Service-Oriented Approach for Agile Support of Product Design Processes

Safa Hachani[1], Hervé Verjus[2], and Lilia Gzara[1]

[1] G-SCOP Laboratory,
Grenoble Institute of Technology,
Grenoble, France
{safa.Hachani,lilia.Gzara}@grenoble-inp.fr
[2] LISTIC Laboratory,
University of Savoie,
Annecy, France
herve.verjus@univ-savoie.fr

Abstract. The need to answer quickly to new market opportunities and the high variability of consumer demands tend industrial companies to review their adopted organisation, so to improve their reactivity and to facilitate the coupling with the business enactment. Therefore, these companies require agility in their information systems to allow business needs scalability and design process flexibility. We propose in this paper, the business activities as a service based on the service paradigm and whereas a design process is made of agile services orchestrations. We discuss the interest to use a service-oriented approach and propose a layered architecture for design process enactment.

Keywords: Design process, agility, PLM, SOA, service orchestration, MDA.

1 Introduction

Traditional business processes (BPs) with a fixed structure are no longer adequate to meet the continuing evolution of the market, enterprises' organization and custumers'expectations. Traditional BPs tend to be inflexible and time consuming to change. In fact, in a context where organizations are in a constant seek of balance facing up to more and more constraints of the competitive environment; working methods cannot be fixed definitively. This is due to product design processes (PDPs) specificities; they are emergent and non deterministic because of the creativity aspect in product design projects. Furthermore, unpredictable events often occur during PDPs due to external constraints (such as sub-contractor or supplier constraints, etc.) and/or internal constraints (such as delay constraints, staff/resources availability, etc.). Some of these factors, such as satisfying suppliers' needs, may only cause temporary changes of PDPs. While others, such as regulation evolution may cause permanent changes. PDPs are thus constantly changing. Reflecting these changes on time represents an ongoing challenge. As a result, companies face some obstacles, including the limited implementation of new working methods. Needs in terms of rapid and automated support of business operations[1] are necessary to reflect such changes. Service oriented architecture (SOA) can address this issue by facilitating enterprise solutions flexibility and therefore business agility. SOA have different perspectives depending on the user roles (business, architectural or implementation perspective) [1]. Our research focus on the first perspective (*i.e.* business one) and we consider SOA as an architectural style that supports integrating the business as linked services. It can support BP goals and enterprise objectives. So, we can say that a service-oriented approach based on the concept of service[2] can promote a support of flexible PDPs by considering a PDP as a series of services snapped together like building blocks. SOA makes this possible; it allows decomposing processes into independent business services. Then, using the business services cartography, the services can be bands to realize BP.

Several researches have attempted to solve the problem of BP rigidity. Some of them propose to enrich the expression of BP in order to meet flexibility needs. We provide an overview of these related works in second 2 in order to release their limits and disadvantages towards our objectives. Subsequently we present the approach we retained which is based on service oriented approach (section 3). Then, we present our services catalogs (section 4). Section 5 presents the feasibility of the proposed approach. The final chapter concludes this paper.

2 Related Works

Currently, there are scores of BP modeling approaches, which aim to manage flexibility. Bessai proposes a context aware-based approach [2]. The concept of context is used to characterize an activity (designed to operate in a particular state of the world). In this approach the role of the process is to precisely define the order and the context of activities to be performed. The one that should be executed is chosen according to the objects state. Zhao and Liu propose a version-based approach [3]. This work proposes to model versions of BP schemas using graphs: nodes of a graph correspond to activities of a process while arcs between nodes represent the used patterns to link activities. This work also presents a set of operations enabling updates of graphs and defines two strategies to extract versions of BP schemas from these graphs. Lezoche's work proposes a rule-based approach [4]. This approach considers that activities are components reacting to events. Here, the result of one activity represents an initiating event for other ones according to given set of rules. Indeed, an instance of the process model will correspond to one of planned paths. Boukadi resorts to SOA in order to deal with BP flexibility [5]. She focuses on building flexible BP based on service composition and service adaptability. Boukadi's work defines a BP as a sequence of activities. The creation of the BP is essentially based on selecting one or more goal-templates and specifying the flows connecting those goals. The goals-templates

[1] We highlight that PDPs are mainly supported by PLM (Product Lifecycle Management) systems. So, in the rest of this paper we are interested on a solution that supports flexible PDPs enactment in PLM.

[2] Services are repeatable business tasks, accessible independently of implementation or transport [1].

correspond to the tasks to be completed. The composition of the different goals-templates leads to an abstract BP model. Based on the goals-templates, services will be selected to replace them and the executable BP will be generated. The offered services are adaptable and have different behaviors depending on the context of their uses.

To sum up, on the one hand, the proposed approaches [2, 3, and 4] allow modeling a BP with maintaining some flexibility in how to meet business objectives by using conditions and/or triggering event to choose between versions of activities. The disadvantage of using these modeling approaches is that they imply having anticipated all scenarios. These approaches are well adapted to representation of process with a limited degree of complexity. However, it is hard to use them in order to design process with emerging structure. On the other hand, Boukadi's work allows modeling reusable activities unlike other works (rule, context-aware and version-based approaches), which make possible to design decoupled activity that can be used by different BPs. Besides, by defining the flows and fixing the goals-templates in advance, this decreases the flexibility of the BP. Her vision of flexibility is limited to the capability to change the manner with which the activities can meet the objectives rather than focusing on the flexibility of the process structure.

Based on the findings outlined above, we conclude that flexible BPs require specific methods for their design and implementation. Thus, the expected approach is the one in which not only the behavior of activities are not defined a priori but also the relations between activities. We resort to service based solution in order to map PDPs to service-oriented solution. The idea is to propose reusable activities as services and evolvable PDPs as services composition. The concept of service defined as providers of reusable functions can be composed and reused to quickly respond to PDP changes. That supposes once changes occur we can add to, delete from or replace one service by another one. Indeed, the generalization of SOA to information systems (and thus, the design and requirements analysis layers) would allow the definition of PDPs and their implementation by reusing existing services.

3 The Proposed Service-Based Approach

3.1 Product Design Process as a Service Orchestration

The aim of a PDP model is to depict interactions between business partners and model their corresponding activities. In past decades, these processes could operate in relatively stable and predictable environments. Now a product design process may not remain steady due to the business environment. That's why we need process flexibility. Process flexibility depends on the easiness to modify PDP model and to set up the new business activities. This perception of process flexibility arises from the need to have a method, which allows composing evolvable PDP models. In other words, flexibility requires processes made of piece of functionalities that can work together and that can be quickly reconfigurable. The challenge here is to address the mechanisms needed for a solid implementation of dynamic PDP change on a real PLM system.

In order to address the problem of PDP rigidity in PLM systems, we propose to resort to SOA and to enrich the formalization of PDP models and open the way for process modeling by dynamic service composition. Thus, we should have a set of product design services (PDSs) that expose the business activities of the industrial engineering domain needed to support PDPs. Afterward, we dynamically compose the necessary identified PDS in order to enact the articulations of PDP. This PDP can be materialized by an orchestration of PDS [6]. In fact, we propose to use service as a means for describing the business operations needed to support the PDPs. These loosely coupled services may be composed in order to enact in a flexible way the articulations of business. This process is called services orchestration [7]. SOA makes this possible; it allows decomposing processes and business activities into independent business services. Dynamic services orchestration stands for assembling and reassembling these business services while the process is executing. Thus, service can be composed and reused to quickly respond to PDP change and to achieve the new model without needing to replace it completely. Moreover, as services expose operations independently of their real enactment, they can be reused even if the enactment is changing (as consequence, some changes do not affect the services orchestration). This supposes that once a change happens, we can dynamically add to, delete from or replace a service operation with another one. The main characteristic of this service-based approach is that it provides a flexible process structure, which provides the necessary agility to face changing conditions and unpredictable situations.

3.2 Conceptual Architecture

As we have discussed above, business agility is the fundamental business requirement. So, the entire PLM system, starting by IT level, must support business agility. It's important to remember that PDPs are very dependent on the information technology that supports it. So, the business also depends of the IT flexibility. So, we propose the whole system reconsideration and not only a business level reconsideration.

To insure the alignment between technical and business level, there should be a mechanism that allows execution of PDP with the same language chosen for the business level. So, we propose a service type for each level of the organization. The different levels that we consider are justified by the reality of enterprise information system. On the one hand, we find the organizational IS. It consists of information (business objects) and actors whose act on this information through their work methods (business activities). On the other hand, there is the system infrastructure or Computerized IS. The computerized IS consists of an organized set of items, machines and applications. It allows the implementation of the working methods of the companies and the organization of their information. Moreover, the actors of the organizational IS use the computerized IS through the interfaces provided by its tools. Therefore, there are three levels in the enterprise IS: the business level associated with organizational IS, the technical level associated with Computerized IS and the functional level associated with the interfaces of the Computerized IS. Regarding this classification, we propose three layers of services. First, we propose a catalog of Product Design Services (PDS). A PDS is a collection of PDS operations reflecting solutions to some business needs of a product design domain. In fact, each PDS operation partly reflects a business activity typically presented in PDPs. These PDS operations will be used and composed by BP designer to build PDPs. Secondly, we propose a catalog of Functional PLM Services, which (i) ensures alignment between business and technical levels and (ii) aims to be independent of any PLM system. A Functional PLM service is a collection of Functional PLM services operations. The services operations of this layer represent all functions of the PLM as seen by PLM users independently of any existing PLM tool. A set of functional PLM services operations support a PDS operation. They will be used by process performers and composed to achieve the PDS. In other words, once a new business activity is needed to perform a change at the business level (in a PDP), a PDS operation is invoked and added to the orchestration and thus operations of functional PLM services can be solicited from the repository to do it. Finally, we propose a set of technical PLM services that allow the real implementation of functional PLM services. These Technical PLM services cover technical operations carried out in a PLM system and they will be dynamically orchestrated during the enactment of PDPs. They are intended to PLM systems editors. This distinction between functional and technical PLM services allow the reuse of process models, defined only in terms of business and functional PLM services, on different PLM systems. Indeed, once a process deployed on a new PLM system, we have to make correspondence between functional PLM services and technical PLM services.

To make the transition from one level to another one, our conceptual approach is based on a model-driven engineering one (MDE) (Fig. 1). Starting at the top with the business level (Computer Independent Model) it is primarily concerned by the PDPs that comprise the day-to-day activities of the enterprise. It contains also the business services (PDS), which allow composing the PDP. Moving down a level (Platform independent Model), we see the functional PLM services and orchestration fragments that can be predefined or user-defined. Predefined orchestration fragments define the recommended functional PLM services orchestration, which allows fulfilling a given PDS operation.

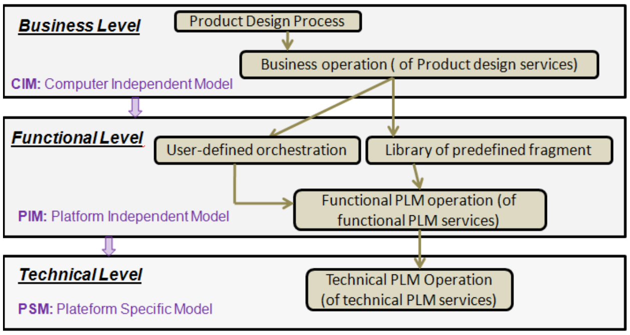

Business Level
Product Design Process
Business operation (of Product design services)
CIM: Computer Independent Model
Functional Level
User-defined orchestration
Library of predefined fragment
Functional PLM operation (of functional PLM services)
PIM: Platform Independent Model
Technical Level
Technical PLM Operation (of technical PLM services)
PSM: Plateform Specific Model

Fig. 1. Conceptual Architecture of the Proposed Approach

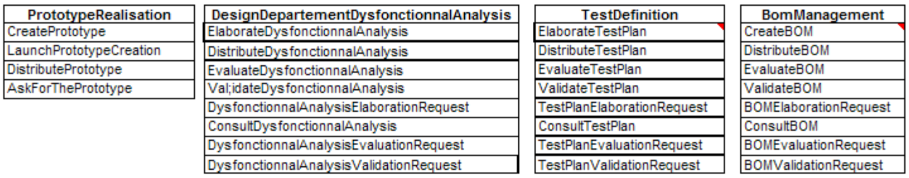
Functional PLM services are non-specific to any PLM system. That's why services can be mixed and matched into meaningful combinations without concern for what systems are actually performing the work. Down this, we find the technical level (Platform Specific Model), which contains the technical services. This layer forms the API (core functions) of the PLM system used.

In order to achieve the proposed approach, first we have to propose the services catalogs. Then we have to express PDPs as service orchestration. Finally we should propose alignment techniques that allow moving from business to technical level. In the rest of this paper we concentrate on the two first steps of our approach (services identification and PDP definition as service orchestration).

4 Service Cartography

As we have discussed above, we aim to offer three services catalogs. A PDS catalog, which expresses the business, needs related to PDPs. Functional PLM services catalog enabling the execution of PDS through a functional PLM services orchestration. Finally, technical PLM services catalog enabling the implementation of functional PLM services. The third catalog is dependent on which PLM system we use, that's why we concentrate on the business and functional service catalogs. For the third catalog, we analyzed existing standard in terms of services for PLM systems. In a previous work we defined the appropriate techniques and necessaries steps to achieve our service identification approach [8]. We referred to literature in the domain of SOA development, like Papazoglou [9], Chang [10] and Arsanjani [11] methods, in order to propose our identification method. More precisely, we proposed a top-down method for the identification of PDS. For the derivation of functional PLM services, we proposed a bottom-up method. We note here that the identification approach cannot be the same for high and low level services. Since they don't have the same service consumer and expose different type of information. High-level services expose business needs. That's why we proposed a top-down method, which consists of the analysis of the business domain. Low-level services expose application functionalities. Since most organizations have existing application systems in place we proposed a bottom-up approach, which consists of the analysis of application functionalities.

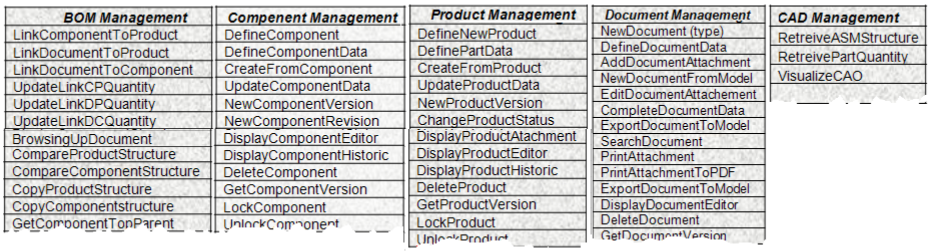
Regarding PDP context, a PDS is a collection of operations achieved during PDP, by design actors. Hence, each operation of PDS consists of one product design action which operates on one or more business objects (i.e. product design object). A business object is a concept or abstraction that makes sense for product design domain actors and corresponds to the entities manipulated by actors during design. To identify design objects we analyzed interviews done within four French companies, organized meetings with product design domain experts in order to enumerate the shared vocabulary and referred to literature to analyze PDP models in order to determine the different steps and their corresponding outcome (*i.e.* a business objects) [13] and [14]. We identified the following list of product design objects: Delay, Quotation, Quantity, Market analysis sheet, Drawing Assembly, 3D Assembly, Prototype, Validation Report, Design Solution, Bill of Materials, Drawing Part, etc. The product design object is used to describe the entities manipulated by the actors within the description of the job (i.e product design action). A product design action is an act that can be done by an actor in order to achieve a specific outcome. We identified a list of product design actions such as produce solution, produce decision/agreement, request information, request evaluation, etc. Thus, the PDP can be described as a network of product design actions operating on product design objects and iterating continuously to produce a result. As a result, we obtained a set of product design operations for which we have defined meaningful names (eg. Elaborate Test Plan, Create Prototype, etc.) and grouped them together on a PDS. The product design operations are grouped based on service cohesion principle as operation should be grouped together that are functionally related. Other criteria such as business rules and sequence logic were used to decide which operations must be grouped together. In fact, during the analysis of design process model we identified recurring cycle between product design actions. For example, *ElaborateTestPlan*, *TestPlanEvaluationRequest*, and *EvaluateTestPlan* are usually used together. This is justified by the fact that there is a cycle between the formulation of the problem and its resolution. We obtained fifteen PDS. Figure 2 shows an expert of this catalog.



PrototypeRealisation	DesignDepartementDysfonctionnalAnalysis	TestDefinition	BomManagement
CreatePrototype	ElaborateDysfonctionnalAnalysis	ElaborateTestPlan	CreateBOM
LaunchPrototypeCreation	DistributeDysfonctionnalAnalysis	DistributeTestPlan	DistributeBOM
DistributePrototype	EvaluateDysfonctionnalAnalysis	EvaluateTestPlan	EvaluateBOM
AskForThePrototype	Val;idateDysfonctionnalAnalysis	ValidateTestPlan	ValidateBOM
	DysfonctionnalAnalysisElaborationRequest	TestPlanElaborationRequest	BOMElaborationRequest
	ConsultDysfonctionnalAnalysis	ConsultTestPlan	ConsultBOM
	DysfonctionnalAnalysisEvaluationRequest	TestPlanEvaluationRequest	BOMEvaluationRequest
	DysfonctionnalAnalysisValidationRequest	TestPlanValidationRequest	BOMValidationRequest

Fig. 2. An expert from product design service catalog

A functional PLM service is a collection of PLM operations which reflect generic PLM functions that should be (at least partially) implemented by PLM systems. Each operation implements the concept of automated business task and exposes a function of PLM that can be reused and composed based on business needs. Thus we have identified three major categories of PLM data: product management, product process management and organization. We then decomposed each category into sub-categories and for each sub-category we have identified a set of operations offered in PLM systems (for instance, "Display product structure" and "Compare BOMs" for the product configuration sub-category).We have classified the identified operations in functional PLM services based on two criteria functional dependencies and process dependencies. As result we obtained nine functional PLM services (Fig. 3).



BOM Management	Compenent Management	Product Management	Document Management	CAD Management
LinkComponentToProduct	DefineComponent	DefineNewProduct	NewDocument (type)	RetreiveASMStructure
LinkDocumentToProduct	DefineComponentData	DefinePartData	DefineDocumentData	RetreivePartQuantity
LinkDocumentToComponent	CreateFromComponent	CreateFromProduct	AddDocumentAttachment	VisualizeCAO
UpdateLinkCPQuantity	UpdateComponentData	UpdateProductData	NewDocumentFromModel	
UpdateLinkDPQuantity	NewComponentVersion	NewProductVersion	EditDocumentAttachement	
UpdateLinkDCQuantity	NewComponentRevision	ChangeProductStatus	CompleteDocumentData	
BrowsingUpDocument	DisplayComponentEditor	DisplayProductAtachment	ExportDocumentToModel	
CompareProductStructure	DisplayComponentHistoric	DisplayProductEditor	SearchDocument	
CompareComponentStructure	DeleteComponent	DisplayProductHistoric	PrintAttachment	
CopyProductStructure	GetComponentVersion	DeleteProduct	PrintAttachmentToPDF	
CopyComponentstructure	LockComponent	GetProductVersion	ExportDocumentToModel	
GetComponentTopParent	UnlockComponent	LockProduct	DisplayDocumentEditor	
		UnlockProduct	DeleteDocument	
			GetDocumentVersion	

Fig. 3. Expert from Functional PLM service catalog

5 Feasibility of the Proposed Approach

We propose a BP Orchestration meta-model. Figure 4 below presents this meta-model in term of class and relationships between classes. The main concepts of this meta-model are the Business Orchestration, Business Operation, Control Flow, Business Object, Business Actor and Business service. A business process performs business operations, which are associated to business services. A business operation is performed by a business actor and consumes and/or produces business objects. Moreover, it can have pre-conditions or post conditions. The main Control Flows in our meta-model are Sequence, Fork and Join [12]. Modeling PDP as service orchestration is simpler than modeling it using other process modeling approaches that are often ad-hoc and based on complex meta-models.

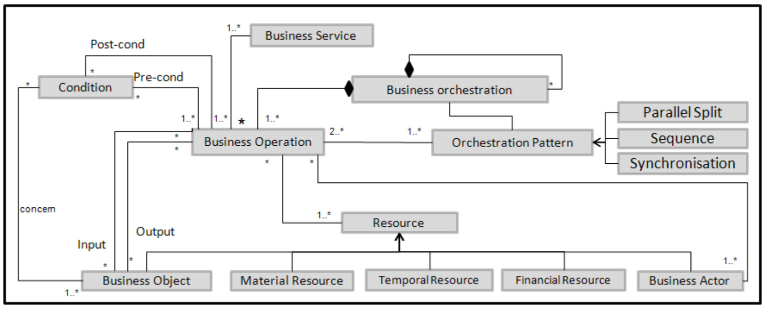


Fig. 4. BP orchestration Meta-model

In order to illustrate the BP orchestration meta-model instantiation, we propose to use the following example. The example describes an Engineering Change Request (ECR). It includes two steps: the elaboration and the validation of the ECR. We modeled this process based on our BP orchestration meta-model and using our PDS (fig. 5-a) to show that we can define all PDP using our PDS catalog and that it's easy to do it using our orchestration meta-model. We used the following PDS operations list; Elaborate Engineering Change Request, Distribute Engineering Change Request and Validate Engineering Change Request. We then mapped it with functional services operations (fig. 5-b) in order to test the articulation between the service levels.

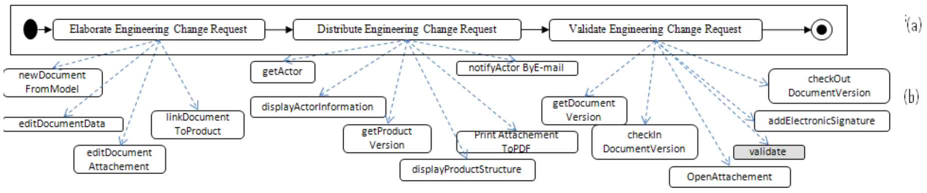


Fig. 5. Engineering Change Request Process

Capturing a business process in a model like this makes it easy for everyone looking at the process model to get a basic understanding of the business process. Each PDS operation is defined by a set of functional PLM services operations that can be executed in turn by invocation of technical PLM services operations. Dotted arrows represent the articulation from one service level to another one. This testing upholds that all business process can be expressed in terms of our identified business services. Moreover it helped us to refine our business service catalog and to ensure that the functional services can meet all PDS.

6 Discussion and Conclusion

In this paper we discussed the problem of PDPs flexibility in PLM system. PDPs are emergent and product design actors cannot deal with existing technical solutions. Existing modeling approaches dealing with BP flexibility were discussed and analyzed. The assumption made is that flexible PDPs require specific methods for their design. Our contributions respond to the limitations and problems described above by providing a methodological approach that aims to provide PDP flexibility by adhering to service orientation. A service-based approach was introduced to address dynamic PDP changes. This approach presents the PDP model as a PDS orchestration. In this case the process refers to business behavior in which all steps are PDS operations. Thus, a PDS can be invoked to perform a given step of a PDP. The challenge here is to react quickly to changes either by replacing some services by other ones or by adding new services to the orchestration. In order to deal with alignment issues between technical and business level, we first proposed a service type for each level. Then, we proposed an orchestration model in each level (business, functional and technical) and a mean that allows transforming the business orchestration model to a functional orchestration model by adhering to MDE techniques. Finally, according to the used PLM system, a mapping between the functional services of the functional orchestration model and the technical services of the PLM system should be done using the same deployment techniques. In this paper we presented the business and functional services catalog. We also defined the meta-model needed to orchestrate PDS.

Techniques that allow moving from one level to another one have not been addressed in this paper. To do so, we proposed a conceptual architecture based on a Model-Driven-Engineering approach [15]. This part of work still under development, we are defining a mapping meta-model between business and functional levels, which we will name business deployment meta-model. According to the business deployment meta-model and the business orchestration model, executing a set of mappings rules can generate the functional orchestration model. As far as, we are defining a mapping meta-model between functional and technical levels that we will name functional deployment meta-model. Using the same deployment techniques, the technical orchestration model can be generated based on the functional orchestration model and the functional deployment meta-model.

The distinction between functional and technical PLM services shows the genericity of the proposed approach. In fact, an enterprise can change her PLM system or even have multiple PLM systems. The advantage of our approach is that the company can uses the same business orchestration model (and the same business deployment model), but execute many functional deployments model according to the used PLM systems. Indeed, once a process model deployed on a new PLM system we have just to execute the right functional deployment. In contrast, the limitation of our approach is that we define the deployment only on a top-down manner and we consider that all functional PLM services operations have a corresponding list of technical PLM service operations.

Acknowledgements. The authors would like to thanks the Region Rhône-Alpes for financial support of this research work.

References

[1] Credle, R., et al.: SOA Approach to entreprise integration for product lifecycle management. In: IBM International Technical Support Organization, pp. 66–80 (2008)
[2] Bessai, K., et al.: Context Aware Business Process Evaluation and Redesign. In: Int. Workshop on Business Process Management, Design and Support, at Int. Conference on Advanced Information Systems, Montpellier, France, pp. 407–414 (2008)
[3] Zhao, X., Liu, C.: Version Management in the Business Process Change Context. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 198–213. Springer, Heidelberg (2007)
[4] Lezoche, M., Missikof, M., Tinini, L.: Business Process Evolution: a rule-based Approach. In: Int. Workshop on Business Process Management, Design and Support, at Int. Conference on Advanced Information Systems, Montpelier, France, pp. 407–414 (2008)
[5] Boukadi, K., et al.: CWSC4EC: How to employ context, web service, and community in enterprise collaboration. In: The 8th Int. Conference on New Technologies of Distributed Systems, Lyon, France (2008)
[6] Erl, T.: Service-Oriented Architecture (SOA): Concepts, Technology and Design. Prentice Hall, PTRUpper Saddle River (2005)
[7] Manouvrier, B., Ménard, L.: Intégration applicative EAI, B2B, BPM et SOA. Hermès Science - Lavoisier (2007)
[8] Hachani, S., et al.: Support of Business Processes Flexibility in PLM Systems Using a Services-Based Approach. In: Int. Conference on Industrial Engineering and Systems Management (IESM 2011), Metz, France (2011)
[9] Papazoglou, M.P., Heuvel, W.-J.: Service-oriented design and development methodology. International Journal of Web Engineering and Technology (IJWET) 2, 412–442 (2006)
[10] Chang, S.H., Kim, S.D.: A Service-Oriented Analysis and Design Approach to Developing Adaptable Services. In: The IEEE Int. Conference on Services Computing (SCC 2007), pp. 204–211 (2007)
[11] Arsanjani, A., Allam, A.: Service oriented modeling and architecture for Realization of an SOA. In: The Proceeding of IEEE International Conference on Service Computing (SCC 2006), p. 521. IEEE, Chicago (2006)
[12] Manoloscu, D.A.: Micro-workflow: A Workflow Architecture Supporting Compositional Object-Oriented Development. Phd Thesis. University of Illinois (2001)
[13] Scaravetti, D., et al.: Structuring of embodiment design problem based on the product lifecycle. International Journal of Product Developement, Special Issue on PLM (2004)
[14] Pahl, G., Beitz, W.: Engineering Design; A Systematic Approach. Springer, London (2007)
[15] Schmidt, D.C.: Guest Editor's Introduction: Model-Driven Engineering. Computer Journal 39(2), 25–31 (2006)