From Software Process to Workflow Process: the Workflow Lifecycle

José H. Canós, Mª Carmen Penadés and José Á. Carsí

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
Camí de Vera s/n E-46071 València (Spain)
{jhcanos|mpenades|pcarsi}@dsic.upv.es

Abstract
Despite the great effort devoted to the development of Workflow models and Workflow Management Systems during the last decade, a strong foundation about workflow development is still to come. Assuming that a workflow is a complex software product, in this paper we argue that the principles and techniques of software development —in particular, methodological concerns— can help in the development of high quality workflows. A unified view of the Workflow development process —the Workflow Lifecycle—is presented. Finally, we mention some topics coming from the Software Engineering field that may play a key role in the improvement of the Workflow Process.

Keywords: Workflow Management Systems; Workflow Process; Software Engineering; Workflow Lifecycle.

1. Introduction and motivation

Workflow Management Systems (WFMSs) are complex, large software systems that allow users to define, manage and execute workflows (WF). They provide support in three main functional areas, namely build-time, run-time control and run-time interactions [1]. Build-time features involve WF definition according to a WF meta-model including notions such as process, activity, task, actor, etc. Run-time control and interaction functionality consists of generating an executable version of the WF and supporting the execution of WF instances, including communication with external users and invoked applications as well as access to the process data — usually stored in an external database management system (DBMS). In addition, some WFMSs provide WF prototyping and analysis of WF executions [2], the former to check WF specification and Business Process (BP) model semantics, and the latter to support WF debugging and BP reengineering (BPR).
Most of the work done so far in the WF field has been tool-oriented and technological in nature; the main goal has been during years the definition and development of WFMS including models, modeling languages, execution environments, etc.; this effort has reached a reasonable success, with a number of both research prototypes and commercial systems currently available [3]. However, from our point of view, a strong foundation about WF development is still to come.
As pointed out in [4], methodological issues have received little attention, and WF developers often have to face the problem of WF development with neither a methodological support nor a global view of the process. Quoting Osterweil's words, *"(...) the importance of orderly and systematic processes as the basis for assuring the quality of products and improving productivity in developing them (...) I was starting to see the creation of a software process*

universe parallel to the universe of notions and tools surrounding application software development. The more I looked, the more similarities I saw. (...) Thus it seemed important to suggest that software process technology might not need to be invented from scratch (or reinvented), but that much of it might be borrowed from application software technology." ([5], page 1).

These statements, originally referring to Software Processes (SP), can be applied to WF as well. Software Engineering (SE) research has been during years looking for methods, procedures and tools to develop quality software products; as a matter of fact, many topics have emerged which have become research areas themselves (analysis and design methodologies, project management, quality assurance, evolution, etc.). But wrapping all them a key concept governs all the SE practice: the software lifecycle (or, in a broader sense, the SP) understood as the systematic and time ordered application of different SE techniques.

Assuming that a WF is a complex software product, in this position paper we introduce the WF Lifecycle (section 2) as the seed of an interesting research line that might be called "the WF Process" (WFP) by analogy with the notion of SP introduced years ago. The study of the WFP will undoubtedly lead to borrow methods, procedures and even tools from SE in order to build correct, complete and efficient WF; some of them are outlined in section 3.

2. The Workflow Lifecycle

A WF is a complex software product; during its development, all the activities mentioned above have to be performed in a given order. The *Workflow Lifecycle* (WFLC) defines both the way in which the different activities involved in the development and use of WF are ordered in time (the control or process flow), and the input and output of each activity (the information flow). It comprises three main processes:

1. *Construction and stabilization* (steps 2 and 3 of figure 1): starting from a business model (BM) produced at the BP Analysis phase[1], a (graphical) language is used to model the workflow using concepts such as process, activity, task, etc.; as it occurs with any software product, the WF construction process is not error free, so it is desirable to have means to detect errors prior to any further step; an iterative prototyping process permits to simulate WF executions in order to detect and correct specification mistakes, making the WF as close as possible to the BP model.

2. *Debugging* (steps 4, 5, 6 and 7 of figure 1): once a stable version of the BP is available as a WF specification, a full-featured, operational version is generated by mapping modeling concepts into an executable representation. As a result of the execution or enactment of the WF a log is created (Exec log) which collects information about the process execution (e.g., tasks executed, starting and ending of tasks execution, resources used, etc.) that can be analyzed a posteriori (WF Analysis) in order to detect anomalous situations (deadlocks, bottlenecks, etc.) derived from a bad WF design; once an anomaly is detected, a revision of the WF model must be performed, i.e. a new stabilization process must be carried-out. Note that

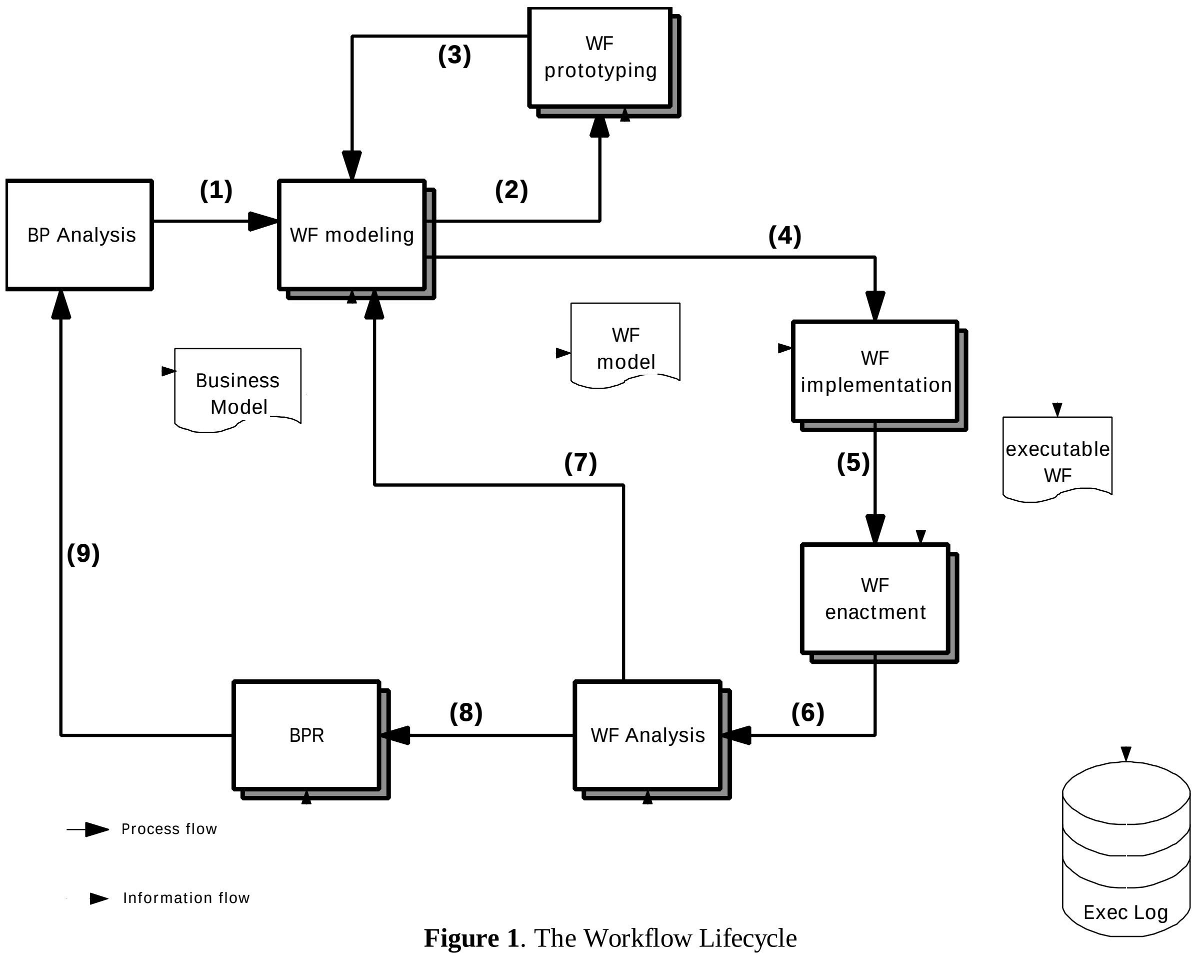

Figure 1. The Workflow Lifecycle

the BP model remains fixed during all this process: the purpose of the debugging process is to improve the quality of the WF design only.

3. *Business process reengineering*: apart from WF-related data, the Exec log may also contain business-related data recorded during the WF enactment phase. Those data are used in order to check the adequacy of the BP model to the goals of the organization. If this is not the case, a revision of the BP model is mandatory. The BPR phase leads to the BP analysis, where the BP model is revised. The updated model is the starting point of a new iteration of the processes of stabilization and debugging. This process corresponds to steps 8, 9 and 1 in figure 1.

[1] the separation between BP analysis and WF modeling is deliberate: the business model may already exist at the time a company decides to use a WFMS; however, if it is not the case, the BP Analysis and WF modeling phases could be drawn as a single one

This WFLC is inspired by the prototyping-based software lifecycle approach [6]. It represents an iterative process in which stages different methods and tools may be used. In the following section we enumerate some of them which can be taken from SE current practice.

3. The Software Engineering contributions to the Workflow Lifecycle

As mentioned earlier, SE methods and tools can play a key role in the improvement of the WFP. From our experience in the last decade, we suggest now a number of SE topics whose application to WF development can result in a better WFP.

Formal specification techniques. The use or formal specification languages in the early phases of the lifecycle allows rapid prototyping of requirements specifications and the early detection of analysis errors [7]; the operational semantics of such languages provide executablity to the specifications, having WF prototypes at low cost. Graphical WF specifications may be automatically translated to equivalent formal specifications which are animated in the prototyping phase of the WFLC, provided that the underlying formal model has the expressiveness needed to support WF concepts.

Object-Oriented technology. The adoption of object-oriented (OO) technology in all the phases of the software lifecycle has proven to be very successful in the development of conventional software as developers handle a unique model during all the lifecycle. This reduces impedance mismatches between the different models used in the phases of modeling and enactment, and in the external DBMS. Moreover, the know-how acquired through the use of OO requirement analysis techniques (scenarios, use cases, etc.) may help BP analysts in the WF requirements elicitation process.

Automatic code generation. A number of tools providing automatic code generation from requirements specifications are currently available. For instance, in OO-Method Case [8], a formal OO specification in OASIS is generated from a set of graphical models, and later an automatic process generates executable applications in C++, Java and other programming languages; persistence is provided by a relational DBMS. A formal, OO WF modeling language would enable the automatic generation of enactable WF models written in programming languages. Then, standard static and dynamic program analysis techniques [9] could be used to improve the quality of the models.

Metalevel constructs and evolution. Many efforts have been devoted to cope with the problem of evolution in the fields of SE and the SP. Among them, those based in the dynamic evolution of meta-models [10,11,12] are of particular interest, due to the fact that a similar approach may be taken to define an evolvable WF meta-model whose dynamic properties must include support to both structural and behavioral consistency.

4. Conclusions and future work

In this paper we have pointed out that Software Engineering and Software Process concepts can be applied to WF development in order to build better WF products. In particular, a unified view of the activities related with WF management has been introduced. The definition of the WF lifecycle as the composition of three main processes (namely construction and stabilization, debugging and business process reengineering) subsuming the well known tasks of WF modeling and enactment plus prototyping and analysis is the first step in the development of new frameworks for WF development.

Future work include the definition of a homogeneous model to be used along the whole WFLC in order to improve the traceability of WF specifications and reduce impedance mismatches. Moreover, this model should be expressive enough to integrate evolution aspects in a seamless way.

5. References

[1] D. Hollingsworth. "The Workflow Reference Model", Workflow Management Coalition, TC00-1003, December, 1994.

[2] Geppert, A. and Tombros, D., *Logging and Post-Mortem Analysis of Workflow Executions based on Event Histories.* Proc. 3rd Intl. Conf. on Rules in Database Systems (RIDS), LNCS 1312, Springer Verlag, Heidelberg, Germany, pages 67-82, 1997.

[3] Alonso, G., et al., *Functionality and Limitations of Current Workflow Management Systems.* IEEE Expert 12(5): 0- (1997).

[4] Sheth, A. et al., *Report from the NSF Workshop on workflow and Process Automation in Information Systems.* Computer Science Department Technical Report, UGA-CS-TR-96-003, University of Georgia, October 1996. Available at http://lsdis.cs.uga.edu/activities/NSF-workflow/final-report.ps

[5] Osterweil, L.J., *Software Processes Are Software Too, Revisited.* Proceedings of the Nineteenth International Conference on Software Engineering (ICSE 1997), pp. 540-548, May 17-23, 1997, Boston, MA.

[6] Pressman, R., *Software Engineering: a Practitioner's Approach*,4th edition. McGraw-Hill, 1997.

[7] Balzer, R., et al., *Software Technology in the 1990's: Using a New Paradigm*, IEEE Computer, Nov. 1983, pp. 39-45.

[8] Pastor,O.;Insfran,E. ;Pelechano,V. ;García,J. *From Object Oriented Conceptual Modeling to Automated Programming in Java.* Proc. of the Intl. Conference on Conceptual Modeling-ER'98. LNCS 1507, Springer-Verlag, 1998, pp. 183-197.

[9] Marick, B., *The Craft of Software Testing*, Prentice Hall, 1995.

[10] Warboys, B. (ed.), *Meta-Process*. In Derniame, J.C. el al. (ed.), "Software Process: Principles, Methodology, and Technology", LNCS 1500, Springer-Verlag, 1998, Chapter 4.

[11] Tresh M., Scholl M., *Meta object management and its applications to database evolution*. Proc. of the 11th International Conference on the Entity-Relationship Approach, LNCS, Springer-Verlag, 1992.

[12] Carsí J.A., et al, *A DOOD System for Treating the Schema Evolution Problem*, EDBT'98 Demo Session, VI Intl. Conference on Extending Database Technology, Valencia, March 1998.