Executive Summary

The steady rise in the abstraction level of technical software platforms enabled a similar rise in the abstraction level of programming languages and model-driven development tools. Business process platforms are a new type of platform at an even higher level of abstraction, offering business functionality that sits on top of technical platforms. Business process platforms enable a further rise in the abstraction level of model-driven tools that compose services into more complex services and applications.

Over time, composed services will migrate into the business process platforms, lifting the platform abstraction level still higher and pushing the abstraction level of model-driven tools yet higher. The ultimate outcome will be computer-assisted business management tools that realize the vision of executable business process models.

These advances will come together over a protracted period, and there are many issues that need to be worked through along the way. If we manage the transition properly, each incremental improvement toward the ultimate vision will offer good business value and propel the next move forward.

The Technical Software Platform: A Brief History

A technical software platform consists of technical infrastructure that provides computer-oriented capabilities. It frees application developers from having to program those capabilities in each application.

Operating Systems

The first platforms were computer operating systems. Before operating systems, applications directly addressed sectors and tracks on disks when they needed
to store and retrieve data, and had to maintain linked lists to keep account of the available free spots where they stored logically contiguous data. Operating systems present the illusion of a contiguous block of data called a file, and keep track of the free spots where virtually contiguous data is stored.

Operating systems gradually became more capable over the years. By the 1970s, there were operating systems that could mediate multi-user access to a processor. Early PC operating systems were primitive compared to the mainframe and mid-range operating systems of the day, but, over time, PC operating systems have grown more capable, and provide additional capabilities such as services for constructing graphical user interfaces.

**Database Management Systems**

Database management systems (DBMSs) came on line in the late 1970s. Applications no longer had to write their own data sorting and indexing routines. These and other capabilities of the DBMSs dramatically raised the abstraction level for data management, providing a new kind of technical software platform that extends the operating system.

**Network Technologies**

Network technologies, such as TCP/IP and IPX/SPX, and remote computing systems such as SDLC, made it easier for applications that reside on different computers to talk to each other. By providing basic network services they freed applications from having to keep track of how messages are broken up into packets and reassembled, and from other laborious tasks. They provided a platform for distributed computing, adding to the platform stack above the operating system.

**Transaction Processing Systems**

IMS is the classic, high-volume business transaction processing system. It provided a technical platform at a yet higher level of abstraction. Transaction systems manage the roll back of a partially completed transaction when some part of the transaction fails. A classic example is double-entry bookkeeping, which requires a debit and credit entry for each event entered into the general ledger. What happens when the credit entry fails after the debit completed (or vice versa)? Transaction processing systems manage this kind of issue.

Later-day “two-phase commit” transaction monitors extended the rollback capability to transactions that span multiple databases.

**Middleware**

The advent of middleware has provided the application developer with yet more capabilities. Middleware takes its name from its positioning between the application and the rest of the platform stack (see Figure 1). Middleware includes a number of technologies, including:

- **Distributed computing technologies** such as CORBA and RMI, which provide distribution services that are at a higher abstraction level than basic network technologies.
• **Application servers** that provide high-level services to applications for managing transactions, persistence, security, resource pooling, and more.

• **Message queue systems** for asynchronous messaging among applications.

• **Data integration engines**, which perform transformations of data from the format required by one application to that required by another application.

• **Service integration engines**, which mediate disparate data formats and different application program interfaces by encapsulating them with relatively high-granularity, reusable services.

At the same time that the abstraction level of technical software platforms has been trending upward, the abstraction level that we use to develop software applications has risen as well.

**From Assembly to 3GLs**

Early programs were written in 1s and 0s and literally manipulated bits and bytes in registers of the processor and memory addresses. Assembly language made it possible to use mnemonics to refer to registers and memory locations. Third generations languages (3GLs) such as COBOL, FORTRAN, C, and Java raised the abstraction level above the paradigm of the processor. One line of 3GL code can easily correspond to a dozen or more lines of assembly code.

**Modeling Languages and Model Compilers**

Model-driven approaches to software development, such as the OMG’s MDA® and Microsoft’s Software Factories, are raising the abstraction level still further.
They use models, crafted with general-purpose and domain-specific modeling languages, to specify applications and components, and use model compilers to generate 3GL code. Model-driven approaches to deployment and runtime management are also on the horizon.

**The Technical Platform as a Key Enabler for Model Compilers**

The rise in the abstraction level of the technical software platform is a key enabler of the model-driven approach. A model compiler would have a more difficult job without higher-level platforms. Imagine a model compiler that had to generate assembly code, and had to do so without the services of an operating system. It’s considerably easier to generate Java code that sits on top of an operating system, and easier still to generate Java code that uses J2EE’s plethora of middleware, or to generate C# code that leverages .NET. Figure 2 illustrates the fact that middleware shrinks the abstraction gap between the model and the platform, a gap that the model compilers have to fill in.

![Figure 2. Model Compiler Generating Code that Uses Middleware](image)

**The Rise of the Business Process Platform**

Amidst the cacophony of voices heralding the age of service-oriented architecture (SOA), a new kind of platform is emerging that will gradually change the face of IT as we know it. To understand the value of this new kind of platform, consider the current state of SOA.

**The Impedance Mismatch that Slows SOA’s Progress**

As I wrote in last month’s MDA journal, service-oriented architectures (SOAs)
are a response to the demands of value chain oriented business. Today we put SOAs together by wrapping relatively monolithic applications with service interfaces. There is inherently an impedance mismatch between the offering of well-defined services on the one hand and monolithic business applications on the other. Clever architects and tools manage that mismatch quite heroically and there are a significant number of success stories where good teams and good tools have produced workable SOA implementations by constructing a service-enablement layer between the monoliths and client applications (see Figure 3).

Figure 3. SOA via a Service-Enablement Layer

But impedance exacts a price in machine and human resources that have to be applied to bridging between the monoliths and the service wrappers. The basic shape of the monoliths is alien to the basic shape of a framework of well-defined services; the translation is expensive, and inevitably some things are very difficult to translate.

SOA would be more powerful if the monoliths themselves were to morph into well-defined frameworks of business services. Clients could access the business functions directly, with less impedance.
**The Business Process Platform: Removing the Impedance**

This direct offering of business services, obviating the need for impedance management, is the essence of a business process platform. A business process platform sits on top of a technical software platform. It offers services that execute business behavior, exposing the behavior directly rather than through wrappers struggling to encapsulate elements of monoliths.

*Figure 4: A Business Process Platform*

---

**Crawl, Walk, Run**

The dissipation of the impedance mismatch via refactoring of the monoliths will not happen overnight. The captains of the monoliths are deeply engaged in this endeavor, and useful results are already available from this work, but this is a case where the aphorism “crawl, walk, run” applies. Business process platforms are definitely coming. Their formation will stretch over a number of years, but their impact will be profound.

**Re-Deploying Resources**

As business process platforms remove the impedance mismatch, resources formerly dedicated to managing that mismatch will be redeployed to create powerful tools that make it possible to compose services together in order to support more complex business scenarios.
As recounted above, the rise in the abstraction level of technical platforms has made it feasible to raise the abstraction level at which we develop software, giving rise to model-driven software development. Business process platforms represent a new kind of platform at an even higher abstraction level, and they in turn allow us to push the level of abstraction of development tools even higher, giving rise to service composition tools.

The service composition tools will become more and more powerful over time, as some of the best technical talent moves away from managing the impedance and onto the task of developing these tools. As the tools become more effective, reusable compositions of services will be pushed into the business process platform, raising the platform’s abstraction level even higher. A good business process platform will not simply be a flat space of fine-grained services. In addition to simple services, it will offer compositions of services that support business scenarios that are more than simple single business behaviors.

**Computer-Assisted Business Process Management (CA-BPM)**

Computer-assisted business process management (CA-BPM) represents the ultimate ascendance of the platform and of the software development languages to the level of abstraction of the business itself. The goal of CA-BPM is to be able to use modeling tools to design, simulate, and tune business processes, and then to execute the business process models directly. Today’s CA-BPM tools do some impressive things, but the impedance mismatch hinders them from reaching their full potential. As an industry we face a tough road to fully realize the CA-BPM vision. It will take time no matter how well we manage the transition.

However, we can speed our progress by focusing on achieving useful objectives along the way. We should not have to realize the full vision before we can garner business value. Good business process platforms with good tools will help IT execute business processes better than before, and great platforms with great tools will help even more. The smart business process platform vendors will concentrate on productizing results along the long road to ensure that their customers get significant business value at each stage of the evolution; and they will not oversell what they have achieved at the relatively modest milestone points.

**Conclusion**

Amid the hype about service-oriented architecture and computer-assisted business process management, a truly far-reaching transition is under way. The transition is not a simple matter and will take some time to play out fully. The refactoring of monolithic business applications into business process platforms will gradually eliminate the impedance mismatch that slows the move to service-orientation, and will move us closer to the longer-range vision of executable business process models. This transition will definitely occur, but the pace of the transition will be determined by the skill with which we offer business value at every step along the path.