

A Reuse Parable

Mahesh H. Dodani, IBM Software Group, U.S.A.

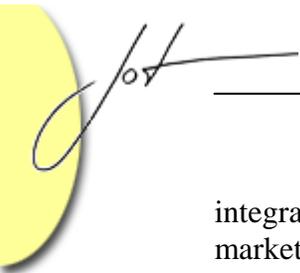
1 THE SETUP

The ever-growing complexity of computer applications and systems has made it necessary for the major forces in software engineering to integrate in order to effectively and efficiently design and implement solutions. This article presents a parable for leveraging assets to address this complexity and serves as a guide for reuse in the 21st century.

First, let's meet the players in the story: complex business needs, large reusable assets to accelerate solution development and ensure that established best practices are used, and agile methodologies to help build solutions to address these complex needs.

The complexity of computer applications and systems continues to grow at an exponential rate. In a very short period of time, we have evolved from single applications addressing a specific functionality or set of requirements on a dedicated mainframe to an integrated set of applications representing an enterprise e-business that are able to sense and respond to fluctuating market conditions and provide products and services to customers on demand (<http://www-106.ibm.com/developerworks/ondemand/>.) Companies can quickly increase or decrease their requirements as their markets change (<http://www.timesonline.co.uk/article/0,,9023-856626,00.html>), and will be able to acquire additional functions or infrastructure as they need them.

To keep up with this ever-growing complexity, software engineers have targeted their work efforts on building reference architectures, patterns, frameworks, and open standards that capture best practices and proven experiences into customizable reusable assets. Of course the larger the asset, the more useful it is in helping drive the entire solution and the more difficult it is to reuse. In this story, we focus on IBM Patterns for e-business (<http://www-106.ibm.com/developerworks/patterns/>) which define a set of proven, reusable architectures that can drive the design, development, implementation and extension of e-business applications. They match business challenges with Business and Integration patterns, use proven Application and Runtime patterns, populate the Runtime patterns with pre-tested Runtime Product Mappings, and establish best practice guidelines for application design, development and management. The large reusable assets are the composite e-business patterns, which combine several core business and



integration patterns to provide a proven solution for a class of problems (e.g. e-marketplaces (<http://www-106.ibm.com/developerworks/patterns/b2bemp/index.html>).

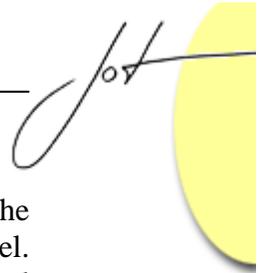
To be effective, we need to facilitate the reuse of these large assets in the context of methodologies and workproducts that the architects are familiar in using to design solutions. Such industrial-strength methodologies (<http://www-306.ibm.com/software/rational/library/whitepapers/rup-bestpractices.html>) typically span distinct phases—Inception, Elaboration, Construction and Transition. Each of these phases has a well defined set of activities and tasks that produce associated models which guide the architect to understand the requirements of the system and design a solution to meet these requirements. The primary phase in which reusable assets, best practices and experiences can be leveraged is the Elaboration phase wherein the requirements are understood and the solution is designed. As you move through the project lifecycle and iterate through the actual development of the solution, the assets that can be leveraged are smaller and more focused on the implementation of the solution (e.g. design patterns, coding guidelines). These latter phases are much better understood and well documented.

So how can large reusable assets be leveraged to design complex solutions? In order to make these assets and best practices reusable on real projects, we need a guide that embeds these best practices and experiences into an industrial-strength methodology and defines a step-by-step process for leveraging the large reusable assets.

To facilitate a short story in the next section, we focus on the Elaboration phase in which the solution to the problem is designed in three major steps. The first step is to establish the requirements of the system defined through use case models. The next step is to design a solution that meets these requirements through an architectural model which shows the logical design of the solution. The logical design is transformed into an operational physical design by defining the components that make up the logical system and the packaging of these components into applications that run on a physical infrastructure of servers, storage and networking. The physical design is primarily modeled using the component and operational models. Note that we will only cover the first two steps in order to show how to leverage large reusable assets.

The goal of the first step is to define the requirements for the solution and understand the complexity and scope of the solution. The process of outlining the solution requirements involves identifying both the functional and non-functional requirements and documenting architectural principles that will drive the solution. In most methodologies, the functional requirements are derived and documented in a use case model. Once these requirements are defined, it is important to start the process of leveraging reusable assets. We start by creating an inventory of potential reusable asset candidates. To be useful, these large reusable assets must define generic use cases that define the scope of applicable solutions in which they can be used. Matching the required use cases to these generic use cases is a sure way to find the right reusable asset. The unmatched use cases define the delta requirements that need to be addressed by extending the asset to a solution that addresses all the requirements.

The goal of the second step is to design the solution that meets the requirements of the system, and is captured by an Architecture model. In building the solution, the



Architecture model typically goes through two stages. The first will identify the subsystems of the solution; we will refer to this as the Subsystem Architecture model. The second will denote the logical nodes within the solution; we will call this the Logical Architecture model. The matched large reusable asset can be used to seed both the subsystem and logical Architecture model. The seeded Architecture model can be extended to handle the delta requirements by using other patterns, frameworks, and best practices.

2 THE STORY (ABRIDGED VERSION)

To illustrate the use of the guide in practice, we discuss it in the context of a typical complex problem of modernizing disparate government agencies to facilitate collaboration and integration. This e-Government modernization targets the need for governments to collaborate across countries, agencies, regions, states and the private sector to determine incidents that have the potential of causing problems, and managing the incident to solve the problem. Agencies that participate in some aspect of the incident have information pertinent to the incident and subject matter experts that can resolve the incident. These experts are required to participate in a collaborative environment to work together to resolve the incident effectively and efficiently. Figure 1 captures these requirements as a use case model.

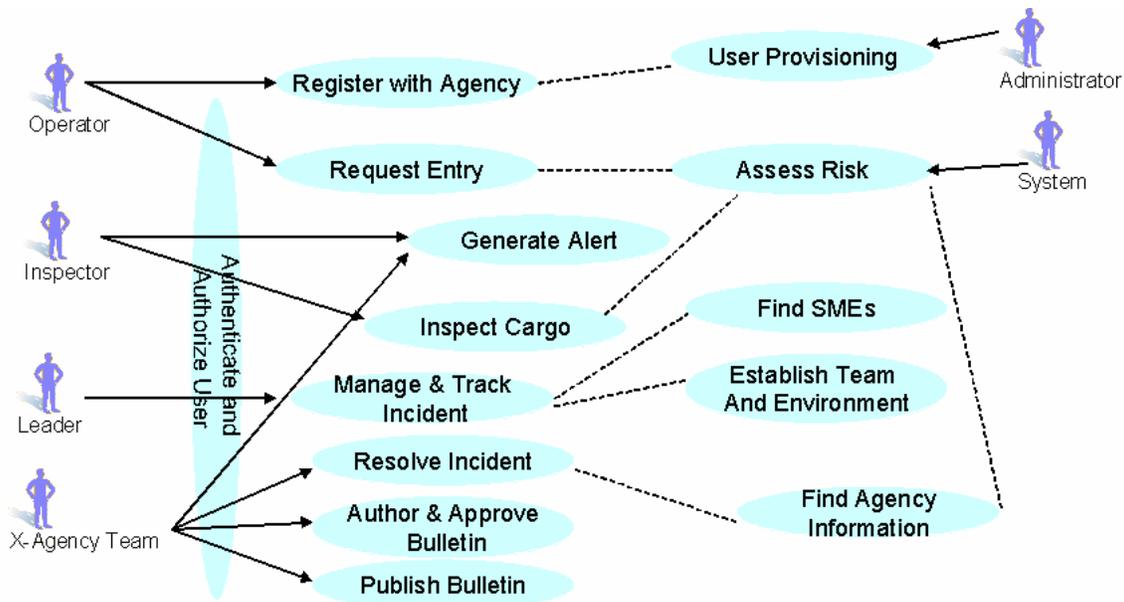


Figure 1: e-Government Requirements Use Case Model

Following the guide, we build an inventory of potential large reusable asset candidates. After a thorough investigation, we focus our attention on the Portal composite pattern (<http://www-106.ibm.com/developerworks/patterns/portal/index.html>.) This large reusable asset is promising because it tackles business and IT drivers that are close to the

requirements and focus on collaboration, information sharing, and application access. The generic use case model for the portal composite pattern shown below highlights the business and integration patterns that the use cases are associated with:

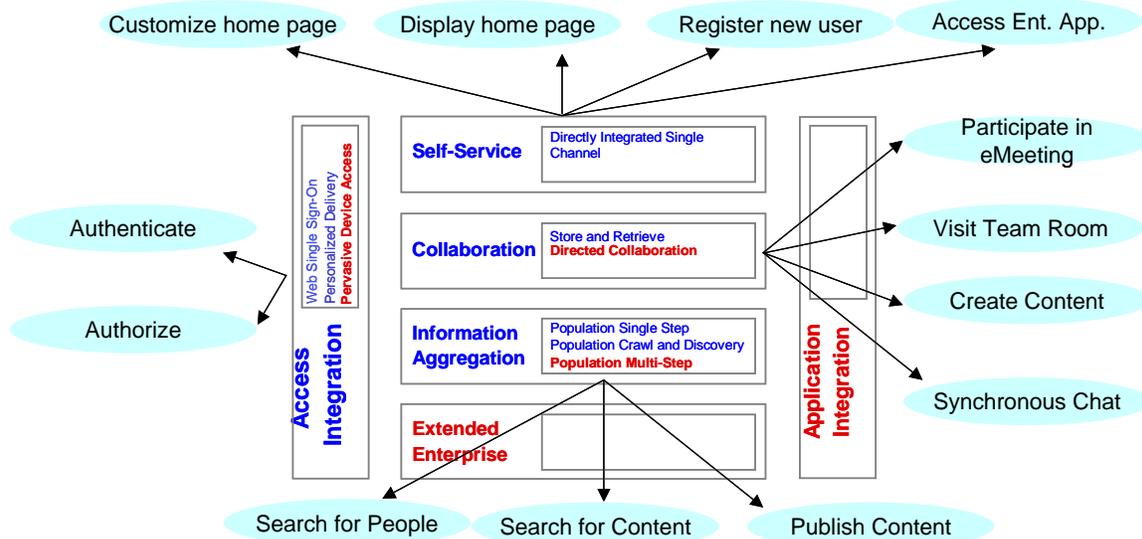


Figure 2: Generic Use Case Model For Large Reusable Asset

Matching the intended solution to the Portal composite pattern tackles all of the requirements except those captured by the Generate Alert, User Provisioning, Inspect Cargo, Assess risk, and Find Agency Information use cases. These unmatched use cases will be the focus of our solution design after we have used the large reusable asset to seed the initial design. Note that this large reusable asset has taken care of a large percentage of the requirements, and so the 80/20 rule is intact – 80% of the solution can be seeded by leveraging the large reusable asset allowing us to focus on the 20% delta requirements.

The design of the solution begins with the Subsystem Architecture model depicting the functional blocks of the solution representing logical grouping of use cases. Figure 3 shows the case study's Subsystem Architecture model after the Portal composite pattern has been applied. Note that the pattern is capable of handling a large percentage of the solution as shown by the dashed lines in the figure. The subsystems and integration points which were not absorbed by the Portal composite pattern represent the delta requirements that will be the focus of the rest of the solution design process.

The next step is to transition the Subsystem Architecture model into the Logical Architecture model. The large reusable asset should provide a logical architecture model which can be used to seed the solution. The Portal composite pattern provides a runtime pattern (<http://www-106.ibm.com/developerworks/patterns/portal/access-sso-runtime.html>) which is a middleware representation of the functions that must be performed, the network structure to be used, and the systems management features for most portal solutions.

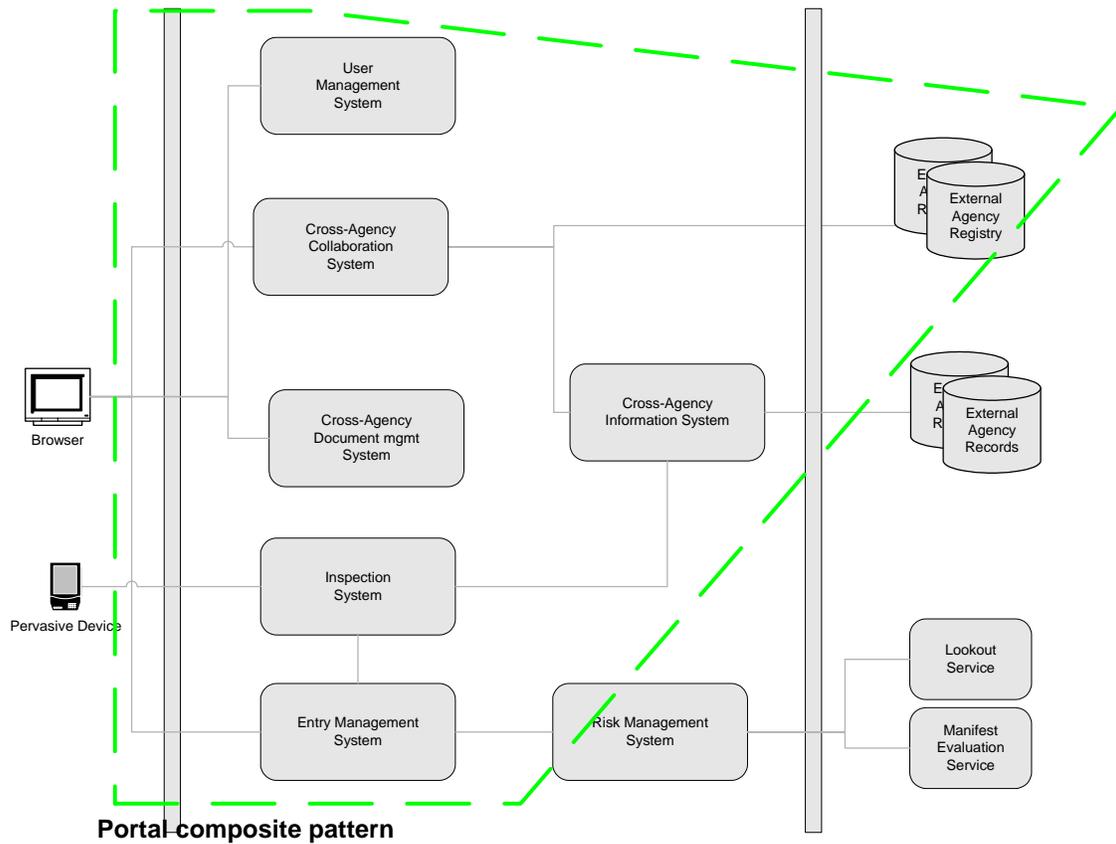
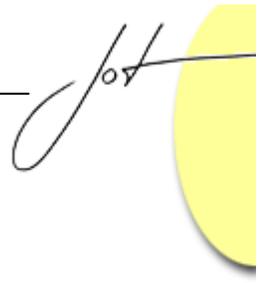


Figure 3: Seeding the Solution with the Matched Large Reusable Asset

Figure 4 shows the Logical Architecture model for the e-Government case study. It shows the Portal composite pattern seed that was applied and the first step in extending the logical middleware to address one delta requirement (this extension is highlighted by the blue oval.) Note that the architecture model exposes the major middleware nodes, their roles and the interfaces between them. These logical nodes can then drive the physical model through best practices, usage guidelines, reference implementations and personal experience.

It is important to note that this story is abridged, and does not do justice to the complexity of creating large reusable assets, matching these assets to a set of requirements and the customization and extension of these assets to build complex solutions. For a more complete coverage of leveraging large reusable assets, please visit <http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247011.html?Open>.

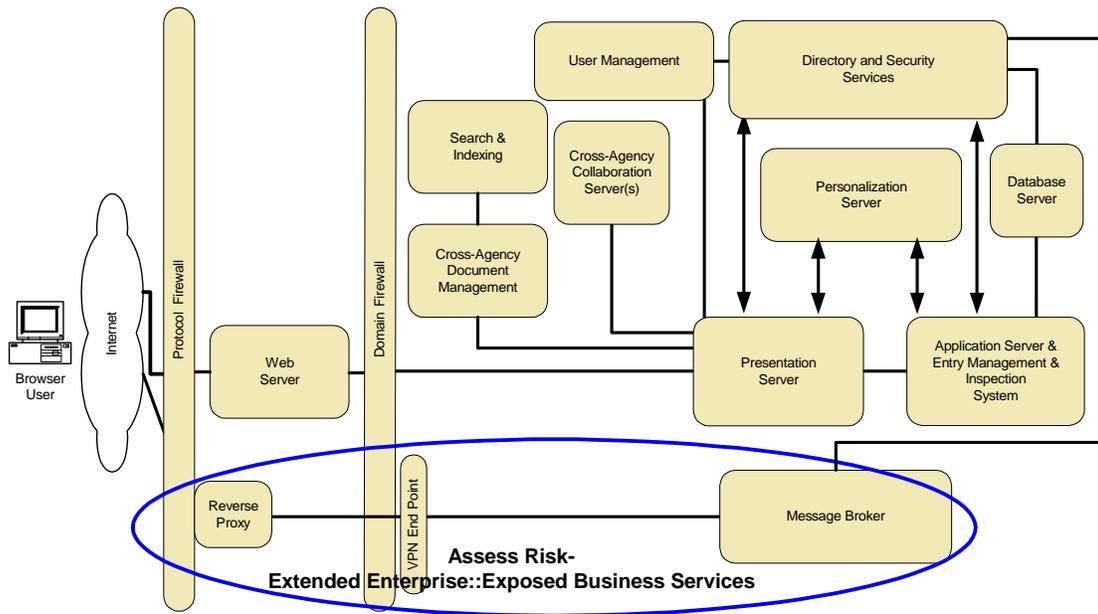


Figure 4: Seeding Logical Design With Large Reusable Asset and Extending For Delta Requirements

3 THE MORAL OF THE STORY

As the story shows, the larger the asset the more effective the reuse. However, for it to be effective, the large reusable asset must be integrated into the methodology and provide the necessary support to be able to match it early in the process, seed models through the project lifecycle and be integrated with the phases, activities, and tasks of the methodology.

In conclusion, the moral of the story – as you sow, so shall you reap!

About the author



Mahesh Dodani is an e-business architect with IBM Software Group. His primary interests are in enabling individuals and organizations to tackle complex e-business solutions. He can be reached at dodani@us.ibm.com.