Data Flow Part of MDE Software Processes: a position paper

Ibrahima FALL*, Xavier BLANC
Laboratoire d'Informatique de Paris 6
{Ibrahima.Fall, Xavier.Blanc}@lip6.fr
Moussa LO
Laboratoire d'Analyse Numérique et d'Informatique
Université Gaston Berger - B.P: 234 - Saint-Louis, SENEGAL
lom@ugb.sn

Abstract

At this time no definitive approach for process enactment exists since the execution context changes become more and more important and include a high number of heterogeneous involved performers (tools and developers), and a large-scaled distribution. Also, the software process engineering community is hihly influenced by the Model Driven Engineering (MDE) vision. Such a situation leads to new important data flow management needs for process enactment. We try to hihlight such new requirements and give orientations we think can help to meet them.

Key words.
Software Process Modelling, Software Process Enactment, MDE, Software Process Data Flow, Large-Scaled Distribution.

1 Introduction

Two decades ago, software process modelling and execution have been identified as a very challenging topic of the software engineering domain [Ost87]. Even if it is commonly accepted that a software process is defined by a set of steps organizing the works done by developers for building software deliverables, there are still ongoing research works that aim at providing both a software process language and a process enactment engine. The main current issue addressed by those works is to reach a tradeoff between expressiveness and understandability of the software process language and executability of the software process.
In this article our objective is to point out that an important part of current research works on this topic put a too strong focus on the control flow part of the software processes and underconsider the data flow part. Indeed, most of the current approaches consider that the data managed by any software process is only composed of individual software artefact that are inputs and outputs of steps. Hence, software process languages define very few concepts for specifying software artefacts and their relationships, and process enactment engines only provide few support for the management of artefacts, mainly reduced to a very basic access management.
With the advent of MDE, we argue however that the data control part of a software process is as much as important as the control flow part. Indeed, in MDE software processes, all artefacts and deliverable are considered to be models with complex relationships. During the execution of the software process, developers participate to the construction of some subpart of these models. Hence, a special interest has to be taken into account in order to manage the developers accesses to the models depending on the role they play and the state of software process execution. Moreover, as the development context is becoming

*Also Laboratoire d'Analyse Numérique et d'Informatique, Saint-Louis, Senegal

more and more distributed and the size of the models is increasing more and more drastically, specialized technics have to be used in order to make the data flow management scalable.
In section 2 we present the identified problem related to data flow management for a software process enactment in the large-scaled dimension and will try its illustration by a simple case study we developed in fact. In section 3 we make a stop on some works that have been done on the domain of software process modelling and execution as we aim to precisely address it from a new context. Section 4 is devoted to the presentation of the principal directions we hope our future work is to take.

2 Data flow needs for software process management

In [WD07], Wicks has shown the growing interest of the reaserch community for the software process domain. It is important to note that there is mainly one shared goal that consists in finding a framework for a full specification and execution of any software process. However, even if the found approaches differ from their basic ideologies or from the technologies they are based on, all of them put a strong focus on the control flow part of the software process and underconsider the data flow one.
In this section, we highlight the new requirements of data flow part management in MDE software processes. Such needs are classified in two categories: one dealing with the software process language and the other dealing with software process enactment. A simple case study is used to illustrate them.

2.1 Software process language point of view

The goal of any software process is to organize the work that have to be done by developers for building a system. Activity, step and phase are classical concepts that are used to decomposed this work in several units. Work product, artefact and deliverable are classical concepts that are used to represent input and output data consumed and produced by those several units of work. A software process language provides rigourous definitions of such concepts and defines the structure of models that specifies particular software processes.
With the advent of MDE, all data that are manipulated within a process are considered to be models. Hence, all work products, artefacts and deliverables are models and compose the so called global system specification that is also a model. In such a context, there is a need then for a software process language to provide concepts for specifying the relationships between work product, artefact, deliverable and models. In particular, concepts should be provided in order to specify what are the models that are inputs and outputs of work units (activity, step and phase). Concepts should be proposed in order to define their types (i.e. metamodels), etc.
Considering that all data (work products, artefacts and deliverables) are models and that the global system specification is also a model, there is a need for concepts defining consistency rules between those models. Indeed, all those models are specifications of some subparts of the system. Hence, the modification of one model can impact several other models. The software process language should propose concepts in order to include consistency rules definition in the software process definition.
Software process languages should also provide concepts in order to specify the different states a model can have during the process enactment. For example, a model can be 'in modification' or 'finalized'. It should be noted that every model is composed of a set of model elements and it is often the case that some model elements are shared between several models. Moreover, some models can be extended during iteration phases of the process and hence some of their model elements may have to be modified. All those information concerning the definition of model states have to be included in the software process definition.
Finally, software process languages should provide concepts in order to specify access rights that have developers regarding models and their model elements. Indeed, such information have to be included within the software process definition and is mandatory for the software process enactment.

2.2 Software process enactment point of view

In a wide project, developers can be organized into development teams and largely distributed around the world. The developers at any time during the process enactment require single artifacts or whole deliverable work products to carry out the tasks they are responsible for. Also in actual industrial processes, a deliverable work product is a composition of a large number of artifacts, this make it possible to have data elements with a very large size. Hence, there is a need for storage and exchange mechanisms. During the process execution, the constituent artifacts are sometimes modified (this can be caused by some requirements or design changes for instance). Any impact of a developer's task on an artifact (creation, modification, destruction, etc.) may directly or indirectly impact a range of the other work products. This situation makes the management of the consistency of the involved work products during all the process execution and their access policy be actual problems.
An other problem is related to the heterogeneous nature of the deliverable work products. In fact, a deliverable work product can be seen as a package and is composed by artifacts that are not only UML models but are of different nature such as text documents, spreadsheets, non standardized models, and so on. This makes the data integration very difficult since an intermediate layer is necessary between all the tools used during the process execution.
The needs listed above are linked to the process control evolution and should be satisfied during the whole process execution. We notice that the changeability of the process model itself during its enactment make them more accute than ever. As a synthesis, we can say that execution engines should exist for fully specified process models with regard to the previously described language view of the data flow new requirements.

2.3 The case study

2.3.1 A brief description of the case study

The case study corresponds to the definition and the realization of a management system for a firm's employees' training requests. We have defined and modeled the corresponding development process with the Eclipse Process Framework Composer (EPF Composer) [Hau07]. The development process model is considered as simple as possible and comprises the activities of *requirement definition*, *analysis*, *design*, and *coding*. We actually have executed the process using two approaches: a messages passing based approach and a centralized approach.
In the messages passing approach, the process execution is entirely decentralized and is essentially based on messages passing between concerned developers. In this solution, each developer stores his work products in a local repository. A web application is used to represent the process engine. With the aid of this application one can identify his roles, tasks, associated work products for each task (mainly the inputs), and the other developer(s) to whom address a message for any work product need.
The centralized approach is based on a unique CVS repository for the storage of all the different work products of the process, and a Web application is used to represent the process controller. The controller and the data storage server are administered by the process manager who is responsible for receiving and satisfying all the data requests coming from the developers who use the controller for the identification of their roles, tasks, and associated work products, and for their requests expression.

2.3.2 The data flow needs illustration

As an illustration of the previously specified needs, let us consider the*Analysis* activity (represented below in ***figure 1***). The activity is composed by two tasks (*analyse preliminaire* and *analyse statique*) and is performed by the *analyste* role, it takes the *specification document* (a deliverable work product) as input. The outputs of the activity are an *iteration plan* (an artifact), the *specification diagrams*, and the *staticAnalysis diagrams* (both deliverable work products). The *specification document* is a set of text documents with screen shots of diverse nature, and of other spreadsheets; this gives an heterogeneous nature to its content which is then impossible to automatically perform. The same problem concerns the

output work products, and then the next activity in the process model (namely the *Design*). In fact the *itrations plan* is a text document, the *specification diagrams* is a UML2.0 model composed by use case and sequence diagrams, as the *staticAnalysis diagrams* contains a package diagram, two class diagrams and a text document for specific analysis elements.

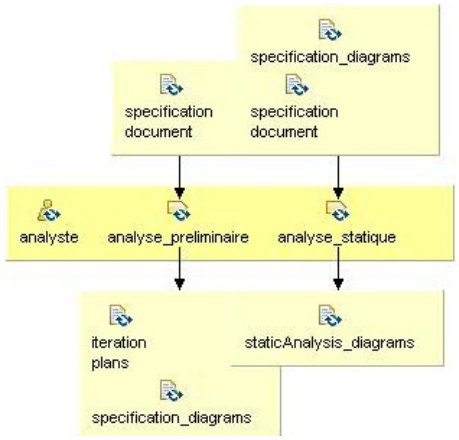

Figure 1: An EPF Composer representation of the *Analysis* activity of the case study

The next section exposes some related works in the domain of software process modelling and enactment and precises the identified problems and the necessity of a novel approach for process programs building.

3 Related works

In the early 80s [Ost87] stated that "software processes are software too" and introduced the concept of software process programming for process description. The author defined the bases that largely influenced the further works on the software process domain. These bases included works on software process description, process programming language studies, and software environment architecture, and represent an important step towards process execution automation.

Ambriola et al [ACM90] present the Oikos software process programming environment. Oikos is one of the first environments that take into account the main requirements in software process execution: a multi-user distributed nature, an open-endedness nature, and the ability to take into account the possible changes of enacted processes during their execution. Oikos is based on its own vision of the process and environment concepts which then are mutually and recursively defined. The process is seen as a hierarchy of services and some standard services are defined. These services provide the basic facilities for any environment and remain highly customizable to permit Oikos for instance to support process changeability. Nevertheless, even if the Extended Shared Prolog (ESP) is used for the agents' activities coordination, this control doesn't cover the large-scaled distribution context. In Oikos, data is stored by the DBS (Data Base System) wich then is interrogated by a Workspace Service (WS). The WS supports a data access management policy, but completely bypasses the nature of the data it manages; this unfortunately makes its approach not model-centered and then unconform to the new process enactment systems' requirements.
Estublier et al [BEM93] state on the importance of the many related works for Process-centred Software Engineering Environments (PSEEs) improvement and present Adele, a framework for formalizing and enforcing software processes. They also present the Adele-Tempo formalism for process definition and enactment in Adele. Build to overcome some Adele limitations (essentially related to the triggers-based approach it uses), Tempo is based on the concept of *role type* and reuses some many object oriented

techniques but doesn't address the large distribution of processes. Also, the Adele Data Model is composed by objects with relations between them (as in the entity/relation model) but without reference to the nature of the data. The same research team further present the Mélusine environment for services modeling and coordination [SE05]. Using a MDE vision (but not in the whole approach unfortunately) for process programs building, Mélusine represents an important reference for the software process control. Nevertheless the paper doesn't anywhere address the data flow management problem.
Through the pesentation of the APEL formalism in [DEA98], Dami et al. clearly consider the data flow separately from the control flow of a process being modeled. The used mechanisms for an activity's data management are defined in associated *desktops*. These mechanisms include products states and access policies, but completly ignore the deliverable concept and the nowadays requirements of model-centered processes.
In [CG99] the proposed PROSYT Process Support System (PSS) is based on newer technologies such as mobile code and events for the support of distributed and dynamic processes. But even if the PROSYT system supports any business process in general, it does not take into account the new requirements in the particular software process engineering domain, particulary for the data management. In fact, PROSYT uses the concepts of *artifact type*, *repository type*, and *folder type*, but does not include specific mechanisms for MDE work products (models) management.
In [CLM+99] is described the Juliette engine, a process execution system for the Little-JIL[CLS+00] coordination language. The authors point to references in the domain (from STATEMATE and IDEF0 to then more recent engines as APEL and Endeavors) and demonstrate their non adequateness for nowadays processes enactment. Juliette is based on a modularized environment and a physically distributed process control system which keeps the performers locations topology and the control units topology congruent. As Mélusine, Juliette is based on an architecture that represents a very important reference for distributed software process enactment. Nevertheless, its robustness, agents' mobility support, and scalability need to be improved, also Juliette defines an execution framework for the Little-JIL agents coordination language that essentially addresses the control flow and omits the process data flow management support which then is only basically represented.
Pierluigi et al [ALG+04] present the GENESIS PSS. Here the vision of cooperation and collaboration covers all software engineering methods and tools and supports cooperative teamwork flexibly and effectively. The GENESIS Artifacts Management System (AMS) introduces an artifact access policy, it also associates an XML file to each artifact to represent its metadatas, and uses DTDs for the determination of the artifacts types. These issues are very important for the data management and their adaptation to the MDE model-centered vision would be very useful.
In addition, the Software Configuration Management (SCM) community is the active domain that studies products management for evoluating complex systems. SCM works are diverse and include [CW98], [Est00], [ELC+02], and have led to a uniform version model developed in [CW02]. These works propose a schema for data evolution management which is based on seperated definitions of both a structured product space and a version space followed by the definition of interplaying rules beween the two spaces. Even if these identified works all consider data as documents (files, directories, programs, etc.) with basic relationships, we believe that the SCM community has given the guidelines needed for data management in enacting MDE-oriented processes.
The software process engineering domain is very active. With regard to the presented related work, accomplished efforts must be completed by adapted data flow management for a full support of MDE-oriented and large-scaled distributed processes. Such a situation considerably determine the principal guidelines of our future work, as presented in the next section.

4 Our future directions

In this section, we present the approach we hope will lead us to a full support of MDE software processes. We still are working on the definition of a MDE-oriented approach for process data management in the new context. Such an approach includes the definition of a metamodel of process work products in which

we intend to capture all the data-related aspects. Then the main objectives of such a metamodel include:

- the possibility of a fine-grained description of any process product kind (artefact, deliverable, etc.),
- the possibility of an expression of the relationships beween process products.

We believe that this is the way that will lead us to the definition of a MDE-adapted product model. The metamodel will be used to extend UML4SPM [RMX05] and give it a MDE support. UML4SPM is a UML-based executable software process modelling language that higly profits of the SPEM [OMG07] standard experience and reuses UML2.0 activity constructs, but doesn't support the addressed data flow needs.
The above product model, will be competed by an adapted process engine. We are looking foward to refine the SCM community results to design a solution that takes benefits from our metamodel.
XMI [OMG03] is the standard created by the Object Management Group for metadata interchange, it was designed to describe MOF-compliant models using XML documents and can be used for models of any other type (metamodel). Process products in MDE are essentially models structured by the metamodel we above refer to, it is therefore natural to represent them (and their relationships) as XMI files. The XMI format is understood by all MDE-compliant modelling tools, its use for products representation is then a convenience according to the data integration issue [WD07].
Then the metamodel is to structure process data in the modelling space. It is to be completed by the management of the XMI files in the enactment space. The process engine that will have this in charge will be based on SCM works. In fact, by representing software process work products and their relationships both using XMI files, we intend to adapt the product model structured by the metamodel to the one defined in the Adele configuration manager ([EC95]). The suitness of such a product space according to the new described context has to be demonstrated by our future works. The definition of the adapted version space to be associated to such a product space will complete our current works we believe will bring us to a whole process-integrated version model for the support of data management for enacting processes of the MDE context.

5 Conclusion

Software process modelling and enactment is a concern that draws the attention of the software engineering community since more than two decades. With the advent of the MDE, new requirements occur in the domain. As considered after our analysis of the related work and illustrated through the case study, the data flow management with regard to the control flow evolution is a challenge to take up for a full process enactment support, including the large-scaled distribution. In this position paper, we have tried to highlight the challenge and therefore have commented lots of related works in the domain. We finally propose a MDE oriented approach to offer the possibility of highly generic solutions for the new identified requirements of the software process community. Our future work will be dedicated to the exploration of the issues raised by such a proposition.

References

[ACM90] V. Ambriola, P. Cincarini, and C. Montangero. Software process enactment in oikos. *Proceedings of the fourth ACM SIGSOFT symposium on Software development environments, Irvine, California, United States*, pages 183 – 192, 1990.

[ALG+04] Lerina Aversano, Andrea De Lucia, Matteo Gaeta, Pierluigi Ritrovato, Silvio Stefanucci, and Maria Luisa Villani. Managing coordination and cooperation in distributed software processes: the genesis environment. *Software Process: Improvement and Practice*, 9(4), 2004.

[BEM93] Noureddine Belkhatir, Jacky Estublier, and Walcelio Melo. The adele-tempo experience: an environment to support process modeling and enaction. *In Proc. of the Second Int'l Conf. on the SoftwareProcess*, pages 75–83, 1993.

[CG99] Gianpaolo Cugola and Carlo Ghezzi. Design and implementation of prosyt: A distributed process support system. *Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1999.

[CLM+99] A. G. Cass, B. S Lerner, E. K. McCall, L. J. Osterweil, and A. Wise. Logically central, physically distributed control in a process runtime environment. *University of Massachusetts, Amherst Technical Report UM-CS-1999-065*, 1999.

[CLS+00] Aaron G. Cass, Barbara Staudt Lerner, Stanley M. Sutton, Jr., Eric K. McCall, Alexander Wise, and Leon J. Osterweil. Little-jil/juliette: a process definition language and interpreter. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 754–757, New York, NY, USA, 2000. ACM.

[CW98] Reidar Conradi and Bernhard Westfechtel. Version models for software configuration management. *ACM Comput. Surv.*, 30(2):232–282, 1998.

[CW02] Reidar Conradi and Bernhard Westfechtel. Towards a uniform version model for software configuration management. 1235/1997(5):1–17, 2002.

[DEA98] S. Dami, J. Estublier, and M. Amiour. Apel: A graphical yet executable formalism for process modeling. *Automated Software Engineering*, 5(1):61–96, 1998.

[EC95] Jacky Estublier and Rubby Casallas. The adele configuration manager. pages 99–133, 1995.

[ELC+02] Jacky Estublier, David Leblang, Geoff Clemm, Reidar Conradi, Walter Tichy, André van der Hoek, and Darcy Wiborg-Weber. Impact of the research community on the field of software configuration management: summary of an impact project report. *SIGSOFT Softw. Eng. Notes*, 27(5):31–39, 2002.

[Est00] Jacky Estublier. Software configuration management: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 279–289, New York, NY, USA, 2000. ACM.

[Hau07] Peter Haumer. Eclipse process framework composer - part 1: Key concepts, second revision, 2007.

[OMG03] Inc. Object Management Group, 2003.

[OMG07] Inc. Object Management Group. Software process engineering metamodel (spem) 2.0. specification proposal with change bars, 2007.

[Ost87] L. Osterweil. Software processes are software too. *Proceedings of the 9th international conference on Software Engineering, Monterey, California, United States*, 1987.

[RMX05] Bendraou R., Gervais M.P., and Blanc X. Uml4spm : A uml2.0-based metamodel for software process modelling. *MoDELS*, 3713 of LNCS:17–38, 2005.

[SE05] Sonia Sanlaville and Jacky Estublier. Mélusine. un environnement de modélisation et de coordination de services. *Ingénierie des Systèmes d'Information*, 10(3):29–48, 2005.

[WD07] M.N. Wicks and R.G. Dewar. A new research agenda for tool integration,journal of systems and software. *Journal of Systems and Software*, 80, 2007.