A BPTrends Column September 2010

MDA Journal

David S. Frankel
Lead Standards Architect – Model Driven Systems
SAP Labs

David.Frankel@SAP.com
Mr. Frankel's SAP WebLog is:
https://www.sdn.sap.com/fr/sdn/weblogs?blog=/pub/u/55914

Author:
Model Driven Architecture:
Applying MDA to Enterprise
Computing

Business Applications in the Cloud-Mobility Era

The rise of cloud and mobile computing has tremendous potential to drive economic growth by providing new capabilities to the modern enterprise. However, applying these new technical capabilities to business software also has the potential to increase complexity to such a degree as to substantially reduce the benefits. Model-driven approaches to enterprise software, although not a cure-all, can play a role in exploiting the new technology in a tractable manner.

The Business Impact of Cloud Computing and Mobility

Despite the tendency to overhype new technology, cloud and mobile computing will substantially impact how computer software acts as a business enabler. We can see impact already, uneven as it might be at this preliminary stage.

Enabling New Business Models

Recently I spoke with an entrepreneur who was running a startup company that acts solely as a mortgage servicer, which is just one node in a multi-role business network. He told me that the software he needs to run his company would have cost hundreds of thousands of dollars just three or four years ago, whereas now he accesses the required software as a service for a recurring per-seat charge that, cumulatively, is orders of magnitude cheaper. The new software delivery model allows his startup to have a short enough projected ROI horizon to make the venture a risk worth taking. Beyond the reduction in direct software cost, avoiding the complications of installing the software and maintaining the computing infrastructure on which it operates drastically reduces the company's direct and indirect costs during its early life.

The story reflects a truly profound change that is likely to surface in many contexts. This lowering of a key barrier to entry for new business ventures means that we will see all sorts of specialized business models emerge that would not have been viable previously because of the baseline cost of getting the business off the ground. Higher baseline costs tend to force startups to address a larger scope in order to justify the investment. Lower baseline costs mean the trend toward specializing within a business network is likely to accelerate, and the use of software that digitizes information exchange within business networks will increase.

Re-Thinking Business Operations

In larger companies, a relatively small percentage of employees touch the enterprise software that supports back-office operations. Mobile computing is changing that by connecting employees who work on the manufacturing shop floor or in field operations.

Years ago we already had a glimpse of the impact of mobile computing on business operations when we encountered rental car company employees holding specialized mobile devices, checking us in when we returned cars. Now the proliferation of highly programmable, general-purpose handheld devices is starting to trigger managers to re-think their operations so as to exploit the technology. Once again, we see a lowering of a barrier to entry, as many operational changes require new software apps but not new mobile devices.

As is typical with the introduction of new technologies, there is a first wave of applications that tend to enable businesses to do the same old things more efficiently, while later waves involve more fundamental re-thinking of operations. So, in a follow-on wave today, some car rental companies send a keyless entry code wirelessly to a customer's general-purpose mobile device that then acts as the key to the car, and the device propagates information to the back-office on-premise system to register that the renter has picked the car up or returned it.

Of one thing we can be sure: The new technology will impact business operations in ways we have not yet imagined.

The New Level of Complexity: Implications for Scalability

Point solutions that exploit the new technology do not have to grapple much with the complications presented by scenarios that require coordination across cloud-based, mobile-based, and on-premise applications. But scalable enterprise software must address such complexity.

Orchestrating Across Cloud, Mobile, and On-Premise

A large manufacturer that uses on-premise software to support its back-office operations may have many relatively small suppliers that use cloud-based enterprise software. In this scenario, collaborative supply management requires coordination between the manufacturer's on-premise software and the suppliers' cloud-based software. Furthermore, the manufacturer and the suppliers may deploy mobile devices to employees involved in the collaboration.

There are also intra-enterprise business scenarios that require coordination among cloud, mobile, and on-premise software components, such as when a large enterprise uses cloud-based software at peripheral offices and operation centers with feed-in to the on-premise system, while other employees interact via mobile devices.

Thus, integrated solutions must ensure consistency among cloud-based, mobile device-based, and on-premise software. This represents a new level of complexity that enterprise software did not have to deal with formerly.

The Scalability Risks

In order to address the complexity, it helps to identify some of the risks that could undermine the scalability of enterprise software that orchestrates across the cloud, mobile, and on-premise levels.

A key potential problem is managing change. Installing new or updated business software components (or reconfiguring some components) at one level could impact not only other components at that level, but also components at the other levels. The software component dependency graph, complicated enough in traditional on-premise installations, becomes even more complex when coordinating across the three levels. Third-party extensions to applications, which have traditionally been an additional factor complicating the dependency graph, could complicate matters even more when they include extensions at the cloud, mobile, and on-premise levels.

Changes in the technical topology also can be tough to manage, such as changing the mix of public versus private clouds, moving some functions from on-premise to the cloud (or vice versa), or changing the deployment model for mobile devices. On-premise systems traditionally have had to manage technical topology change; however, the increase in the number of topological parameters in an orchestrated cloud/mobile/on-premise scenario increases the topological permutations non-linearly.

Model-Driven Approaches and Complexity Mitigation

A model-driven approach to software, while not a panacea, has the potential to help manage the increased complexity. The level of software complexity in the pre-cloud/pre-mobile era was sufficient to justify model-driven approaches, and, indeed, we have seen a slow but noticeable trend toward model-driven software over the past decade. The latest jump in complexity increases the motivation to go this route.

Application Transparency

The impact of the new computing technologies on enterprise software increases the need for applications to be transparent about their capabilities and their dependencies. Burying the capabilities and dependencies in low-level code makes it difficult to coordinate multiple components and manage change, in the best of circumstances.

A Consistent Component Model

Application transparency works best when built around a consistent component model. The application presents itself structurally as a set of collaborating components, each of which explicitly specifies the services it offers as well as its dependencies on the services of other components.

Although the technical infrastructure by which a software component provides and consumes services is apt to differ to some extent, depending on whether it resides on a mobile device, a cloud-based system, or an on-premise system, the basic paradigm of a component providing and consuming services, and the associated metadata schema, can remain consistent across these levels.

Metadata, Metadata, Metadata

A model-driven approach to software improves transparency by making application capabilities, structure, and dependencies available in machine-readable metadata, and by using that metadata to directly drive execution. The metadata can be entered into the system via graphical modeling tools, forms-based user-interfaces, textual languages, or some combination of the above.

It is not necessarily feasible or desirable to put everything in metadata and do away entirely with coding in lower level languages. Tools that enforce a good component-based, model-driven architecture – and such tools are critical! – fence off low-level code in confined spaces, such as in the detailed implementation logic of a component, but in a truly model-driven approach, such low-level code leverages the metadata, and programmers strive not to duplicate it.

Promoting Consistency

Using metadata based on a consistent metadata schema[1] to drive how components behave and interact with each other promotes a level of consistency that makes it easier to coordinate the functions of multiple components. The need to reap this benefit increases with the complexity of the systems at hand, making it especially compelling when orchestrating across the cloud-based, mobile, and on-premise levels.

Managing the Dependency Graph

When the dependency graph resides in machine-readable metadata, rather than being obscured within millions of lines of code, there is greater potential for tools to analyze and manage the impact of changing, retiring, or adding components.

However, a model-driven approach does not obviate the need for sensible dependency management on the part of application architects. Gratuitous dependencies that good architecture can eliminate are still problematical when widespread, even if the model-driven approach makes it easier to deal with them.

Managing Topological Change

Any good software architecture separates technology-dependent concerns from the functional behavior of a system. The more that metadata drives applications, the easier it is to enforce this separation, because the separation occurs at the level of transparent metadata rather than at the level of opaque code.

Some metadata should be topology-independent – particularly metadata that defines the functional aspects of component – and some should be topology-specific. This separation of concerns – again, when enforced with good tools – makes it more tractable to reconfigure topologies without disrupting the functional capabilities.[2]

Stay Tuned…

In future articles about business applications in the era of cloud and mobile computing, I will discuss the following subjects:

- Integrating business and engineering models to make software's responsibility for business process improvement more explicit
- The temptation to erect cloud silos
- Navigating the ocean of information generated by billions of new devices
- Semantic interoperability across cloud, mobile, and on-premise systems

---

[1] A consistent metadata schema does not preclude that the schema is actually a federation of multiple schemas, each of which focuses on a viewpoint of the system that is appropriate for a certain type of human or machine consumer of the metadata.

[2] Note for the technically inclined: It has been demonstrated for decades that enterprise software scales better when certain kinds of technical functionality are specified as much as possible by declarative metadata rather than by low-level code. Enterprise transaction-processing systems since the 1960s have to varying degrees prohibited application program code from activating and releasing record locks, initiating and committing transactions, spawning threads, and so on. Good infrastructure driven by metadata can do better, by recycling and pooling database connections, threads, and other expensive technical resources.

David Frankel has over 30 years of experience in the software industry as a technical strategist, architect, and programmer. He is recognized as a pioneer and international authority on the subject of model-driven systems. He has published two books and dozens of trade press articles, and has co-authored a number of industry standards.

David is a member of SAP's Technology Strategy team, which is part of the CTO's Technology and Innovation Platform organization. He focuses on standards for the financial services sector and for model-driven systems

BPTrends Linkedin Discussion Group

We recently created a BPTrends Discussion Group on Linkedin to allow our members, readers and friends to freely exchange ideas on a wide variety of BPM related topics. We encourage you to initiate a new discussion on this publication or on other BPM related topics of interest to you, or to contribute to existing discussions. Go to Linkedin and join the **BPTrends Discussion Group.**