Experience in Using Business Scenarios to Assess COTS Components in Integrated Solutions

Sharon Lymer
IBM Toronto Lab
IBM Canada Ltd.
lymer@ca.ibm.com

WenQian Liu
Dept. of Comp. Science
University of Toronto
wl@cs.toronto.edu

Steve Easterbrook
Dept. of Comp. Science
University of Toronto
sme@cs.toronto.edu

Abstract

Constructing software by integrating commercial off-the-shelf (COTS) components is widely practised, particularly in the IT service industry. For vendors of COTS components, requirements engineering is particularly challenging. To continually improve their products, vendors must identify and analyze problems that occur when their components are used in a wide variety of integrated solutions, and they must anticipate new applications in which their components could be used. In this paper, we describe a scenario-based framework developed at the Software Group division of IBM Corporation (IBM SWG) The framework mimics the solution integration process for new business opportunities, allowing the development teams to evaluate their components, discover and resolve integration issues, and to surface new requirements for future releases. This paper describes the framework, gives an example of its use in a business scenario, discusses the experience of using this framework at IBM SWG, and relates the lessons learned.

Keywords

Commercial-off-the-shelf (COTS), Software Integration, Business Scenario

1 Introduction

In order to reduce development costs, companies in many industries have reduced the size of in-house development teams and increased outsourcing for building customized applications for their own business. The use of middleware and commercial off-the-shelf (COTS) components tends to dominate over building from scratch in such development projects. This approach is commonly known as *software integration*, and a system that is put together using multiple COTS components is an *integrated solution*.

In this paper, we focus on the problem of maintaining and evolving the COTS components themselves, and in particular the problem of surfacing new requirements for such components. For example, IBM SWG produces a portfolio of COTS components from the WebSphere®, DB2®, Lotus®, Tivoli®, and Rational® brands. These components have numerous capabilities including application and process integration, information management, collaboration, systems management, and software development.

Managing the evolution of such a broad portfolio of COTS components presents many new challenges for requirements elicitation and management:

- Interface compatibility between COTS components.
- Capability overlaps or gaps.
- Time to value – COTS components must have a short set-up time and learning curve, require little installation effort, and

be easy to tailor and incorporate into a solution.

- Usability – The overall usability is lowered if the components look and act different from one another, target different user groups that have varying skill sets, or if there is a discontinuity in solution tasks across the product.

Although the use of COTS components may cut down the development cost of individual software components, extra effort is required to deal with the kinds of integration issues listed above. Customers of COTS components are rapidly discovering these hidden costs, leading to a demand for improvements to COTS components that can reduce the overall effort needed to build integrated solutions. This demand increases the pressure on software vendors and IT service providers to change their own development paradigm.

At IBM SWG, we responded to these challenges by creating a team that will use customer solutions to understand the integration issues that arise when building solutions with COTS software. Furthermore, we needed to gain insight into the full customer development life cycle using our COTS components by exploring how individual integration solutions are envisioned, designed, implemented, tested, deployed, and managed.

Our approach, the *Business Scenarios Framework (BSF)*, is similar to scenario-based approaches described in Sutcliffes article, Scenario-based requirements engineering [13]. Both use some form of scenarios to represent and understand the problem. However, general scenario-based approaches are mostly used to elicit requirements and generate initial design fragments of the intended system. Our framework provides means to explore and describe the entire life cycle of an integrated solution such that integration issues can be identified and missing features uncovered from the selected COTS components. The OpenGroup also introduced the use of business scenarios as a way to view requirements in relation to one another in the overall problem context and as a key input to the development of the architecture [2]. Our framework makes use of business scenarios in a similar way but with a broader perspective.

We expand the idea of a business scenario to incorporate not just a particular way of using a solution, but the entire cycle by which a set of COTS components might be integrated and deployed in response to a business need. Our business scenario represents a particular business situation and the *integrated* software solution required to accomplish a set of business goals. It describes the solution architecture and maps components to both current and future COTS components. Most importantly, it provides a framework for acting like the solution builder[1] throughout the solution development process and for evaluating the COTS components as they are customized, integrated, deployed, and managed. Collectively, the business scenarios help IBM SWG product teams to not only identify integration issues, but also surface requirements for the next generation COTS components.

Currently, we have developed nine business scenarios. In this paper, we will describe the development of one of these in detail, namely the Employee Workplace business scenario. This scenario addresses the problem that, in many businesses, employees struggle to manage content, to locate and access information, and to overcome technological challenges. We will describe BSF itself, illustrate its use with the Employee Workplace scenario, and then discuss our experiences in applying BSF at IBM SWG.

2 Background

Today, user-centered design and requirements engineering methodologies deal with identifying and stating software requirements. User-centered design, in particular, emphasizes understanding and meeting the needs of human users with strong focus on the design and implementation of the user interface. Requirements engineering identifies the goals of stakeholders and precisely elaborates the desired application behaviour to determine requirements. With the focus on human users

[1] A solution builder refers to either an IT services provider or the IT organization of a customer.

and application-level needs, requirement elicitation methods from these approaches are not well-suited to understanding and determining COTS software requirements. For example, there is little or no guidance for gathering requirements for COTS components that are to be used by large numbers of broader solutions.

Much of the literature on COTS components in software engineering is oriented towards the COTS components consumer to provide advice on procurement (e.g., see [1, 3]). Various frameworks have been proposed for the evaluation and selection of COTS components [6, 7, 12], and alternative development processes are proposed to effectively leverage the needs of COTS-based system construction [11, 4]. Recent reports have also described the experiences and lessons learned in integrating COTS components into systems (e.g., [5]). Our perspective is different from these in that we come from the vendor's point of view, and our concern is with managing the requirements of COTS components to determine what goes into the future releases.

Unlike application-level software that tends to have well-understood human users and usage, COTS components often have both system and human users, and usage scenarios that are as varied as the applications that utilize the COTS components. Additional requirements arise from the concerns of the systems integrator. The systems integrator must deal with conflicting software dependencies and prerequisites, duplicate components, mismatched architectures, and missing or incomplete functionality coverage across the COTS components required in an integrated solution. This makes the process of requirements elicitation much more difficult.

Another challenge is the need for a COTS component to integrate and inter-operate with other COTS components to produce the overall capability needed by an integrated solution. Again, the problem is the multiplicity of solutions. Several solutions may require the same underlying COTS components but each solution drives a different set of usage patterns and combinations of interactions across the middleware. Different non-functional requirements and constraints are highlighted by each solution.

Finally, it is essential that COTS software vendors understand not only the runtime requirements of their product, but also the other aspects of its use. For example, what tools are provided to aid with overall solution development? What are the administrative interfaces that assist with the management of the solution? What programming model is used to access the COTS function? What are the required performance characteristics?

Our approach provides a systematic framework for eliciting requirements that arise from all these concerns. Requirements elicitation techniques are usually conceived as suitable for one of three types of software development: building custom software for a single customer, creating generic software that is targeted for mass market, or procuring COTS components and developing glueware that will serve in the company's own business context [8]. Our approach combines aspects of all three of these contexts. Our COTS components themselves are intended for a mass market, but the requirements can only be elicited by considering how the components will be procured and integrated by a specific customer, and the problems that may arise when a solution provider attempts to use them for a custom solution.

BSF is an adaptation of scenario-based methods that have been widely used in requirements engineering and software design [13]. The contexts within which scenarios are used are diverse, ranging from the social environment of the system to the event sequencing in a design [10]. The purpose of using scenarios is typically to describe the "continuum from the real-world descriptions and stories to models and specifications" [13]. Our framework is a variation of scenario-based methods. A business scenario includes a COTS-based software application and its development life cycle; it includes both the business context of the system and detailed event sequences within the system.

3 Business Scenario Framework (BSF)

Three years ago, IBM SWG started using business scenarios to describe the needs of

its customers for custom software applications built by integrating COTS components. The approach has gradually evolved to its current shape influenced by the Rational Unified Process® (RUP®) [9] and OpenGroup's Business Scenario approach [2]. The Business Scenario Framework (BSF) is a formal description of both the artifacts and process in this approach. The approach includes discovering customer needs, investigating potential integrated solutions that best fit these needs, and mimicking the actual development activities to identify weak spots in the integration. Findings are used to produce requirements for major releases of individual COTS components.

The framework has three key parts: a business scenario, a defined process, and supporting artifacts. A *business scenario* is a collection of artifacts that describes a business problem faced by a customer (or a group of customers with similar needs) and the desired software solution composed of COTS components. The artifacts included in a business scenario are *Solution Definition*, *Architecture*, and *Design*. Together, these artifacts illustrate the desired solution in generic or prototypical terms such that only the common needs of similar businesses are addressed with respect to the intended business problem. Focus is placed on the use of COTS components in accomplishing the business goals. Details of each artifact are given in Section 3.1 together with the supporting artifacts.

The process provides guidance in creating and validating a business scenario and directs the activities for analyzing and evaluating any covered COTS components. An important part of the process is the emulation of a typical development cycle from requirements gathering and architectural design to implementation and testing by a solution builder.

The supporting artifacts are produced as a result of validating the business scenario and evaluating covered COTS components. These include a solution builder critique, analysis and evaluation criteria, COTS component (enabling) use cases, implementation assets, test cases, and results (expressed as an experience report and recommendations).

Since the purpose of the framework is to test the compatibility and interoperability of participating COTS components during integration, a partial solution with just enough detail is constructed during implementation to evaluate the intended business scenario. The framework does not, and is not intended to, result in a comprehensive integrated solution.

3.1 Artifacts

The BSF artifacts consist of business scenario artifacts and supporting artifacts. The former include solution definition, architecture, and design. The latter include a solution builder critique, analysis and evaluation criteria, COTS component use cases, implementation assets, test cases, and results. We discuss each in depth below.

As a business scenario artifact, *Solution Definition* describes the business opportunities and problems to be solved by the integrated solution for a selected business situation. End users are identified and described. The user experience with the integrated solution is highlighted through usage stories and described in more detail through use cases. The desired features are outlined. Constraints, assumptions, and dependencies are identified, and non-functional requirements such as usability, performance, or reliability are defined. The solution definition artifact is represented by a RUP vision document and use case document.

It is important to review and validate the information in the solution definition with appropriate solution builders. The responses and comments from validation sessions are documented in the supporting artifact *Solution Builder Critique*.

The *Architecture* artifact is composed of an architecture overview document (AOD) and associated Unified Modeling Language (UML) models. The document content is based on the results of RUP activities for determining the early solution architecture. It details architectural goals and constraints. Key items from the domain model are identified and modelled in UML class diagrams. The logical view of the architecture presents the proposed subsystems. The necessary subsystem components are determined through realizations of architecturally significant use cases shown as sequence diagrams. The *Design* artifact refers to revisited

and refined architectural artifacts with the addition of design perspectives expressed through the process, deployment, data, and implementation views. The design artifact is expressed as a document styled after the RUP software architecture document (SAD). Both the architecture and design artifacts are part of the business scenario.

The *Analysis and Evaluation Criteria* is a crucial supporting artifact. It identifies available COTS components that correspond to subsystems or components of the integrated solution, presents the selected releases of the COTS components to be used in further emulation activities, and describes areas of focus for analysis and evaluation. The features of the integrated solution are prioritized for investigation, and important functions or capabilities are highlighted.

In addition, some aspects or facets of the overall solution development, such as installation, configuration, implementation, and security management, are selected for exploration and evaluation. Important use cases are identified, and specific evaluation criteria for each facet are defined in the analysis and evaluation criteria document. Exploration of the solution development facets leads to the identification and development of another relevant supporting artifact, the *COTS Component Use Cases*.

Emulating solution development activities as part of the BSF process leads to a partial *Implementation* of the integrated solution. Any code resulting from the implementation phase is provided as further supporting artifacts within the BSF. In preparation for the test phase of the BSF process, *Test Cases* are developed and included as additional supporting artifacts.

Finally, the outcome of the analysis and evaluation activities are provided in the *Results* supporting artifact. It is documented in two parts: an experience report and a set of recommendations. The experience report presents an evaluation of the COTS components used in the various emulation activities and discusses the level of satisfaction of the experience with an emphasis on any inhibitors. The recommendations suggest improvements to the COTS components with respect to interoperability within the context of the integrated solution, and in some cases, suggest COTS component enhancements or changes to better meet the identified needs.

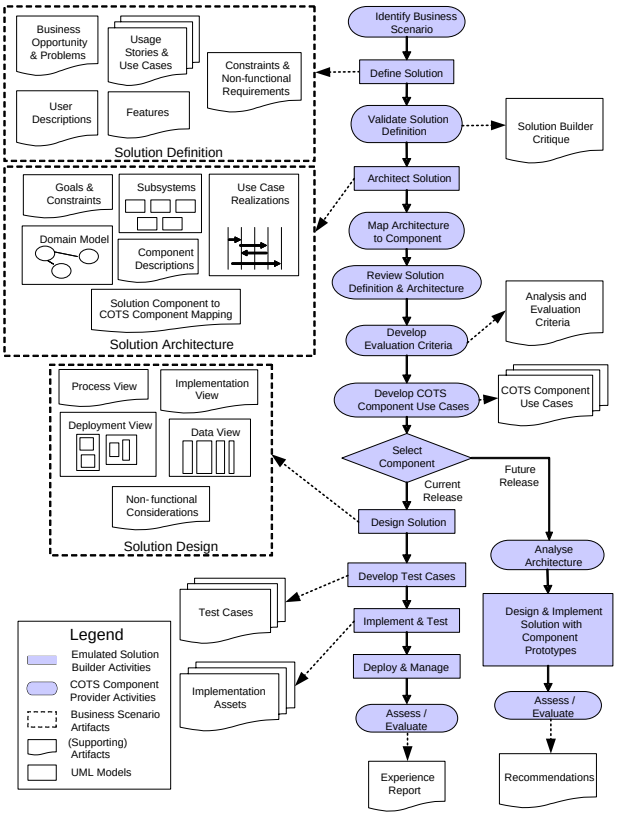

Figure 1: The BSF Process.

3.2 Process

The process is based on the Rational Unified Process (RUP) platform [9] and begins with the identification of a business situation requiring a software solution that uses at least one COTS component of interest. There are two main parts: (i) emulation of an actual integration project by solution builders through a full development cycle from requirements gathering and architectural design to implementation and testing, and (ii) analysis activities for determining and evaluating relevant COTS components in terms of their capability and behaviour. Figure 1 shows how emulated activities are interspersed with analysis activities.

The emulation process begins with the analysis of the business situation, identification of

needs, and proposing of the integrated solution. This information is captured in the solution definition artifact. It is essential that this definition be validated by appropriately identified COTS software customers or IT service providers as to how well it reflects the actual business requirements, how accurately the business problems are represented, and whether the stated features and capabilities meet their needs.

The next step involves the production of an architecture overview through the identification and description of (i) main objects in the business domain such as a customer order, (ii) use cases that affect the architecture, (iii) the logical subsystems, their major components and interfaces, and (iv) realizations of significant use cases using sequence diagrams.

COTS components (including both existing and projected components) are then mapped to the logical subsystems and major components identified in the architecture. The choice of COTS components influences the solution development life cycle. Exploring the effects of each choice is essential to understanding COTS component requirements and reducing integration effort. Such effects include new constraints and utilities, and in general, new ways of handling various facets of solution development. These facets include: solution evaluation and planning, installation, configuration, implementation, quality of service, security management, and solution management including problem determination and the ability to support and service the solution.

The effects are explicitly identified for each solution. To minimize the impact on the overall solution development process, evaluation criteria are defined for more rigorous analysis and assessment. In particular, new and changed solution builder tasks, as a result of the introduction of the COTS components, are determined and documented as COTS component use cases.

Then, we act like solution builders and design the integrated solution using a released COTS components. The design includes the subsystems details, packages and classes, the definition of the runtime architecture in terms of processes and threads, its physical distribution, implementation layers, and its persistent data. It is documented and reviewed to ensure its consistency with the solution definition and architecture developed earlier, and to verify its emphasis on desired focus areas and high priority features or capabilities. In addition, test cases are provided to verify these aspects.

In the implementation step, a partial solution is constructed according to the design. Emphasis is placed on the exploration and evaluation of the COTS development utilities and environment using the *analysis and evaluation criteria* produced earlier. Use cases associated with the implementation facet are executed in order to evaluate this aspect of solution development.

During the testing step, the partial implementation is tested, not to eliminate defects from the integrated solution, but to assess the operation of the COTS components within the solution. Again the analysis and evaluation criteria are used to assess the usage and capabilities of the COTS components in solution development facets such as the installation, configuration, and solution management, especially problem determination.

Finally, the software solution is deployed to, and managed, in a test environment that is configured similarly to a typical production environment. Like the previous step, the focus of this step is to execute the COTS component use cases and evaluate any effects on the solution management, quality of service, and security facets.

To complete the process, the experience with the released COTS components is assessed and documented in a report. Component defects are recorded and the resolution is tracked using the same support channel as an actual COTS component customer. The experience report and defect records are the key results of the process and lead to specific recommendations.

It is possible to modify the overall process to evaluate, and hopefully, improve the design of future component releases, well before component implementations are available. In this case, early designs or prototypes for future components are used during the emulation of the design phase. Naturally, it is not possible to realistically mimic solution builder activities in this situation, so more analysis is required rather than explicit evaluation. Typi-

cally, an architectural analysis is conducted to explore the integration of the COTS components within the solution. This results in the identification of integration issues or gaps. In addition, early component prototypes are used to investigate the affects of the COTS components on the other facets of the solution development life cycle such as utilities for implementing, deploying, or managing the solution. The outcome of the exploration is a set of recommendations for enhancing the planned component or the addition of new features to close any identified gaps. These recommendations are documented and communicated to the COTS component development teams for consideration in their requirements process.

3.3 Roles

Developing a business scenario requires time and resources. However, the additional effort is justified because integration problems that are hard to anticipate might remain undiscovered in the COTS components and thus contribute to higher integration costs. To do it well, particular skills and experience are necessary. The framework recommends the following roles.

A *Program Manager* outlines the solution definition, arranges its validation, produces the Solution Builder Critique, schedules and tracks activities, and handles communication with COTS component development teams. Project management expertise and a strong technical background are important for this role.

A *Business Scenario Architect* identifies use cases and features for the solution definition, develops the architecture and the analysis and evaluation criteria, and oversees analysis and evaluation activities. This is usually a senior architect with broad technical expertise and in-depth knowledge of the COTS components associated with the scenario.

A *Business Scenario Developer* produces use cases and the design artifacts (under the direction of the business scenario architect), and implements a partial solution using selected COTS components. If future components are being evaluated, then the implementation is done with early component prototypes. A representative set of solution test cases are produced to exercise key aspects of the solution. The primary goal of the business scenario developer is to conduct analysis and evaluation of the COTS components throughout the implementation, test, management, and deployment phases of the solution development cycle. This role is suited for an experienced software developer.

A complete business scenario process may involve a business scenario architect, a program manager, and one or two business scenario developers. With such a team, the overall process from business scenario identification to the production of an experience report takes approximately six to nine months. Naturally, the solution implementation and test activities need to be aligned with the availability of the desired components. If the business scenario is used in the analysis of future components, the activities leading to the solution architecture take three to four months and the design and analysis activities are constrained to about three months.

3.4 Example

Teams at IBM SWG started using the business scenario approach about three years ago. Initially, only the WebSphere brand COTS components were explored. However, in the last year, the business scenarios began to include COTS components that span across the IBM Software portfolio. Currently, nine business scenarios have been developed and three are introduced below.

Employee Workplace scenario describes a solution that enhances employee effectiveness and productivity by providing employees with self-service applications and makes it easier for them to collaborate across departments and locations. It also improves the employee's ability to effectively manage the growing volumes of corporate information such as documents, e-mail, digital images, and Web content.

Mergers and Acquisitions scenario describes the situation of a merger between a large traditional insurance company providing insurance through agents and a new Internet operation working entirely through the Web. The solution provides a single customer view of the merged companies to give a single administrative view and cut operational costs.

Private Exchange scenario examines a supplier's use of a Business-to-business (B2B) exchange to streamline the delivery of product information to potential buyers. The solution centers on the automatic processing and publishing of product information.

We present, in greater detail, a more recent scenario – Employee Workplace (EW). The complete artifacts are too extensive to present here because of a lack of space. Partial text and diagrams extracted from the EW artifacts give the reader a flavour of the actual content.

Employee Workplace presents a solution to the challenges many businesses face in improving organization-wide efficiency and productivity and dealing with increasing volumes of digital content. The business perspective for the solution is provided in the solution definition artifact and includes a description of the business problems that need to be addressed, the proposed features of the solution, end users, and use cases.

The primary business problems fall into four categories: (i) employee productivity inhibitors, (ii) difficulties managing information, (iii) difficulties finding information, and (iv) business risks.

The key features of the EW solution are:

- Collaborative capability: e-mail, instant messaging, calendar, and Web conferences
- Document and Web content management, including life-cycle management and search
- Task management
- Records and retention management
- Employee directory
- Application integration services

There are three primary types of end users in this solution: (i) employees who use the workplace to carry out various roles within the organization, (ii) records managers responsible for creating and administering the records and retention management system, and (iii) solution administrators who maintain the running solution.

There are many use cases for the Employee Workplace solution. A few important use cases are:

- Collaborating with other employees using instant messaging and e-mail.
- Creating, browsing, and editing documents or Web content.
- Searching for content.
- Creating content life cycles and completing tasks associated with the life cycle.
- Creating and maintaining a retention plan including creating, viewing, suspending, and destroying records.
- Managing access to users, groups, and resources.
- Backing up and restoring repositories.

The Employee Workplace solution definition was reviewed with a few major IBM customers. These customers agreed that the solution definition reflected the reality of their businesses and accurately described the business problems and resolution needs. This discussion and comments were documented in the *Solution Builder Critique*, but this is not publicly available for confidentiality reasons.

Another major artifact is the architecture of the business scenario. The first section of this artifact is the discussion of architectural goals and constraints. Examples include single point of access to business services, security, and ease of solution administration.

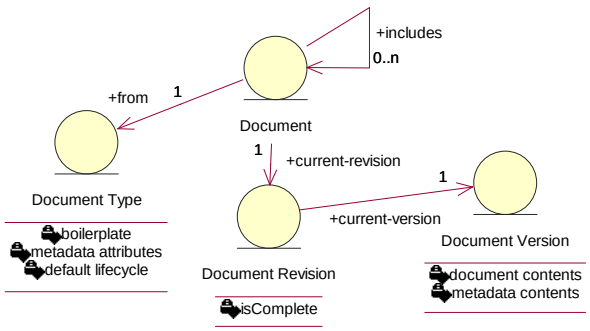

Figure 2: Document Types

An important part of the architecture is the domain model, which is described using UML class diagrams. A key business object for Employee Workplace is the *document type*, the

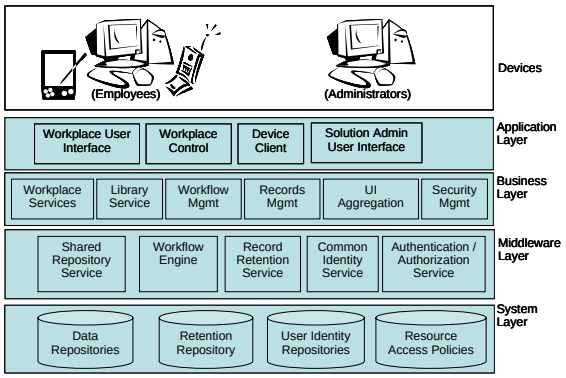

Figure 3: Subsystems and Components

structure to which each document conforms (see Figure 2). It may specify required metadata attributes and a boilerplate for the document body. For an owner's manual, the metadata attributes may be the car's make and model, and the boilerplate may be the layout of chapter and section headings such as Introduction, Maintenance Schedule, etc.

Figure 3 shows the subsystems and components. Each subsystem is briefly described in the *Architecture* artifact.

A key part of the architecture is the exploration of significant use cases in terms of their realizations as described in the solution definition. For example, "*Create new document revision*" is a use case and its realization is shown in Figure 4 using a UML sequence diagram.

Finally, the architecture includes a mapping of both current and future IBM software products that correspond to components in the subsystem diagram. The EW architecture includes the current[2] IBM products: IBM® Workplace Team Collaboration™, Messaging and Documents 2.0, DB2® Content Manager 8.2, DB2 Records Manager 3.1.2, WebSphere® Portal 5.0.2, and Tivoli® Directory Server 5.1.

The design artifact is a document that discusses the various perspectives of the Employee Workplace design using a particular set of IBM COTS components as the components and covers the implementation, deployment, and data views. This material is too detailed to include in the paper.

[2]We cannot disclose information about planned future IBM products.

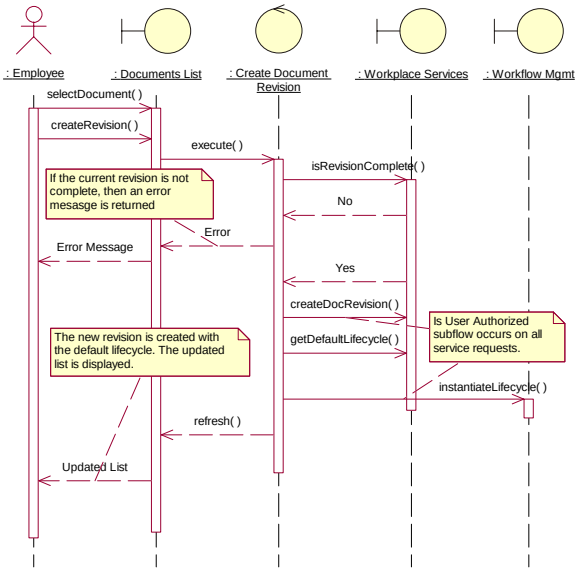

Figure 4: Use Case Realization

The analysis and evaluation criteria are documented for the selected component sets. The criteria are presented by feature and also by solution development facet. Based on the usage stories in the solution definition, important capabilities for each feature are identified and prioritized for evaluation. Expectations for each component set are noted, including any known limitations. In addition, this artifact provides a detailed discussion for each solution development facet. In Employee Workplace, the installation and configuration facet was selected, since the IBM® Workplace™family is a relatively new addition to the IBM software portfolio. The solution management facet was selected because of the large numbers of components in the solution and the goal of ease of administration. The security facet was selected because it is important for employees to be able to transparently access the services offered by the various COTS components constituting the solution with a single sign-on. Finally, the solution development facet was selected, because we wanted to explore the ability for solution builders to integrate existing systems into the Employee Workplace and to develop the types of content that need to be incorporated in the solution.

The COTS component use cases and the cor-

responding users are identified and elaborated for each solution development facet. For example, the use cases for Employee Workplace solution development include:

- Develop, test, and deploy templates for Web content presentation.
- Develop and test portlets for application integration.
- Develop and test new workplace applications such as departmental applications, expense reporting, etc.

The implementation assets for Employee Workplace were created in Java^{TM}code and are not included this paper. The test cases are also too extensive to include, but follow a typical format. However, there are a limited number since the purpose of the testing is not to evaluate the solution implementation. Rather, the goal is to test the interoperability of the COTS components in the solution and to exercise the important features as identified in the analysis and evaluation criteria.

The Employee Workplace business scenario effort involved the evaluation of recently released IBM COTS components and also the analysis of upcoming components. Based on the selected solution development facets, the experience report evaluated the experience of installing, configuring, developing, managing, and securing the COTS components in the context of the Employee Workplace. Recommendations for more effective interoperability with future releases have been conveyed to the component development teams.

4 Lessons Learned

The lessons learned in our application of the Business Scenario Framework (BSF) are based on our observations and include areas for improvement and identification of limitations that need to be addressed in future.

Lesson 1 *Using business scenarios, we are able to improve the design and quality of our COTS components. Component developers gain a deeper understanding of the customer requirements that affect component usage and behaviour by considering the concrete business situations and the various development perspectives offered in the business scenario.*

Without the information provided by business scenarios, development teams tend to take a product-centric approach and typically have little opportunity to explore the integration requirements that arise when the COTS component is used with others in a solution. There is a tendency to focus on the production usage rather than to consider product requirements arising from software development facets such as product installation and configuration, using the product to develop and manage the integrated solution, and the effects of their product on the overall solution quality.

Business scenarios provide two perspectives to the developers. The first provides the context of use – a spectrum of applications of the component. The second highlights the spectrum of solution development activities as the COTS component is used throughout the integrated solution development life cycle from requirements gathering to implementation, test, and deployment.

Typically, the COTS component is designed to serve a particular need and thus the set of required capability, such as messaging or database services, seems clear. However, unless various integrated solutions are considered, interoperability challenges may be missed. For instance, some COTS software provides underlying services to other COTS software in the solution. For example, database services may be required by application services or content management. It is essential that each COTS component has the same prerequisite level of the underlying COTS software. We have found the prerequisite releases to be inconsistent in many cases. Similarly, the configuration of shared services needs to be consistent across the integrated solution. Often, we have found that different configurations of a particular component are demanded by other components in the solution. These mismatches must be resolved so that the service may be shared, and avoid the need to install multiple copies of the same COTS component.

Once the runtime behaviour is established, it is important to consider the other facets of

component behaviour. For instance, the programming model and development tools for utilizing the COTS component in integrated solution development are a key to its acceptance. The COTS component needs to be easy to install, configure, and manage since it is only a part of a larger integrated solution effort. In fact, business scenarios were originally introduced in IBM SWG to reduce the effort our customers were spending on these activities with early releases of the WebSphere Application Server product. Installation and configuration of Application Server has vastly improved since version 3.0. Other COTS components have made improvements to the development environment to make it easier to build and test the integrated solutions described in the business scenarios.

Lesson 2 *The framework introduces a rigorous process and establishes the essential content and activities of business scenarios, which not only helps to maintain consistent quality among scenarios, but also reduces effort.*

When business scenarios were first developed, an ad hoc approach was employed in the production of business scenario artifacts. The business scenario development process emphasized activities such as scenario selection, development, validation, and testing. The process provided little guidance in the creation of artifacts or their composition. This resulted in large variations in the type of content and level of detail. This lack of consistency made it more difficult to communicate business scenarios to development organizations and also made it harder to teach new team members how to develop scenarios.

Our current approach, using the BSF presented in this paper, specifies the business scenario development activities and artifact composition in more detail. Since our goal is to mimic our customers' experience with our COTS components, we adopted RUP (Rational Unified Process) in order to formalize the process of mimicking customer activities and allow us to produce consistent artifacts. The standard RUP vision, use case, and software architecture document templates were tailored to our needs. The use of the RUP-based templates determine the composition of the business scenario artifacts. Guidance and sample text indicate the necessary level of detail. This makes the approach easier to teach and is allowing the team to develop consistent artifacts. In addition, use of the RUP vision encouraged us to focus on additional aspects of the integrated solution such as the users of the solution and the solution features that were not always considered or documented with the previous approach. Often a better understanding of these aspects of the integrated solution leads to a better understanding of the COTS software users and the features needed to support the solution.

In some cases, the new approach increases the effort to produce the business scenario *solution definition* and *architecture* artifacts since it is a more comprehensive approach. However, the time to review these artifacts is reduced since the artifact composition is more clearly defined and easier to compare to, and contrast with, other business scenarios. It is also easier to produce the design artifacts using the consistent and more detailed information provided in the solution definition and architecture. Overall, the business scenario development effort is reduced.

Lesson 3 *It is crucial to balance the effort to produce the business scenario against the potential to discover integration issues or surface new requirements. The effort is affected by the choice of the integrated solution described by the business scenario, its breadth and the extent to which the solution is implemented.*

It is extremely important to carefully select the business scenario in order to maximize the benefits of the scenario development effort. To be beneficial to the COTS provider, the solution described in the scenario needs to apply to a significant share of COTS component customers. This is accomplished by finding common business objectives and developing general solutions to meet those objectives. The generalized solution must preserve the essential aspects of the solution requirements derived from unique business situations. Also, in order to maximize exploration of component integration, the selected business scenario must require several COTS components that are of interest to the component provider. Increasing

the coverage of COTS components often results in a broader solution scope and a larger scenario. This leads to other potential problems, as discussed below. Interestingly, as the success of the business scenario approach is recognized, it is our experience that COTS component development teams lobby for the inclusion of their component in a scenario. Avoid "force fitting" a COTS component into an existing scenario. Instead, seek a solution that naturally includes a COTS component and is of significance to the business of component customers.

Large business scenarios have the advantage of wide domain coverage and are likely to uncover more integration issues and surface additional new requirements. However, because of their size and complexity, there is a higher cost to create, implement, and test the resulting solution. It is also more difficult to communicate a larger business scenario to interested COTS product development organizations. Product teams are reluctant to resolve integration issues or consider new requirements if these requests are not easily justified with information from the business scenario. Conversely, a smaller scenario is less expensive to create, implement and test, and it is easier for the development organization to see the role of the component under development in the solution described by the scenario. However, a less complex solution typically integrates fewer COTS components, making it less useful for discovering interoperability issues. If the business context of the solution is too narrow, critical elements may be excluded and the needed behaviour of the COTS component may not be realized. Therefore, it is important to weigh the effort to develop the scenario against the potential to discover problems when choosing the scope of a business scenario.

Ideally, the BSF would provide guidance to assist with decisions about solution selection and scope. From our experience, some suggestions are: consider the strategic importance of particular COTS components to the component provider; identify the key features of the integrated solution through validation of the solution definition and focus on these features in the partial implementation of the solution; identify important solution development facets for the integrated solution, and minimize emulation activities for other areas.

Lesson 4 *While the use of business scenarios helps to identify and resolve integration issues, a more important benefit of business scenarios is to gain a better understanding of the integrated solutions COTS software customers want. This understanding is essential to determine the COTS component requirements that these solutions demand.*

Our original motivation for BSF was to respond to the integration challenges COTS component customers experience during the development of integrated solutions with IBM software products. Using BSF to think and act as the customer, IBM SWG has successfully identified problems such as conflicting software dependencies, overlapping components, and difficulties installing and configuring COTS components within the integrated solution.

However, we have observed that this approach tends to surface new COTS software requirements not previously considered. The business scenario illustrates the use of a combination of COTS components that must work in concert to provide the capability for the integrated solution. When considered together, especially in terms of common goals for the overall solution, it may be possible to utilize the same underlying service; for example, the access and management of shared information such as user credentials may be shared.

Based on this observation, we believe that the BSF provides a powerful method for surfacing COTS component requirements. For this reason, we have made the elicitation of requirements our priority in our future efforts with BSF.

Lesson 5 *The use of an independent team to produce business scenarios reduces the effectiveness of analysis and evaluation of COTS components, because of communication challenges with the product development organization. To overcome this problem, we propose integrating the business scenario framework into the product development process as a requirement elicitation method.*

The challenge of communicating the findings from each business scenario with the COTS

product development teams can lead to limited acceptance of the recommendations. It can be particularly hard to convince the development teams to accept and implement new requirements. This is largely due to the effort involved in determining the relevance of the business scenario to a particular COTS component. Usually, the team has to go through the entire set of business scenario artifacts in detail in order to understand the business context for their product, and thus understand the rationale for the business scenario analysis and evaluation. Recommendations that are accepted are often not adopted since they are received too late in the product development cycle.

In future, we want to explore the adoption of BSF by the COTS product development teams and have business scenarios integrated into the development process. Product teams would be responsible for selecting and using scenarios oriented to their product, increasing the relevance and avoiding the problem of acceptance. Tighter integration of the processes and the elimination of an independent organization resolves the communication issues and promotes the elicitation of relevant requirements only.

Although this may solve our current problem, it raises new issues to be explored in the future. How feasible and effective is it to have individual product teams independently creating business scenarios? Will business scenarios continue to cover multiple products so that the requirements that lie between products are identified? How is effort coordinated across products so as to avoid the redundancy that may occur because of overlapping scenarios developed by different product teams? How is the collective effort balanced against the individual efforts?

5 Conclusions and Future Work

As COTS-based system construction emerges to be a mainstream development method, new challenges are faced both by the consumers and providers of COTS components.

In this paper, we presented the Business Scenario Framework (BSF), a variation of the scenario-based approaches in requirements engineering, to address some of the challenges faced by COTS component providers. We have applied evolving versions of the framework to nine scenarios during the past two and a half years, to identify integration issues in the current releases of the COTS components and surface requirements for future releases. We described in detail the artifacts and processes involved in a business scenario development and our experience in doing so.

The lessons we drew from our experience reflect both positive results and limitations. The approach enhanced the quality of COTS components in the short term by resolving integration issues. However, we need more comprehensive techniques for identifying and managing new requirements for long-term success. The approach also deepened developers' understanding of customer requirements. The defined process of the BSF allowed us to maintain consistent quality among scenarios and reduced the effort.

Two areas for improvement were identified. Firstly, we need better techniques to scope the scenarios, both in the content and coverage. Secondly, we need to overcome communication difficulties between the business scenario team and the COTS product developers. We believe this can be solved by integrating the BSF into the product teams' development process as a requirement elicitation method.

In future, we would like to gear the BSF towards the requirement elicitation of COTS components and middleware. We plan to pilot the incorporation of BSF into the component development process. The approach will be validated by surveying individuals from the development organization regarding the usefulness of applying BSF for eliciting requirements and improving the component design. We will focus on how results collected from the scenarios help the component teams to make architectural decisions, not only in discovering issues, but also in successfully implementing the resolutions of the issues. We also plan to define more fine-grained tasks, refine the analysis and evaluation on the scenarios to reduce the dependence on experienced designers, and explore the roles and skills needed for the elicitation work.

Acknowledgements

We thank Marin Litoiu and Mike Starkey for feedback and review; the SWG Business Scenarios and Solution Test teams at IBM for the business scenarios. This work is funded under IBM CAS Fellowship and National Sciences and Engineering Research Council of Canada.

About the Author

Sharon Lymer is a senior Software Group architect working at the IBM Toronto Software Lab. She has over 20 years experience with the entire software development cycle, from architecture and design to implementation, test, and support. Sharon is a member of the Professional Engineers Ontario. She graduated from the University of Toronto with a Bachelor of Applied Science.

WenQian (Wendy) Liu is a Ph.D. candidate in Computer Science at the University of Toronto. She received her Hon. B.Sc. in Computer Science and Mathematics and M.Sc. in Computer Science both from the University of Toronto. Her primary research interest is in software architectural design and requirements engineering.

Steve Easterbrook is a Professor of Computer Science at the University of Toronto and the Academic Director of the Bell University Labs. He received his Ph.D. from the Imperial College of Science, Technology and Medicine at the University of London (1991). His primary area of research interest is in requirements engineering and consistency management in large software development projects.

Trademarks

Disclaimer

The views expressed in this paper are those of the authors and do not necessarily reflect the views of IBM Corporation.

References

[1] International Conference on COTS-Based Software Systems. `http://www.iccbss.org/`.

[2] OpenGroup Guidelines on Business Scenarios. `http://www.opengroup.org/architecture/togaf7-doc/arch/p4/bus_scen/bus_s%cen.htm`.

[3] SEI: COTS-Based Systems (CBS) Initiative. `http://www.sei.cmu.edu/cbs/`.

[4] Cecilia Albert and Lisa Brownsword. "Meeting the Challenges of Commercial-Off-The-Shelf (COTS) Products: The Information Technology Solutions Evolution Process (ITSEP)". In *Proceedings of ICCBSS'02*, pages 10–20, 2002.

[5] Thomas G. Baker. "Lessons Learned Integrating COTS into Systems". In *Proceedings of ICCBSS'02*, pages 21–30, 2002.

[6] Jesal Bhuta and Barry Boehm. "A Method for Compatible COTS Component Selection". In *Proceedings of ICCBSS'05*, 2005.

[7] Sudipto Ghosh, John Kelly, and Roopashree Shankar. "Enabling the Selection of COTS Components". In *Proceedings of ICCBSS'05*.

[8] Ann M. Hickey and Alan M. Davis. "Elicitation technique selection: how do experts do it?". In *Proceedings of RE'03*, pages 169–178, 2003.

[9] Philippe Kruchten. *"The Rational Unified Process An Introduction Second Edition"*. Addison Wesley, 2000.

[10] Kari Kuutti. "Workprocess: Scenarios As a Preliminary Vocabulary". In J.M. Carroll, editor, *Scenario Based Design*. 1995.

[11] M. Morisio, C. B. Seaman, A. T. Parra, V. R. Basili, S. E. Kraft, and S. E. Condon. "Investigating and improving a COTS-based software development". In *Proceedings of ICSE'00*, pages 32–41, 2000.

[12] Fadrian Sudaman and Christine Mingins. "BiCom: An Evaluation Framework for COTS Components". In *Proceedings of ICCBSS'03*, pages 207–218, 2003.

[13] A. Sutcliffe. "Scenario-based requirements engineering". In *Proceedings of RE'03*, pages 320–329, 2003.