Market Forces and End-User Programming for Mission-Critical Systems

Lutz Prechelt
Institut für Informatik, Freie Universität Berlin
Takustr. 9
14195 Berlin, Germany
+49 30 838-75115
prechelt@inf.fu-berlin.de

Daniel J. Hutzel
abaXX Technology AG
Forststraße 7
70174 Stuttgart
+49 711 61416-0
daniel.hutzel@abaxx.com

ABSTRACT

The abaXX Workflow Engine (WFE) is a J2EE COTS software component, part of a larger suite for building web-based systems. Although these systems are usually mission-critical (the customers often being financial institutions), a visual tool that could be used for end-user programming, called the Process Modeler, proved important for marketing the WFE and the component suite in general. The promise of end-user programming (EUP), however, never materialized. This article sketches the evolution of the WFE. It de-scribes why the EUP capabilities were required, why they were never really used in practice, and how to reconcile these two facts.

Categories and Subject Descriptors

D.1.7 [**Visual Programming**]
D.2.9 [**Management**]: Software Quality Assurance
J.1 [**Administrative Data Processing**]: Financial
K.1 [**The Computer Industry**]: Markets
K.6.4 [**System Management**]: Quality Assurance

General Terms

Design, Economics.

Keywords

Workflow engine, workflow modeler, quality assurance, marketing requirements.

1. INTRODUCTION

abaXX.components is a suite of COTS (commercial off-the-shelf) software components for building high-end mission-critical e-business and portal solutions (often in the financial industry) based on Java™ 2 Platform Enterprise Edition (J2EE™) component and web technology. Among these components is a Workflow Engine (WFE) component [1], introduced for providing model-driven, code-free development which may be carried out by domain experts without technical staff. This report describes the evolution of the WFE component using qualitative, anecdotal evidence.

After introducing some terminology (Section 2), we will discuss four subsequent versions of the WFE (Sections 3, 4, 5, 6). The description will show

- why visual end-user programming (EUP) capabilities (a Process Modeler) were added to the WFE in version 2 (namely to be able to explain to the non-technical domain experts the advantages of having a WFE when trying to sell the components) ,
- why EUP has never actually happened (namely because financial institutions' quality assurance would not allow it),
- why this will probably not change (namely because Process Definitions involve either too much low-level technical detail or are too complex to be reliably grasped by one person alone), and
- that in this context the appropriate goal is probably not EUP, but rather end-user understanding, which may involve similar means.

For each WFE version, we will first consider the requirements from a marketing point of view, then discuss technical aspects, and finally describe the effects and consequences the version had.

2. WHAT IS A WORKFLOW?

Workflow is a buzzword that means *very* different things to different people. In this article, we are concerned with automated workflows only and adopt the following terminology (roughly along the lines of [9]):

- A *workflow* is the automation of a business process, in whole or part, according to a process definition.
- A *process execution* is the result of executing a process definition with specific parameters.
- A *workflow engine* (WFE) is what executes the process definition.
- *Process definitions* consist of activities connected into a graph. Control flow is described by the edges; data flow is realized via reading/writing shared variables.
- *Activities* are distinct processing steps, implemented as Java classes (custom or from a library).
- Process definitions can be structured by introducing subprocesses (much like subroutines).

We discriminate three kinds of processes, with increasing granularity and duration:

- Micro Flows (running milliseconds),
- Page Flows (running a few minutes), and

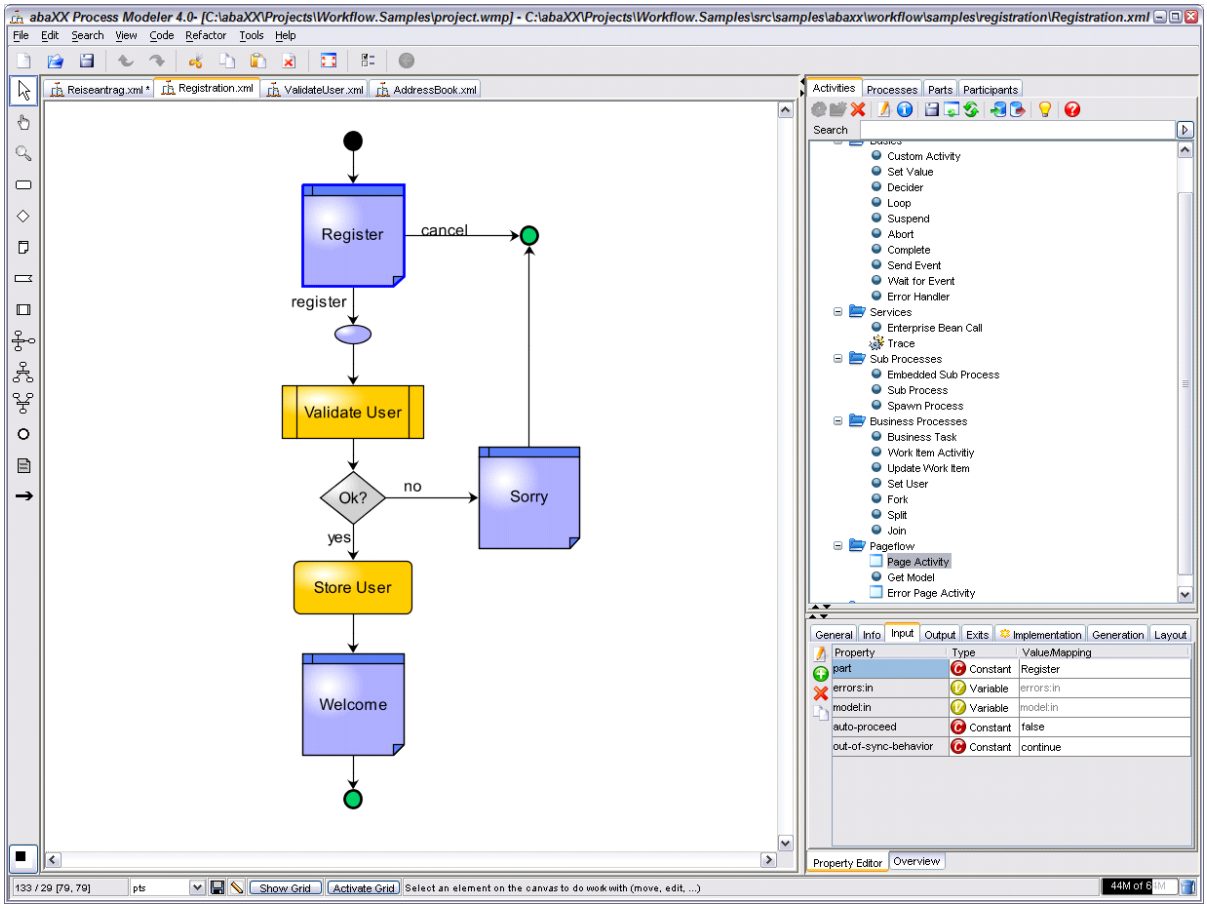

Figure 1: The Process Modeler (as of WFE 4)

- Business Processes (running hours to weeks)

These will be explained below.

3. WFE VERSION 1 (MICRO FLOW)

Marketing requirements: Flexibility is a key issue for enterprise applications, as their requirements evolve continually. Modifications often involve changes to the control logic, which almost always require modifying the source code, even when using flexibility-enhancing design patterns [4]. A requirement is thus to provide a WFE that allows for introducing changes to control logic and other program aspects without touching the source code. Changes should even be possible at runtime, as enterprise systems are often required to run 24/7. Additional requirement: Provide a mechanism by which customers could flexibly adapt procedures provided by us (or other vendors who built on top of our components) without source code access.

Technical requirements and design: As development resources are limited, initially focus on *Micro Flows*: very lightweight, short-running processes with life spans measured in milliseconds, starting and terminating within the same transaction (if any). We started from the reference model published by the WfMC [10], but used only the most fundamental notions to form a small-footprint meta model: process definitions are composed of activities, transitional control flow, and data flow through shared variables. The implementation of any Activity is provided by a Java class.

WFE 1 had no built-in support for user interaction or multiple participants, but we felt this component would quite well fulfill the most important requirement: providing flexibility.

Consequences: Marketing-wise, this solution was a partial success. It was used rather successfully by one of our customers, F, who was building a software product to be bought by banks. It implements banking services to be used via mobile devices and contains over 400 different process definitions. Using a WFE is crucial for this product: The software needs to be extensively modifiable by domain experts at customer banks, as most banks have a lot of unique requirements. However, F would not want to give its customers access to the product's source code. Using a WFE and providing source access to the process definitions only is a good and elegant solution.

However, most of our customers were building custom solutions rather than products. It turned out that most of them eventually tended not to use the WFE at all (a few had bought it, though). They felt the effort of wrapping their methods as activities and producing the process definition never paid off.

4. WFE VERSION 2 (MODELER)

Marketing requirements: For the second version, we chose to believe that a main drawback of our WFE was the effort needed to produce the process definitions textually in XML. We decided to build a graphical editor for process definitions, called the Proc-

ess Modeler, though not everybody believed this would be the right choice.

Our decision was mainly driven by market forces. We needed to sell our components (which are typically sold as a suite and bought in the context of specific projects) not only to technical people, but also to domain experts, who are almost without exception non-programmers. To them, the idea of the WFE 1 had been extremely abstract and it had been hard to convince them that it was worthwhile.

We believed the Modeler would change that: It made program designs and modifications visual, would give the domain experts a means to talk about the system to the technical people at eye-level, and would hence increase the domain experts' technical confidence and be a great selling proposition.

We also intended to sell the Modeler as an end-user programming tool: Changes to process definitions that relied only on previously existing activities could be performed by domain specialists alone, much like editors change a web site through a content management system.

Technical requirements and design: The Process Modeler was implemented as a stand-alone program, a Swing application. It allowed to graphically design process definitions in familiar activity-flow diagrams by selecting pre-defined activities from a gallery, dropping them as nodes on the diagram and wiring them into a control flow by drawing edges from declared exits of an activity to successor activities.

The existing base of the WFE (the process definition language, the framework, and the runtime system) hardly changed, except that we added meta information for Activity implementations (BeanInfo).

Consequences: Marketing-wise, WFE 2 was a success. The Modeler was indeed instrumental in convincing the domain experts of the usefulness of a WFE which so became an important selling point for our component suite.

Technically, however, it turned out that the Modeler hardly added to the practical value of the WFE:

- From the programmers' point of view: Although the usability of the Modeler was quite acceptable, most programmers found it quicker to write the XML process definitions with a text editor.
- From the domain experts' point of view: Whatever solution they were building, with the processes being Micro Flows, most of what they did still tended to be rather internal to the software and hence still quite abstract. The gains from having a visual representation were relevant in only few cases.

Even the aforementioned customer F decided *not* to use our Process Modeler, although in principle such a tool was obviously an important contribution to the value of their product. Rather, they decided – despite the rather large effort involved – to build a similar tool themselves, because they required massive customizability, which our Modeler did not provide.

Overall, it turned out our original assumptions had been mostly wrong; the actual usage conditions often crippled the usefulness of having a Micro Flow WFE:

- Dynamic behavior changes at runtime were often as unacceptable as was end-user programming: The quality assurance processes of many of our customers (many of whom are financial institutions) demanded a full-fledged IT project even for changes where modifying only process definitions could have done the job.
- Many changes were not confined to the process definition: Much of the time, the intended change would require introducing a new Activity or modifying an existing one.
- The vast majority of code in custom solutions goes into web user interfaces. Consequently, modifications in response to evolving requirements would quite frequently affect user interface logic and the flow of web pages, which was not covered by the WFE in versions 1 and 2 at all.

A WFE that supported only Micro Flows was clearly not as useful as we had hoped.

5. WFE VERSION 3 (PAGEFLOW)

Marketing requirements: At this point, however, we recognized that our WFE would become very useful if it supported the type of process we call *Page Flow*:

Page Flows describe user interactions spanning several dialog steps of the same user within a web application. They usually have a medium duration (minutes). Page Flows repeatedly do the following: execute some application logic, then display a page, then wait for the user to send the next request. Page Flows centralize the control flow of a web application, which otherwise tends to be widely scattered over many files and rather hard to understand and modify. We decided to extend the WFE to cover Page Flows.

Technical requirements and design: The Modeler was enhanced by a Page Gallery that displays the pages available in a web application. Dropping such a page to a process diagram adds a page activity to the process definition with control flow exits reflecting the page's declared web events.

Like in version 2, the required changes to the WFE were purely incremental.

First, a generic Page Activity was introduced, instances of which represent web pages within a process definition. Its implementation displays the assigned page and suspends the process. Second, the run-time system's Page Flow Interceptor plugs into the web framework [2], [6], [8], intercepts an incoming request (if it refers to a Page Flow process instance), maps it to the corresponding exit of a page activity, and resumes the process.

Consequences: By and large, this third version of the WFE was a big success, both technically and marketing-wise. Most of the expected benefits were realized:

- While defining a dialog sequence, the visualization provided by the Process Modeler much simplified the communication both among domain experts as well as between domain experts and programmers.
- This was true not just for the initial implementation of a dialog sequence, but also during incremental improvement cycles later on.
- At the same time, the centralization of control flow aided and accelerated frequent changes during the development process, which made rapid prototyping much more realistic.
- Understanding an existing implementation became much easier for the programmers: Normally in a web application, control flow and data flow are scattered horribly across dozens of files (JSP pages and controller Java classes). In

contrast, a Page Flow definition now provided a nice, coherent "big picture" while suppressing the details.

In terms of end-user programming, the addition of Page Flows allowed business experts to quickly change the course of user interactions in certain cases. Overall, however, the situation did not change much compared to that we found for WFE 2:

- Modifying (let alone creating) a Page Flow still usually required too much technical background knowledge (about the behavior of the Activities involved) to be possible for a pure domain expert
- and the prescribed quality assurance processes would not have allowed it anyways.

Nevertheless, our customers now started using the WFE heavily and not before long the obvious and unavoidable thing happened: They wanted to go beyond single-participant page flows towards long-running, multi-participant business process automation.

6. WFE VERSION 4 (BUSINESS PROCESS)

Marketing requirements: Thus, the next goal clearly had to be supporting the next level of processes, too: *Business Processes*.

Business Processes are workflows with a long duration (hours to weeks) and multiple participating users. Most of the time they pause with their state information persisted, waiting for an external event – typically a user re-activating the process by triggering its next step after selecting it as a task from a work list, but often also technical kinds of trigger stemming from other software systems. This requires functionality for modeling participant roles and support for parallelism.

Technical requirements and design: We added that support basically by introducing new generic activities. This was mainly the Work Item Activity which represents a transition from one participant to another (during which the process is suspended/passivated) as well as Split and Join Activities which allow for parallel control flow paths.

In addition, we provided pre-fabricated user interface components, such as the Worklist Portlet which displays a user's list of pending work items and allows him/her to pick an item for further processing.

On the Modeler side, we introduced a Participants Gallery, which reflects organizational roles in a hierarchy. An open XML interface allows customers to put their own organizational model into this gallery. An entry from the Participants Gallery can be dropped to a process diagram and associated with a Work Item Activity, with the effect that the work item created at runtime will be assigned to all users who have the respective organizational role.

Consequences: At the time of writing, version 4 of the WFE has only just appeared on the market, so we do not yet have much actual experience with its consequences. It is obvious, however, that the extended functionality does not make end-user programming any easier:

- the previous stumbling blocks are still relevant,
- the complexity of concurrent, multiple-participant processes is much higher (deadlock, escalation, substitution, etc).

7. LESSONS LEARNED & CONCLUSION

Past and present: As we introduced the Process Modeler, we did not make a conscious decision whether it would primarily be targeted to end-users or primarily to software engineers. We are still not sure whether the EUP claims[1] we made helped more than they hampered in convincing potential customers to buy our component suite. But we have meanwhile understood this much: as unrealistic as actual end-user programming by means of the Process Modeler may be in our setting, the Modeler's contribution towards both convincing (before buy) and empowering (after buy) the domain experts is substantial and relevant – the Modeler is *not* just some arbitrary non-end-user software engineering tool.

Future: Further development of the WFE will likely be in the field of web-service-based process collaboration and orchestration, maybe using a standardized notation such as the BPEL (Business Process Execution Language, [5]) or BPML (Business Process Modeling Language, [3]), plus the underlying somewhat extended paradigm [7]. This will further increase the complexity faced by people creating Process Definitions and will make end-user programming still less likely to occur.

This is probably a trend valid for all potential mission-critical EUP in enterprise applications: The complexity of the requirements handled by these applications increases in such a way that end-user programming becomes increasingly unlikely because multi-person teams (which can then include a software engineer) are needed anyways – independent of the actual tools or techniques used.

Conclusion: Sometimes end-user *programming* may be one half the right idea and one half a red herring: end-user *understanding* may be the goal to go for.

8. REFERENCES

[1] abaXX Technology AG, abaXX.components Workflow Engine User Guide V4.0, Stuttgart, 2004.

[2] abaXX Technology AG, abaXX.components WebApp Framework User Guide V4.0, Stuttgart, 2004.

[3] Business Process Management Initiative, *Business Process Modeling Language BPML V1.0 Specification*, www.bpmi.org, 2002.

[4] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[5] OASIS, Business Process Execution Language for Web Services Version 1.1, www.oasis-open.org, 2003.

[6] H.M.L. Ship, *Tapestry in Action*, Manning Publications, Greenwich, CT, 2004.

[7] H. Smith, P. Fingar, *Workflow is just a Pi process*, BPTrends, www.bptrends.com, January 2004.

[8] J. Turner, K. Bedell, *Struts Kick Start*, SAMS, 2002.

[9] Workflow Management Coalition, *Terminology and Glossary*, WFMC TC 1011 v.3, www.wfmc.org, Feb 1999.

[10] Workflow Management Coalition, *The Workflow Reference Model*, WFMC TC 1003 v.1.1, www.wfmc.org, Jan 1995.

[1] Better: claimlets, as they were never really loud.