurable services from a configurable process, covering all perspectives of a configurable process. Thus, it is important to mention that the proposed approach will allow generating all the models representing a configurable service including contract, interface, Message Type and participants. These models will be transformed into configurable web services.

III. STATE OF THE ART

In this section, existing solutions in service generation are analyzed, and are evaluated how suitable for the purposes of this work they are. This analysis also provides valuable input regarding the requirements of this proposal.

All existing works on service generation [14] [16] [17] [18] [25] adopt MDA approach. Before analyzing these approaches, the evaluation criteria are listed below:

- **Transformation level** is about two kinds:
  - From business process model to service models (CIM2PIM)
  - From Services models to web services (PIM2PSM).
- **Representation language** determines the language used.
- **Variability** specifies whether the management of business process and service variability is assured.
- **Perspective** mentions the elements supported by the mapping rules.
- **Method** indicates if the approach proposes a method to assist the designers when generating services.

The following existing works are presented:

- MINERVA Framework [14]

In this work, authors develop a Framework called MINERVA (Model Driven & Service Oriented Framework for the continuous Business Process Improvement & related tools). MINERVA generates, from a BPMN business process models, SoaML service models (corresponding to CIMtoPIM transformation). The SoaML models are then transformed to

execution models represented in WSBPEL or XPDL (corresponding to PIMtoPSM transformation).The authors use a combination of Eclipse plugins (BPMN modeling, Medini QVT, Magic Draw and Model Pro) to implement their solution. However, no explicit definition of mapping rules at different levels of the Framework was found. Moreover, the approach does not support the concept of variability.

- BPMN-SoaML mapping [21]

In this work, authors define mapping rules for transforming BPMN models toSoaML models (CIMtoPIM).Thus, authors focus on the mapping of activities, Pool, Message Flow, and ignore the data and sub-processes. Regarding the target model, they focus on the contract, the service interface, the participant and the service architecture, messages and the choreographies. The rules are implemented using ATL (Atlas Transformation Language). However, the approach does not support the concept of variability.

-Business –IT alignment [17]

This approach is part of the business IT alignment based on a BPM-SOA convergence. In this work, authors propose a method to implement the business process as a service using the MDA approach. This work covers the mapping of the main perspectives of BPMN process models (business, sub-processes, Pool, Lan and process fragment) to the service model elements (Service Architecture, interface, contract and participant). A detailed definition of mapping rules is provided, as well as implementation in ATL language. However, this approach does not support the concept of variability.

- BPMN-SCA (Service Component Architecture) [16]

This work provides a mapping between business process models represented by BPMN and service models represented by SCA, which provides composition of applications using the principles of SOA. This approach focuses primarily on collaborative elements, participant and activity. The functional view and the service view is not supported by the approach. In addition, BPMN models supporting variability are not included.

- SVDEV [25]

This work proposes a development method SVDev (Service Variability Development) using the MDA approach. The SVDev development method is organized according to the levels of MDA:

- CIM level: it focuses on the study of the preliminary analysis and the specification of the business process models and classes.
- PIM level: it represents services (supporting variability) by VarSOAml. These models are organized in terms of views (functional, service, structure and composition).
- PSM level: it implements services witch support the variability by using VarWebService.

The Table III provides a summary of the following comparative study:

TABLE III. THE SERVICE GENERATION WORKS COMPARATIVE STUDY

Approaches	Transformation levels		Representation language		Variability		Perspective		Method
	CIM 2PIM	PIM2PSM	BP	Service	BP	Service	BP	Service	
[14]	Yes	Yes	BPMN	SOAml	-	-	-	-	No
[21]	Yes	-	BPMN	SOAml	-	-	Activity, Pool and message	contract, interface participant, service architecture, messages and choregraphies.	No
[17]	Yes	Yes	BPMN	SOAml	-	-	activité, sub processes, Pool, Lan	Service architecture, interface, contract and participant	Yes
[16]	Yes	-	BPMN	SCA	-	-	Activity, collaboration , conversation and participant	Component, and services	No
[25]	-	Yes	DA UML	VARSOAml	Yes	Yes	-	Participant variable, contrat variable, interface de service variable et message type variable	Yes

All approaches cover CIM2PIM level except SVDEV which covers only PIM2PSM. The most approaches use BPMN for business process models and SoaML for service models. Only SVDEV uses VARSOAML for service models. None of the approaches uses BPMN with supporting variability. Only the work [17] supports the mapping of all elements.

None of the analyzed works was found suitable to fulfill the needs of business IT alignment with supporting variability. In order to overcome these limitations, an MDA based method for aligning configurable services to configurable processes is proposed. This approach should enable the generation of configurable services from a configurable process while covering the generation of all views of a service.

IV. MDA Based Method for Configurable Services Generation

The main contribution that we look forward to in this work is the MDA based generation method (cf. Fig. 1) which is developed for PAIS designers. In what follows, all method steps are given.

Thus, the proposed method consists of the following steps:

- (1) Configurable process modeling
- (2) Decomposition of the configurable process into several fragments, each fragment represents a configurable service
- (3) Configurable service generation which consists of two sub steps :
  - VarSOAML configurable service generation which describes the configurable services associated with the identified fragments.
  - Configurable web services generation which corresponds to the service implementation.

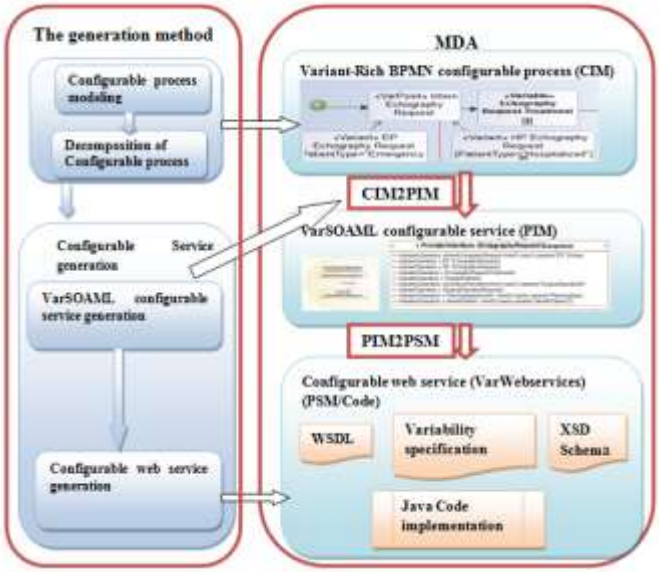

Fig. 1. The proposed MDA based method

A. Configurable process modeling

In order to illustrate the various steps of generation of configurable services, an echography request process represented in Variant-Rich BPMN (see Fig. 2) is used. This configurable process includes three variants: echography request of a simple patient (variant 1), echography request of a hospitalized patient (variant 2) and echography request of a emergency patient (variant 3):

- Variant 1: the request is made by the patient. It is then received by the assistant who is responsible for the management of patients and then studied by a hospital actor (the Cardiologist Doctor) who is responsible of patient examination.
- Variant 2: the request is made by a hospital Actor (doctor) and received by the assistant. The same process as a simple patient is applied.
- Variant 3: the request is sent by a hospital actor (emergency doctor), then it is sent directly to the cardiologist to perform an emergency examination.

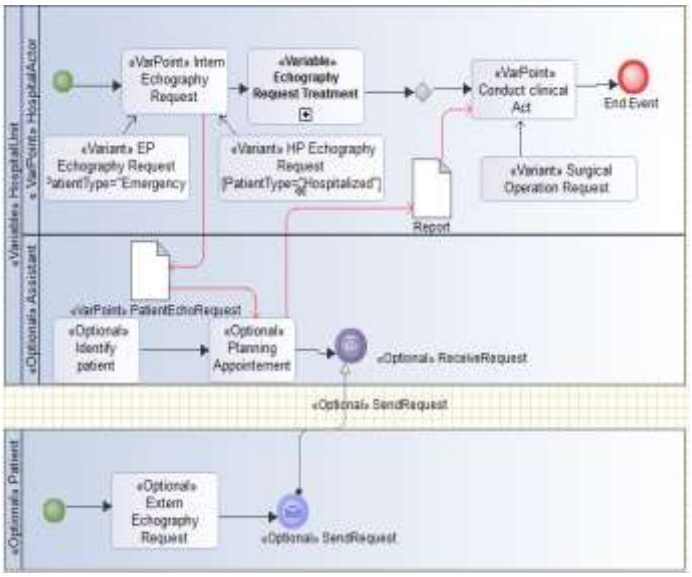

Fig. 2. An echography request configurable process

For each variation point activity, the type of the variation point (through the stereotype) and the associated variants are indicated, and for each variant activity, the feature helping the designer in the variability resolution step ({PatientType = "Hospitalized patient"}) is represented.

B. Decomposition of configurable process

The decomposition of a business process is to extract from a business process model, a set of fragments that encapsulate a business objective. A fragment is identified from a series of sequential activities or from a Gateway. A fragment corresponds to a business service [26].

In this work, the decomposition of a configurable process (shown in Variant-Rich BPMN) is to extract several configurable fragments (supporting variability). These configurable fragments correspond to configurable composite activities. A composite activity is called configurable if it contains at least one variation point or variable activity. A configurable fragment is identified from a series of sequential activities carried out by the same participant. For each configurable fragment identified, a configurable service will be associated.

Thus, the configurable process can be decomposed as follows (see Fig. 3):

- The composite activity

« EchographyRequestManagement» consists of:

- A variation point activity «InternEchographyRequest»
- A Variable activity «EchographyRequestTreatment»
- A variation point activity «Conduct Clinical Act»
- An optional activity «Planning Appointment»
- An optional activity «IdentifyPatient»

- The composite activity «PatientEchographyPatient» consists of:

- An optional activity "ExternEchographyRequest»

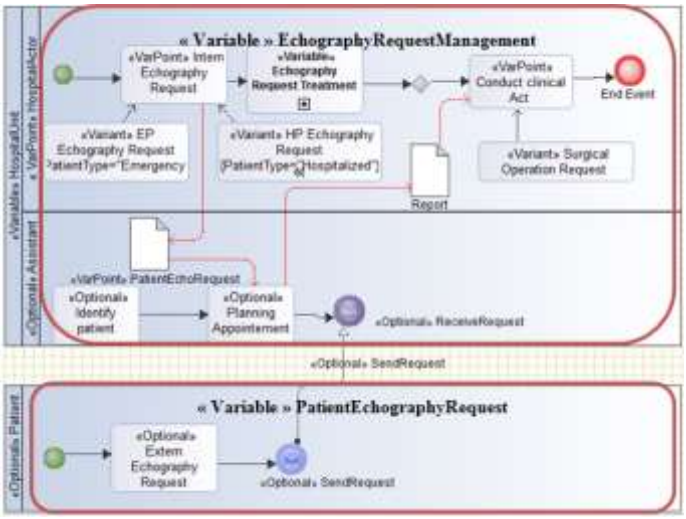

Fig. 3. Decomposition of the configurable process

Each configurable composite activity is associated with a configurable service. Two configurable services:

- EchographyRequestManagement
- PatientEchographyRequest

For each identified configurable composite activity, the service interface, the messages, the service contract and participants are generated.

C. VarSOAML configurable service generation

Before generating a VarSOAML models of a configurable service, it is first to identify the elements of the source metamodel and the target metamodel of those affected by this generation.

The type of transformation which is operated is the multiple transformation (1 to 3).A Variant-Rich BPMN model is transformed to four VarSOAML models by applying generation rules.

Example of the proposed generation rule [15]:

The rule for the transformation of the VariableCompositeActivity element to ProviderInterface or ConsumerInterface elements is given, 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
        VariableCompositeActivityelement Do
           Create an element of
           ConsumerInterfaceOrProviderInterface
           types
           The name of the ConsumerInterfaceOr
           Provider element is the name of the
           VariableCompositeActivityelement
           If (the IncomingMattribute is Null)
           of the first SimpleActivity or
           VariationPointActivity or Event
           elements contained in the
           VariableCompositeActivityThen
            Create ConsumerInterface element
           Else
           Create ProviderInterface element
           Apply
           VariableCompositeActivity2VariableIn
           terface//Create aVariableInterface
           element which represents the service
           interfacewhich provides the
           ProviderInterface
           End If
        For each element SimpleActivity element
        in VariableCompositeActivityDo
           Apply SimpleActivity2Operation
        End For
        For each VariationPointActivity element
        Do
           Apply
           VariationPointActivity2VariationOper
           ation
        End For
```

The example of VarSOAML models which represent the service associated with the composite activity «EchographyRequestManagement» is given bellow.

a) Service contract model

By applying the rule

VariableCompositeActivity2VariableContract [15] the service contract model «EchographyRequestManagement» can be generated (cf. Fig. 4).

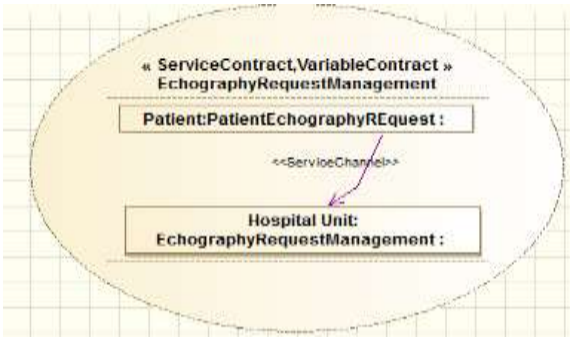

Fig. 4. Service contract model

b) Service Interface Model

By applying the rules proposed in [15]:

VariableCompositeActivity2VariableInterface

- VariableCompositeActivity2ProviderInterface & ConsumerInterface
- VariationPointSimpleActivity2VariationOperation
- VariantSimpleActivity2VariantOperation
- SimpleActivity2Operation

The configurable service interface is given bellow (cf. Fig. 5).

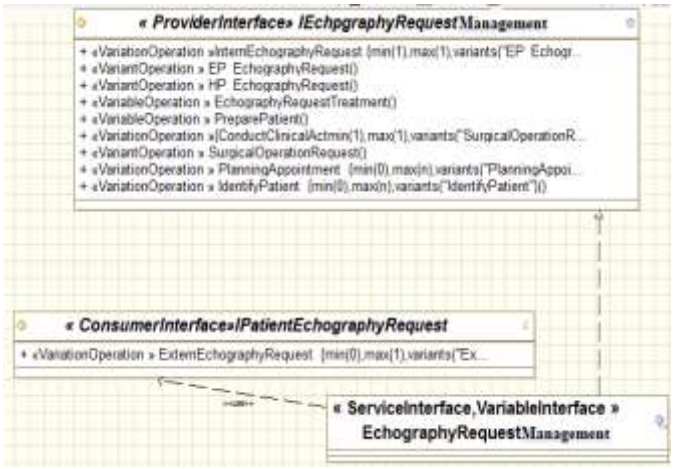

Fig. 5. Service Interface model

This interface uses the required interface «IPatientEchographyRequest» and provides «IEchographyRequestManagement».

c) Message Type Model

By applying the rules [15]:

- VariationPointActivity2VariableMessage
- VariableActivity2VariableMessage
- VariationPointDataObjectInput2VariationType
- VariationPointSimpleDataInput2VariableAttribut

The Message Type model is exposed in (cf. Fig. 6)

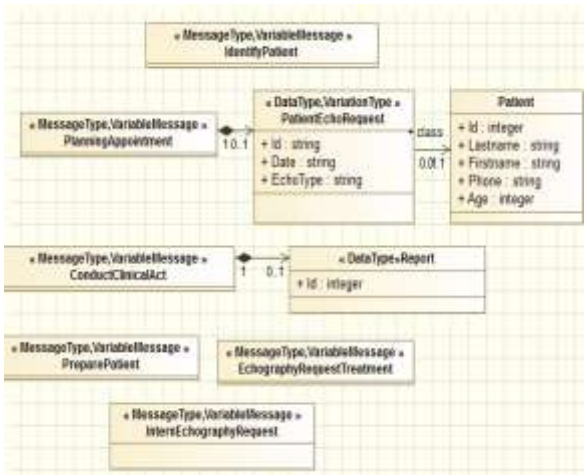

Fig. 6. Message Type model

The model describes the variables and simple messages. The model also describes the data contained in the message.

D. Configurable web services generation

In this section, the generation of configurable web services (called VarWebservice) from VarSOAML models is detailed.

The VarWebService generation approach proposed in [25] is applied.

This work establishes the transformation of VarSOAML models to configurable web services. Thus, a web service is defined as a service that is accessible via the Internet and uses the XML standard. It is a software module that exposes the interface through a WSDL (Web Service Description Language). WSDL is a language for describing all operations and messages that can be exchanged [27].

The generation of configurable web services requires generation of the following files:

- WSDL file
- Variability specification file associated with the WSDL file
- XSD schema which describes data which are used by the web service
- Java code implementation

1) WSDL file generation

The WSDL file (cf. Fig. 7) contains all the messages and operations. The following generation rules are applied:

- ServiceInterface2WSDL
- Operation2Operation
- MessageType2Message

Example of generation rule: ServiceInterface2WSDL

The ServiceInterface2WSDL rule enables the creation of instances of Binding, PortType, Schema Types ('WSDL') elements from the instance of ServiceInterface element.

```
Name rule: ServiceInterface2WSDL
Input Elements: ServiceInterface (VarSOAML)
Output elements: Binding, PortType, Schema
Types, Definition ('WSDL'
For each ServiceInterface Do
      Create an element of wsdl
            Create the Binding attribut
            The name of Binding is the name
            of ServiceInterface concatenated
            with the string 'Binding'
            Create the style attribute
            The name takes the value
            "document"
            For each Operation Do
                  Apply Operation2Operation
                  rule
            End For
End For
```

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
name="EchographyRequestManagement"
targetNamespace="urn:// 
EchographyRequestManagement.wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/so
ap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ps="urn://EchographyRequestManagement
Schema.xsd"
xmlns:tns="urn:// 
EchographyRequestManagement.wsdl"
xmlns:wsdl=http://schemas.xmlsoap.org/wsdl/
xmlns="http://schemas.xmlsoap.org/wsdl/">
<wsdl:import namespace="urn:// 
EchographyRequestManagement Schema.xsd"
location=" EchographyRequestManagement
Schema.xsd"> </wsdl:import>
<wsdl:types></wsdl:types>
<wsdl:message name="return">
<wsdl:part name="partreturn" type="xsd:int">
</wsdl:part>
<wsdl:message name="PlanningAppointment">
<wsdl:part name="part PlanningAppointment"
type="ps:PatientEchoRequest"/>
</wsdl:message>
<wsdl:message name="ConductClinicalAct">
<wsdl:part name= "part ConductClinicalAct"
type="ps: Report"/>
</wsdl:message>
<wsdl:portType
name="EchographyRequestManagementPortType">
<wsdl:operation name="PlanningAppointment">
<wsdl:input message="tns: PlanningAppointment"
name="PlanningAppointment_Request"/>
<wsdl:output message="tns:return"
name="PlanningAppointment_Response"/>
</wsdl:operation>
<wsdl:operation name="ConductClinicalAct">
<wsdl:input message="tns: ConductClinicalAct"
name="ConductClinicalAct_Request"/>
<wsdl:output message="tns:return" name="
ConductClinicalAct_Response"/>
</wsdl:operation>
</wsdl:portType> </wsdl:definitions> …
```

Fig. 7. An extract of the WSDL file

The VarSOAML source elements affected by this generation are: the interface provided by the service interface and messages associated with this interface.

2) Generation of the variability specification associated with the WSDL file

The specification of the variability associated with the WSDL file describes the variable operations, messages, types and variables attributes (cf. Fig. 8).

```
<variability service
="EchographyRequestManagement" name
="EchographyRequestManagementVariability">
<operations>
<variationOperation name =
"InternEchographyRequest" min = "1" max =
"1" portype =
"EchographyRequestManagementPortType">
</variationOperation>
<variantOperation name = "EP
EchographyRequest"
</variantOperation>
</operations>
<messages>
<variablemessage name = "
PlanningAppointment" scope ="configurable"
boundElement = " PlanningAppointment"
description ="">
<types> <type>RequestEcho</type> </types>
</variablemessage>
<variablemessage name = "
ConductClinicalAct" scope ="configurable"
boundElement = "ConductClinicalAct"
description ="">
<types> <type>Report</type>
</messages>
…
</variability>
```

Fig. 8. An extract of the Variability specification associated with the WSDL file

3) XSD schema generation

The XSD schema describes all data types used by a web service.

In order to generate the XSD file (cf. Fig. 9), the generation rules proposed in [25] are applied:

- ServiceInterface2Schema
- MessageType2Message
- Attribute2Element
- DataType2Type
- PType2SimpleType

Example of the generation rule: Data Type2Type:

The DataType2Type rule allows for creating an instance of the web service ComplexType from an instance of the VarSOAML DataType element.

```
Name rule: DataType2Type
Input Element: DataType (VarSOAML)
Output element: ComplexType ('XSD')
```

```
For each DataType element Do
       Create ComplexType element
       The name of the ComplexType is the name
       of the Datatype
       For each attribute element Do
              Apply the Attribute2Element rule
       End For
End For
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
xmlns:tns="urn://
EchographyRequestManagement Schema.xsd"
targetNamespace="urn://
EchographyRequestManagementSchema.xsd">
<xsd:complexType
name="sequencePatientEchoRequest">
<xsd:sequence>
<xsd:element name="PatientEchoRequest"
type="tns: PatientEchoRequest" />
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="PatientEchoRequest">
<xsd:sequence>
<xsd:element name="id" type="xsd:integer" />
<xsd:element name="Date" type="xsd:string"
/>
<xsd:element name="EchoType"
type="xsd:String" />
<xsd:element name="Patient"
type="xsd:Patient"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Patient">
<xsd:sequence>
<xsd:element name="id" type="xsd:string" />
<xsd:element name="Lastname"
type="xsd:string" />
<xsd:element name="Firstname"
type="xsd:string"/>
<xsd:element name="Phone" type="xsd:string"
/>
<xsd:element name="Age" type="xsd:integer"
/>
</xsd:complexType>
<xsd:complexType name="Report">
<xsd:sequence>
<xsd:element name="id" type="xsd:string" />
<xsd:element name="ReportSubject"
type="xsd:string" />
<xsd:element name="Date" type="xsd:string"
/>
<xsd:element name="Act" type="xsd:string" />
</xsd:schema>
```

Fig. 9. An extract of the XSD schema associated with the WSDL file

4) Java code implementation of configurable web service

The following generation rules are applied in order to generate the java code implementation [25] (cf. Fig. 10):

- ServiceInterface2Interface
- Operation2WebMethod
- Attribut2WebParam

Example of the generation Rule: Operation2Method

The Operation2Method rule allows creating of the Method instance from Operation.

```
Name rule: Operation2Method
Input Element: Operation (VarSOAML)
Output elements: Method, WebMethods ('JAXWS')
Description:
For each Operation element Do
       Create Method element
       The name of Method parameters is the
       name of the parameters and return type
       of the Operation element
       For each Parameters Do
              Apply Param2Par rule
                     Creates WebMethod
                     The name of OperationName
                     is the name of the
                     operation element
       End For

End For
```

The java code of the web service EchographyRequestManagement is generated (cf. Fig. 10):

```
package EchoRequestWebService;
import javax.jws.*;
import javax.jws.soap.SOAPBinding;
import javax.xml.ws.Endpoint;
@WebService(serviceName="EchographyRequestMan
agementService")
public class EchographyRequestManagementClass
implements
EchographyRequestManagementInterface {
@VariantOperation
(boundOperation="EP_EchographyRequestPatient"
)
public EP_EchographyRequestPatient()
{ …
}
@VariantOperation
(boundOperation="HP_EchographyRequestPatient"
)
public HP_EchographyRequestPatien()
{ …
}
}
```

Fig. 10. An extract of the Java Code implementation

The advantage of this method is that it covers all stages of configurable services development, from configurable process modeling to configurable web services implementation which is described by the WSDL file, the variability specification, the XSD schema and the Java code implementation.

V. Conclusion

Many solutions for business IT alignment have been proposed. However, some limitations such as the weak support of business process and service variability are underlined. The MDA based method for service generation ensures the business IT alignment with managing variability.

It allows for decomposing configurable processes into set of configurable composite activities. Thus, a set of configurable services VarSOAML generation rules were proposed with the aim of generating different VarSOAML models. The advantage is that these models are then automatically implemented as configurable web services which requires the generation of the WSDL file, the variability specification, the XSD file and the java code implementation. In this sense, VarSOAML models conduct the implementation of configurable web services.

Indeed, the generation of configurable services from a configurable process provides better synchronization of changes between the business and IT layers, it will facilitate the propagation of changes from configurable process to configurable services.

Furthermore, the proposed business IT alignment is not sufficient as it excludes changes that may come from the service layer. As future work, it will be necessary to offer a bottom up approach to align business process with services. Another possible improvement is to incorporate the semantic aspect to enrich configurable services in the context of business IT alignment. This can allow for developing intelligent configurable web services.

REFERENCE

[1] M. Dumas, W.M.P. Van der Aalst and A.H.M. ter Hofstede,"Process-Aware Information Systems", Wiley, 2005.

[2] C. Ayora, V. Torres, B. Weber, M. Reichert ,V. Pelechano, "Enhancing Modeling and Change Patterns. BMDS/EMMSAD", CAiSE 2013, Valencia, Spain, pp.246-260, 17-18 Juin, 2013.

[3] M. La Rosa,"Managing variability in PAIS", PHD thesis, Faculty of Science and Technology, Queensland University of Technology, Brisbane, Australia, 25 Mars 2009.

[4] M. Rosemann , W. M.P Van der Aalst,"A Configurable Reference Modelling Language ", Information Systems Vol.32, N°1, pp.1–23, 2007.

[5] A. Schnieders , F. Puhlmann,"Variability modeling and product derivation in ebusiness process families", Technologies Business Information System Book, Springer, pp. 63–74, 2007.

[6] A. Hallerbach ,T. Bauer, M. Reichert,"Capturing variability in business process models: The Provop approach", Journal of Software Maintenance and Evolution: Research and Practice, Vol.22, N°7, pp.519–546, 2010.

[7] V. Kulkarni, S. barat,"Business process families using model-driven techniques Lecture Notes in Business Information Processing", Vol.66, pp 314-325, 2011.

[8] A. Kumar, W. Yao,"Design and management of flexible process variants using templates and rules,», Computers in Industry Vol.63, N°2, pp.112–130, 2012.

[9] T. Nguyen , A. Colman, J. Han ,"Comprehensive Variability Modeling and Management for Customizable Process-Based Service Compositions", in : A. Bouguettaya, Q. Z. Sheng, F. Daniel, Web Services Foundations, Springer, pp.507–534, 2014.

[10] A. Yousfi, R. Saidi and A.K. Dey,"Variability patterns for business processes in BPMN. *Information Systems and e-Business Management*, pp:1-25,2015.

[11] C. Ayora, V. Torres, J L. De la Vara, V. Pelechano,"Variability management in process families through change patterns", Information and Software Technology, Vol.74, pp. 86-104, 30 June 2016.

[12] B. Weber, M. Reichert, S. Rinderle," Change Patterns and Change Support Features: Enhancing Flexibility in Process-Aware Information Systems", Data and Knowledge Engineering, Vol.66, N°3, pp. 438–466, 2008.

[13] H. Sbai, M. Fredj, L. Kjiri," A pattern based methodology for evolution management in business process reuse", IJCSI International Journal of Computer Science Issues, Vol. 11, Issue 1, N°1, pp. 211-220, January 2014.

[14] A. Delgado, F. Ruiz, I.G.R. de Guzman, M. Piattini,"A model-driven and service-oriented framework for the business process improvement", Journal of Systems Integration, Vol.1, No° 3, pp. 45–55, 2010.

[15] H. Sbai, M. Fredj, B. Chakir,"Generating services supporting variability from configurable process model", Journal of Theoretical and Applied Information Technology, Vol. 72, N°2, pp. 111-124, February 2015.

[16] K. Da man, F. Charoy, C. Godart,"Alignment and change propagation between business processes and service-oriented architectures», International Conference on Service Computing (SCC'13), Santa Clara, CA, United States, pp. 168–175, 27 June - 02 July, 2013.

[17] B. Elvesæter, D. Panfilenko , S. Jacobi and C. Hahn,"Aligning business and IT models in service-oriented architectures using BPMN and SoaML,"Proceedings of the First International Workshop on Model-Driven Interoperability (MDI '10), Oslo, Norway, 3-5 Octobre, 2010.

[18] A. Kabzeva and P. M¨uller," Toward Generic Dependency Management for Evolution Support of Inter-Domain Service-Oriented Applications", European Conference on Service-Oriented and Cloud Computing (ESOCC 2012), Bertinoro , Italy, pp.35-40 ,2012.

[19] Y. Wang, J. Yang, W. Zhao,"Change impact analysi"s for service based business processes. In the IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1-8, 13 December, 2010.

[20] J. Simonin, P. Picouet, J.M. Jézéquel, "Conception fonctionnelle de services d'entreprise fondée sur l'alignement entre cœur de métier et système d'information", Ingénierie des systèmes d'information, Vol. 15, N°4, pp.37-61, 2010.

[21] Y. Lemrabet, J. Touzi, D. Clin, M. Bigand, J.P. Bourey, "Mapping of bpmn models into uml models using soaml profile", 8th International Conference of Modeling and Simulation (MOSIM'10), Hammamet, Tunisia, pp. 10-12 Mai, 2010.

[22] BPMN, Business Process Modeling and Notation, Version 2.0. Object Management Group (OMG) http ://www.bpmn.org/ (dernière consultation Mars 2015).

[23] SOAML, "Service oriented architecture modeling language (SoaML) – specification for the uml profile and metamodel for services (UPMS)", Version 1.0, 2012.

[24] J. Bézivin, "Sur les principes de base de l'ingénierie des modèles", ISSN 1262-1137, Vol. 10, No 4, pp. 145-156, 2004.

[25] B. Chakir, "Contribution à l'amélioration de la variabilité des services par la gestion de la variabilité ", doctoral dissertation, Mohammed V University of Rabat, March 2014.

[26] M. Radgui, "Décomposition et adaptation de processus métiers BPMN pour des systèmes d'information flexibles ", Mohammed V university of Rabat, Morocco, June 2015.

[27] W3C, Web Services Definition Language (WSDL) 1.2, : http://www.w3.org/TR/2003/WD-wsdl12-20030611/, 2003.