Business Process Platforms and Software Factories

An Idea Paper

David S. Frankel
SAP Labs, LLC
3421 Hillview Avenue
Palo Alto, CA 94304
+1 530 893-1100

david.frankel@sap.com

ABSTRACT

This idea paper discusses the role that business process platforms can play in a Software Factories approach to software development. A business process platform is a new kind of platform that sits on top of technical software platforms and facilitates the creation of business process oriented software factories that we call *business process factories*.

The paper starts by providing some basic background on how Software Factories integrates the concepts of domain-specific modeling languages and software product lines. It also describes the basic nature of a business process platform. Then it discusses how the reusable assets in a business process platform can form the basis for product lines within an overall Software Factories approach.

Categories and Subject Descriptors

D.2.11 [Software Architectures, Domain-Specific Architectures] J.1 [Administrative Data Processing, Business]

General Terms

Design, Management, Standardization, Economics

Keywords

“Business process” “business process platform” “software factories” “software factory” “product line” “metadata” “level of abstraction” “abstraction level” “software factory” “business process factory”

1. INTRODUCTION

One of the most important contributions of Software Factories is the integration of the concept of domain-specific modeling languages with the Carnegie-Mellon Software Engineering Institute (SEI) concept of *software product lines*.[1] The emergence of business process platforms adds a potentially powerful dimension to this integrated approach, pointing the way to a special kind of software factory called a *business process factory*.

For background, it is useful to review the basic concepts of product line practices, the software factories approach to product lines, and the evolution of software platforms. Sections 2 through 5 provide background on these topics that is necessary to comprehend the discussion of business process platforms and business process factories in sections 6 and 7.

2. SOFTWARE PRODUCT LINES

The SEI defines a software product line (SPL) as “a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.” [2]

The SPL approach (see Figure 1)[1] divides the software development process into two distinct but related processes—*core asset development* and *product development*. Core asset development produces a framework of reusable assets for the product line, and defines an architecture for the framework. Product development uses the framework to produce individual products. *A production plan* provides instructions on how to use the framework in accordance with the architecture in order to produce products.

[1] The “Sims Water Line,” depicted in Figure 2, and invented by Oliver Sims, uses the analogy of a water line to describe the separation of concerns between aspects of a system that surface to the application developer’s viewpoint, as opposed to aspects that the infrastructure handles below the surface.

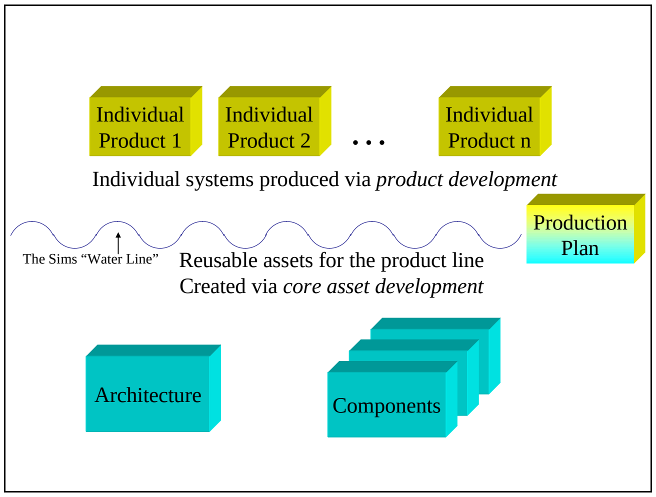
Individual Product 1
Individual Product 2
. . .
Individual Product n
Individual systems produced via *product development*
Production Plan
The Sims "Water Line"
Reusable assets for the product line
Created via *core asset development*
Architecture
Components

Figure 1: Software Product Lines

SPL addresses one of the main problems that has bedeviled component-based development—the problem of scope. Thoughtfully constraining the scope to which frameworks of components apply makes the problem of making the components truly reusable more tractable.

An example of an SPL is a set of products that manages risk for portfolios of tradable financial derivatives. Another example of an SPL is a set of role-access security products.

3. DOMAIN-SPECIFIC LANGUAGES (DSLs) AND PRODUCT LINES

The field of Generative Programming highlighted the notion that the core assets for a product line could include specialized specification languages.[2] It posited that a well-developed understanding of the variability among the members of a carefully-scoped product line makes it possible in many cases to define a language, at a high level of abstraction, that product developers can use to specify individual products. Special compilers process the high-level specifications and generate code that reuses the product line's core assets (see Figure 2).

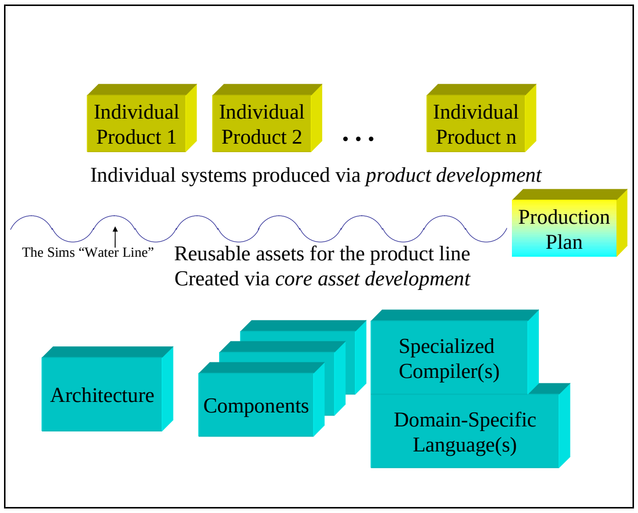
Individual Product 1
Individual Product 2
. . .
Individual Product n
Individual systems produced via *product development*
Production Plan
The Sims "Water Line"
Reusable assets for the product line
Created via *core asset development*
Architecture
Components
Specialized Compiler(s)
Domain-Specific Language(s)

Figure 2: DSLs and Product Lines

4. DOMAIN-SPECIFIC MODELING LANGUAGES

The Software Factories approach takes Generative Programming one step further, by integrating the role of metadata management and modeling languages tools into the picture.[3]

Specification languages—even if at a high level of abstraction—require mechanisms that store, parse, serialize, and version the metadata that specifications capture. The architects of the Software Factories approach recognized that the use of multiple domain-specific languages scales better when common technology and tools are available for managing metadata. Such infrastructures obviate the need to program metadata management from scratch for each language, and make it easier to integrate metadata originating from different languages.

The architects also saw that generic tools that support domain-specific modeling languages, such as Model Integrated Computing[4], provide just such a common infrastructure, and also leverage the power of modeling notations when constructing DSLs for product lines. Thus, domain-specific modeling languages (DSMLs) are an important aspect of Software Factories.

5. TECHNICAL SOFTWARE PLATFORMS

Technical software platforms are an important factor in the emergence of modeling languages and model-driven tools that operate at ever higher levels of abstraction.

5.1 The Rising Platform Abstraction Level

The abstraction level of technical software platforms has risen steadily over the years (see Figure 3). Operating systems were the original technical software platform. Over time, network

[2] Generative Programming contributes much more than this narrow summary describes. See [3].

[3] Software Factories of course integrates much more. We focus here only on one aspect its synergistic integration of a number of concepts.

[4] The Generic Modeling Environment (GME) is the key tool that the Model-Integrated Computing project has produced. See [5].

systems (such as TCP/IP), database management systems, and two-phase commit transaction monitors raised the abstraction level further.

Modern middleware has boosted the platform abstraction level another order of magnitude, distributed object technology such as and COM+ and CORBA; message-oriented middleware; and application servers based on .NET and J2EE that manage transaction processing, persistence, security, component-based development, data integration, and service-oriented architecture.

5.2 Enabling Model-Driven Tools

The rise of the platform abstraction level makes it easier for model compilers to generate code from relatively abstract models, because it shrinks the abstraction gap the compiler has to bridge.

Suppose that we have a DSML for specifying individual products within our example line of products that manage risk for portfolios of tradable financial derivatives. Now imagine that a model compiler that processes specifications expressed in our DSML had to generate assembly language, and could not even make use of the services of an operating system. Writing such a model compiler would be a Herculean task.

Now imagine that our model compiler can generate 3GL code, such as C# or Java, but only “raw” code that does not leverage middleware. That would be easier than generating assembly language with no operating system calls, but it still would be a formidable challenge.

Finally, assume that we can generate 3GL code that makes full use of middleware. Writing such a compiler is not easy, but the availability of middleware makes it more straightforward. That is why the rise of the abstraction level of the technical software platform is a critical enabler of the move to domain-specific modeling languages that operate at a high level of abstraction.

6. BUSINESS PROCESS PLATFORMS

A business process platform (BPP) is a new kind of platform that sits above the traditional technical software platform (Figure 3). It contains frameworks of executable service and business process components. Users of the platform compose specialized applications that support custom services, business processes, and analytics.

To understand the notion of a framework of services and processes better, think of the relationship between an application such as Excel, and the dozens of reusable COM components that Excel contains that make it possible to write clients that drive Excel in customized ways. The components are not random collections of functionality; rather, their respective behaviors are coordinated, each providing behavior needed to construct and drive a spreadsheet.

Similarly, a BPP can contain, for example, a framework of executable services and processes for managing accounts receivable. An accounts receivable framework’s services provide related functionality for processing invoices, rolling over accounts at end of month, managing discount policies, printing statements, reporting the impact of receivables on cash position, and so on. Some services are more comprehensive than others, with the most comprehensive executing more complete business processes.

A BPP pushes the abstraction level of platforms to new heights. In doing so, it is a key enabler of modeling tools at high levels of abstraction that ease application composition on top of the platform. Spurring the production of composite applications is the raison de etre of BPPs.

6.1 Composite Applications: A Business Example

A company had a set of systems covering procure-to-pay, order-to-cash, and manufacture-to-inventory processes. The company found a way to reuse them to become more competitive.

At the start of each week the company asked its suppliers which components they were selling at low prices. The company then ran its advanced planner and optimizer system to determine what it could make with those low-cost components. Once the company knew what products it could make, it ran an electronic auction for those products to maximize the selling price. When the auction resulted in an order, the company bought the components it needed (at low cost) from its suppliers and ran its manufacturing system to plan production for and build the finished products.

As a result, the company significantly increased its profit by minimizing component costs and maximizing the selling price, just by reusing existing systems reconfigured into a new business process. Figure 4 illustrates this innovative integration of existing functions and applications.

This is a classic example of a composite application. We can build such applications from existing functionality without BPPs, but BPPs package the functionality so that it is more readily reusable. BPPs are thus designed to ease the composition of such specialized applications and business processes. The combination of a BPP with model-driven composition tools can make a real difference when a company wishes to customize its operations, such as in the manner described in this example

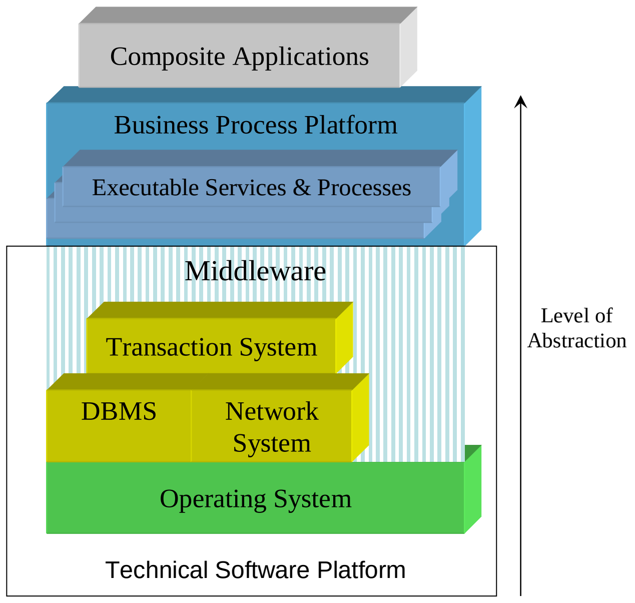

Figure 3: Platform Abstraction Levels

6.2 Business Process Platforms and Software Factories

Up to now, much of the discussion in the industry about the use of modeling languages and tools for composing applications on top of BPPs assumes that such tools are general-purpose, not specific to any particular business domain or sub-domain. Their avowed purpose is to make it easier to construct specialized business applications covering supply-chain management, enterprise resource planning, customer relationship management, and so on.

These general-purpose modeling tools have a potentially more powerful use than being a means for application construction. They can be also be a means for constructing software factories.

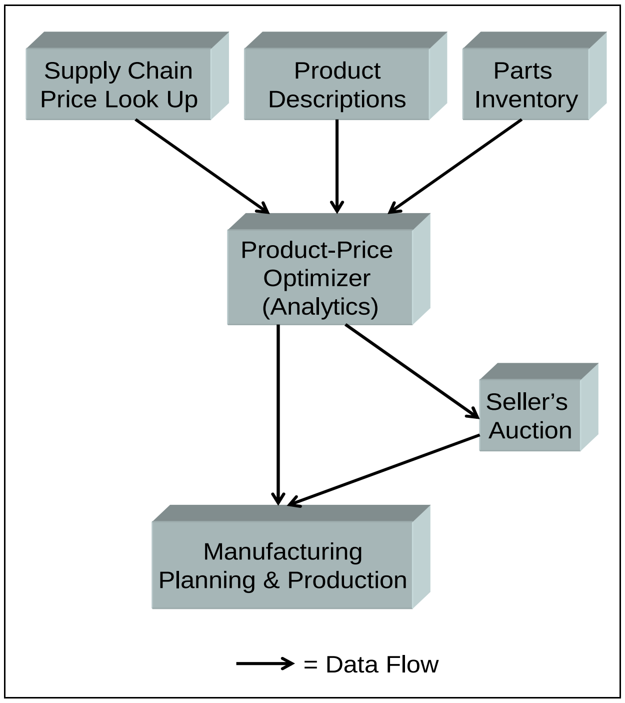
Supply Chain Price Look Up
Product Descriptions
Parts Inventory
Product-Price Optimizer (Analytics)
Seller's Auction
Manufacturing Planning & Production
= Data Flow

Figure 4: Business Integration Example

The company in our example was focused on one custom solution to a specific challenge. However, another way of using the general-purpose composition tools would be to build a software factory for a domain of products that optimize profit and production in the same fashion. The core assets would include a framework of related service and process components that specifically integrate the following functionality:

- Determining products to build based on part prices, inventory, and descriptions of products that the company can produce
- Conducting electronic auctions
- Manufacturing—planning and production

A modeling language specific to this domain would make it possible to customize individual products more efficiently than constructing each such product with the general-purpose composition tools.

An ISV might see an opportunity to offer such a software factory that has the potential for resale in a number of different lines of business; or, our original company might have many lines of business that have similar enough requirements such that the investment in this particular kind of software factory for internal use would be worthwhile.

6.3 Integrating General-Purpose BPP Composition Tools with DSMLs

In our business example the general-purpose model-driven tool for composing solutions on top of the BPP still has a valuable role to play, despite the fact that the software factory provides a DSML. The general-purpose tool provides a way to create the software factory's reusable assets more efficiently than would be possible without such a tool.

The architects of the software factory would have to weigh the tradeoffs between two basic ways of using the general-purpose modeling tool.

6.3.1 A Framework of Pre-Compiled Components

One way would be to use the general-purpose tool's model compiler to generate components for the factory's reusable asset base. The reusable framework would thus consist of specialized, executable service and process components. The model compiler for the DSML, when processing a specification for a specific member of the product line, would generate code that uses those compiled components.

6.3.2 A Framework of Model Components

The alternate approach would be for the DSML's model compiler to generate general-purpose models that it hands off to the general-purpose modeling tool's model compiler. In this case, the reusable framework's components would be parts of models defined in terms of the general-purpose modeling language—in other words, they would be model components. The DMSL model compiler would use these model components to assemble the model for a specific product, before handing the assembled model off to the general-purpose model compiler.

7. BUSINESS PROCESS FACTORIES

The term *business process factory* refers to the special kind of software factory described above, which is built on top of a BPP. The advent of BPPs, and the general-purpose modeling tools that go with them, enable this kind of software factory and continue the push up the abstraction ladder (Figure 5).

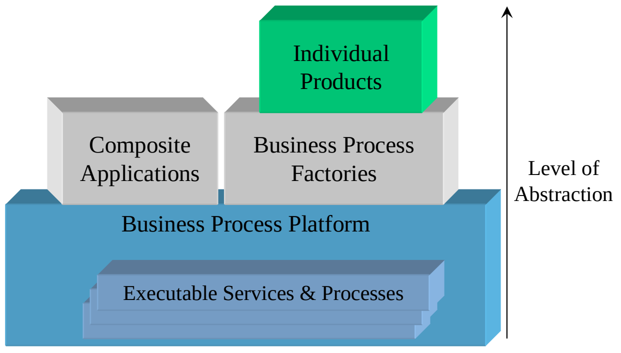
Individual Products
Composite Applications
Business Process Factories
Business Process Platform
Executable Services & Processes
Level of Abstraction

Figure 5: Business Process Factory: A Special Kind of Software Factory

Business process factories can play an important role in fostering the ability of a business to adapt to changing conditions. Within the broad scope of a general-purpose BPP and associated modeling tools lie myriad opportunities for ISVs and IT organizations to develop software factories that help enact business processes with relatively narrow focus on specific, innovative value propositions.

7.1 Business Process Factories and Value Chains

In today's value chain oriented business, companies need to erect and reconfigure B2B value chains rapidly. Since BPPs build on top of technical platforms that provide infrastructure for service-oriented architecture (SOA), a business process factory can also play a role in reconfiguring value chains.

A value chain factory is a special kind of business process factory that promotes a product line of value chains that share a common focus on a particular line of business or kind of B2B transaction.

7.2 Business Process Factories and Moore's Fault Line

Geoffrey Moore, the originator of the "crossing the chasm" concept, uses the notion of a fault line to describe the distinction between business functions that are "core" for a business as opposed to those that are "context."[4] Core functions competitively differentiate a company's offerings. Context is everything else that the business must do in order to succeed but that provide no differentiation.

Moore posits that one key to the long term health of a company is the ability to redeploy or reconfigure context resources such that the resources can contribute to core. This is one of the key value propositions of a BPP. By encapsulating context services and processes in frameworks of reusable components, the BPP makes it possible to reuse and redeploy services and processes in new, innovative ways that contribute to core.

A business process factory approach enhances this basic BPP value proposition because, rather than thinking in terms of constructing one-off special solutions on top of the platform, it orients to constructing new, specialized frameworks that support families of solutions that contribute to core.

8. SUMMARY

A business process platform is a new kind of platform built on top of a technical software platform. The primary value proposition of a business process platform is that it helps businesses redeploy resources "from context to core" in the parlance of the Moore fault line.

A business process factory is a software factory built on top of—and, therefore, at a higher level of abstraction than—a business process platform. Business process factories enhance the value of a business process platform, by raising the level of abstraction even further and thereby increasing the power that companies can bring to bear in their efforts to redeploy resources from context to core.

9. ACKNOWLEDGMENTS

Thanks to David Burdett, of SAP Labs Palo Alto, for the business example in section 6.1.

10. REFERENCES

[1] Greenfield, J., and Short, K., with Cook, S. and Kent, S., *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*, John Wiley & Sons, 2004

[2] Carnegie-Mellon Software Engineering Institute, *Software Product Lines*, http://www.sei.cmu.edu/productlines/index.html

[3] Czarnecki, K. and Eisenecker, U.W., *Generative Programming: Methods, Tools, and Applications*, Addison Wesley, 2000

[4] Moore, G., *Living on the Fault Line: Managing for Shareholder Value in Any Economy*, Harper Collins, 2000.

[5] Vanderbilt Institute for Software Integrated Systems, *Generic Modeling Environment*, http://www.isis.vanderbilt.edu/Projects/gme/