MDA Journal

November 2004

David S. Frankel
David Frankel Consulting

df@DavidFrankelConsulting.com

Introduction

A number of MDA Journal authors have argued that domain-specific languages (DSLs) and frameworks that support DSLs are key technologies for the future of MDA. This month, Ken Orr gives us a different perspective, putting the focus on executable business processes, one of the original themes of MDA Journal.

Ken's approach is pragmatic, grounded in his industry practice. He also ties in agile development processes, showing how they have the potential to dovetail nicely with a business process centric approach.

In my opinion, the DSL and executable business process approaches are not mutually exclusive. Business process modeling languages and script languages are integral to the executable business process approach. Nevertheless, we who dwell in the enterprise software world should not forget that automating business processes is our real goal, and language technology is but a means to achieving that goal.

Ken drives home the point that the pieces of the executable/agile business processes puzzle are in place, and that it is possible to put them together into a cohesive whole. This is no small matter. From this plateau of achievement, we can also begin to attack a number of related challenges, such as scaling agile development to look beyond the single project to reusable assets that span multiple projects. Another important issue is how to deal with components and applications that were not built via the executable business process approach; these must be integrated into the business process machine.

There's much to cover here, and Ken gives us a place to start.

See you in December.

David Frankel

MDA Journal

November 2004

Ken Orr and
Randy Hester

Ken Orr Institute

Beyond MDA 1.0: eXecutable Business Processes From Concept to Code

In the 1980s, there was a concerted drive to build software development environments that would go from "concept to code." That approach ultimately came to be known as *Computer Aided Software Engineering* or *CASE*. The idea was that software engineering, like the rest of engineering, should allow a business user or business analysts to specify what they wanted in high-level, user-understandable concepts, "push a button" and produce a working system, or at least a prototype, much like automobile engineers or architects were beginning to do at the time with CAD/CAE/CAM tools.

The idea was appealing, but the technology wasn't there, and the first generation of CASE fell out of favor in the early 90s, having trouble keeping pace with the move from mainframes to client/server and then to the Web. More recently, however, the emergence of Model Driven Architecture (MDA) has caused a renewed interest in computer-aided business modeling and implementation.

For the most part, the proponents of MDA have been more modest in their goals than were the early proponents of CASE. Indeed, the most popular form of MDA today is one where high-level conceptual modeling gets translated, level by level, into UML constructs with only a small amount of automation at each level. However, another form of MDA, which might be called "executable UML," does look a lot like what CASE was about. It seems that people are once again beginning to think about integrated environments.

At base, however, there is still something of a chasm between those who believe that the future of business process management looks more like CAD/CAM and those who believe that it looks more like an advanced JAVA (.NET) development environment. From our standpoint, it seems that the business process community ought to push for vendors to produce something like the CAD/CAM model for business process modeling. And we think it can be done within the framework of MDA.

Approach 1: Create a more complex language and develop language specific tools

First, it is useful to note that MDA is not strictly limited to either of these current models, or even to UML itself. As Dave Frankel explains it, any formal modeling specification (language) that can be modeled by the Meta Object Facility (MOF) can form the basis of an MDA. This means that any consistent modeling approach can be a candidate for a MDA if it is sufficiently precise. In the most recent MDA Journal, for example, Evans et al., discuss the limitations of the current MDA (UML driven) approaches vs. a more sophisticated "language-driven" (language-oriented) approach.[1]

[1] Evans, et al., Language Driven Development and MDA, MDA Journal, www.bptrends.com, October 2004

Their assertion is that,

"Language driven development realizes substantial productivity gains by constructing rich development environments that automate significant parts of the development process. This includes domain specific modeling, model validation, model simulation, and transformation to code."

While the approach put forward in Evan's article offers an interesting set of concepts for people, especially tool vendors, to think about, the business process audience should not therefore conclude that the future of MDA necessarily needs more sophisticated languages to support advanced development. There are other, more direct approaches to accomplishing this same task. The authors feel that most of what business process analysts want and need in the way of a business process driven MDA environment already exists in the marketplace; what is needed is not a new language, but to combine the right parts of the right technologies into a seamless environment.[2]

Approach 2: Construct an integrated Business Process Toolset from existing components

> The true goal of a business process oriented MDA should be to be able to model, simulate, generate, and manage eXecutable Business Processes (XBPs) in real-time.

What enterprises desperately need is a modeling and development environment that allows them to model their business processes, both "as is" and "to be," and then to prototype, simulate, and ultimately generate them for production. Then, when the processes change, the user/business process analyst should be able to repeat this process in real-time rather than waiting for weeks or months.

While many developers like to believe that their technology is so complex that it is impossible to build tools that will ever generate sophisticated, real-world applications, the fact that other engineering disciplines routinely attack problems that are equally complex suggests that something like an advanced MDA or CASE toolset is a real possibility. In fact, this article is based on a working prototype of just such a tool set.

The model for this tool set, which we call *NextPractice*, is built around three off-the-shelf business process modeling/software development components and an integration component:

(1) a business process modeling tool (ProVision),
(2) a database/application design/generation tool (GeneXus)
(3) a workflow management engine (GX Flow)
(4) an interface tool that integrates the other components (NP Interface)

[2] Indeed, the idea that business process engineers should be greatly concerned with UML and the standards of the object development community seems to get the cart and the horse reversed. Ultimately, the business process analyst wants to be able to model his/her organization's business processes in terms of the things the business user can understand easily and then push a button and be able to have that process generated so that they could try it out immediately with the user to see if what they get is what they wanted—just like their counterparts in manufacturing, electronics, and architecture.

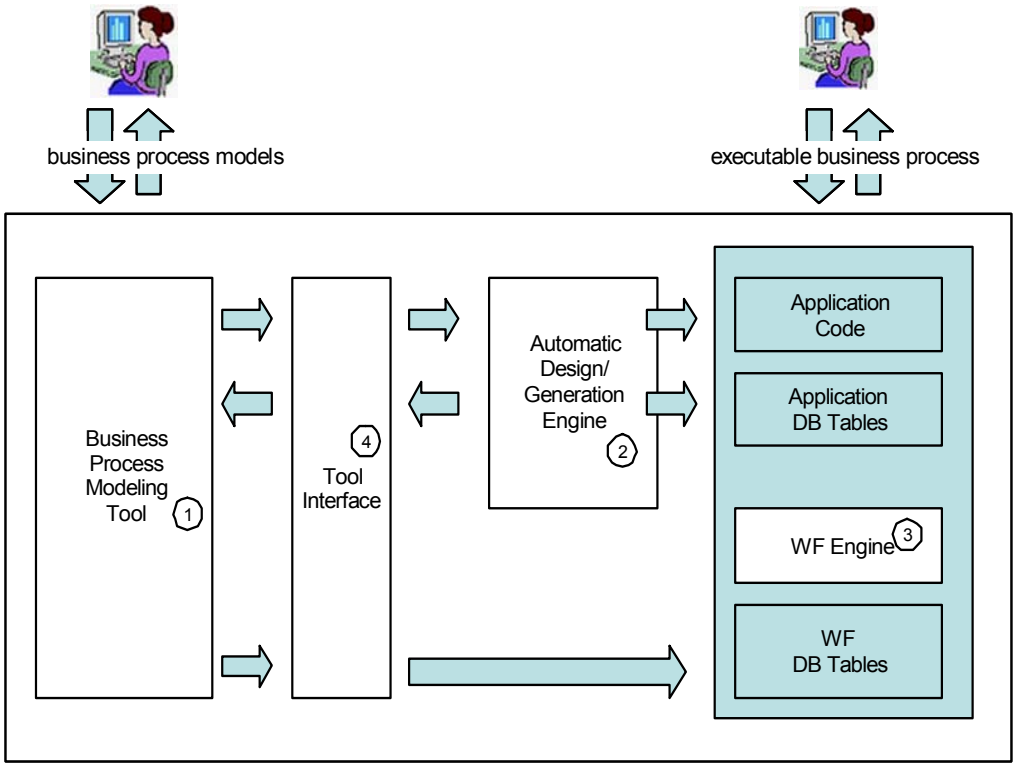

Figure 1 – A CAD-based Model of an advanced Business Process Management Toolset.

In this case, creating a sophisticated XBP environment resulted not from creating a new languages and language specific tools, but rather in integrating (1) a customizable business processing modeling tool [3]and then (2) translating the meta-data contained in these business models into code using very a very intelligent automatic database/program design/code generation tool. [4]

By building the appropriate interfaces between these tools it is currently possible for business process analysts to develop their XBPs, first developing the key concepts with the business users using a well-defined business process modeling approach, and then detailing the model. (As with any development environment, a VB-like script language exists for specifying complex business rules to the database/program design/code generation engine.)

The execution framework for NextPractice XBP applications is shown in Figure 2. In this execution framework, application code (programs that handle screens, reports, automated procedures, etc.) is kept separate from the workflow and persistence and is, ultimately, managed by the workflow engine.

[3] The business process tool allows business users and analysts to capture business models easily understood by non-professional (i.e., non-programmer) business users/analysts (e.g., context diagrams, business process/swimlane diagrams, activity descriptions,and user interface structures).

[4] The three tools that were used in this toolset are widely available. The business process-modeling tool (ProVision) is available from ProForma Corp. www.proformacorp.com, while the Automatic Design/Generation Engine (GeneXus) and Workflow Engine (GXFlow) are available from ARTech, Ltd. www.genexus.com. These component tools are being used in thousands of organizations.

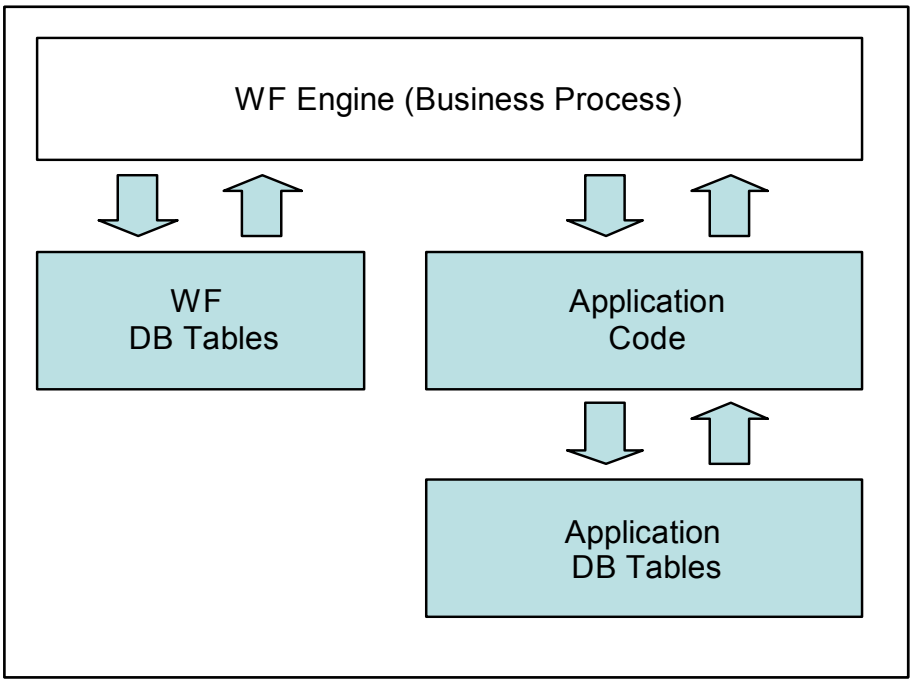

Figure 2 – XBP Executive Environment.

Building an MDA 2.0 around Business Process Modeling and XBPs

Any serious attempt to provide computer-aided tools to help business process initiatives in large organizations needs to focus not on technology but on the users. Let's postulate that it is possible to develop a more exciting MDA—let's call it MDA 2.0. A true MDA 2.0, we believe, ought to be built around a robust strategy that works from business process modeling to information system design to database/code generation (DC/DG) in a seamless fashion (remember concept to code?).

In our case, the methodology used as the framework for NextPractice parallels the popular Rummler-Brache[5] business process approach.[6] The approach uses standard business process diagrams (*business interaction (context)* and *business process diagrams* (*swimlane*) that are easily understood by users but precise enough to define XBPs.

In addition to this we have added *user interface form/data structures (UIFDSs). User Interface Form/Data Structures* are simply screen/report layouts plus the meta-data about the data structure, data attributes, and decision/calculation rules associated with a user interface to a business process. The addition makes automatic designs of complex databases and code generation possible. This allows us to leverage enormously powerful DC/DG technology that takes care of a huge part (80/90%) of the detail design/code activity.

[5] Rummler, G. and Brache, A, *Improving Performance, 2nd edition*, Jossey-Bass, San Francisco, 1995

[6] The NextPractice methodology is a major extension of a methodology called Data Structured Systems Development (DSSD) that had many of the same concepts but has been steadily refined over the last few years.

This DC/DG tool, in turn, makes it possible to be highly platform-independent since DC/DG tool targets a wide variety of popular databases (ORACLE, SQL Server, DB2, ACCESS) and an equally wide variety of languages (JAVA, C#, VB). Automating the code generation based on automatic database design means that the decisions about technology can be deferred or changed while the focus is on the business process modeling, not the technology for implementation.

Ultimately, the greatest strength of the database/code design/generation tool is its ability to support incremental design/prototyping/production. Not only is it possible to automatically design a normalized relational database from a set of independent user interface form/data structures and generate code to navigate against that database, when new or changed structures are introduced, the tool automatically redesigns/converts/regenerates the database and code—a development/maintenance dream.

Marrying eXecutable Business Processes(XBPs) and Agile Development

There has been an enormous amount written over the last decade about fast-track software management approaches such as eXtreme programming (XP) and Agile Development. The success that many organizations have had with Agile Development suggests that direct user/developer involvement speeds up the systems development cycle dramatically. In many cases, organizations are seeing very large increases in productivity and, perhaps more important, equally large decreases in cycle time. And these improvements are occurring using mostly current generation (manual) object development tools.

The rapid, incremental development of prototypes and products that the user can touch and feel, early in the development cycle, turns out to be critically important in all areas of business. Speed matters more today than ever. Due to advanced design/prototyping technology in the manufacturing world, the cycle time for the introduction of new cars has dropped from over seven years to less than two. The same is happening in all manner of other products as well. Product development has had its own agile development for a long time—it is called “concurrent engineering,” and it depends heavily on automated design tools.

What about Standards?

It is easy to get confused about standards. There are so many competing and apparently conflicting standards that exist in the business process/software development world, that it is hard to know which ones are important and which are not. UML, XML, BPML, BPEL, SOA, MFO, and MDA make a confusing alphabet soup. MDA, until recently, has been driven bottom-up from OO programming through OO design and OO analysis.

The business process world, on the other hand, has been largely driven top-down by methods like Rummler-Brache that come not from IT but from organizational analysis. With the advent of UML 2.0, it is possible to make the argument that business process engineering can continue to use the tools that they have found most effective and still fit within the umbrella of UML. In the case of NextPractice, we are fashioning a set of definitions that we refer to as

BP/UML 2.0. It is not a stretch at all to see that the future MDA 2.0 that is completely covered by UML standards.

Over the next few years, workflow management will become more and more integrated into Service Oriented Architectures, and XML will provide a better framework for user interface form/data structures. This will likely be accelerated as the big software vendors try to extend their offerings to provide their own version of eXecutable business processes.

Conclusion

The real-time enterprise is just on the horizon. The pressure to streamline business operations is significant in all areas. Today, more and more advanced organizations are using business process modeling tools to help them understand and model their business processes. But many of the most exciting business process changes die when implementation takes months or years to implement. Organizations need to be able to segue seamlessly from business modeling to information systems prototyping to information systems production. Marrying XBPs and agile development is a natural development.

In many organizations, business process renewal is on the front burner. Organizations are more and more aware that changing their IT systems without changing their business processes doesn't get at many of their most important problems. New generations of business processes are being proposed that can dramatically change how businesses perform. But software development is still trapped in classic waterfall execution cycle.

Most software development tools have not yet caught up with the needs of business. MDA 2.0 represents a second chance for the software community to step up to the plate. Objects are good, components are good, reuse is good, but what end-users want are eXecutable business processes that marry design, prototyping, and production. CASE failed to live up to its promise in the 80s because the technology was not there to do database/code design/generation, and workflow engines were in their infancy (as were the Internet and email). Today's technology makes it not only possible to conceive of concept to code but to create real eXecutable business processes.

Ken Orr is the Founder and Principal Researcher at the Ken Orr Institute, a Fellow of the Cutter Business Technology Council, and a member of the BP Trends Advisory Council.

Randy Hester is the Senior Software Engineer at the Ken Orr Institute.