Component-based development
White paper
June 2009

Rational. software

Driving business success with software reuse.

Leveraging an SCCM solution for effective component-based development

Contents

Increasingly, organizations are turning to software reuse initiatives including component-based development to meet market pressures to create higher quality software more quickly.

Executive summary

As enterprise software systems continue to grow in complexity, IT is struggling to find better ways to meet increasing business demands. With pressure to create higher-quality software more quickly and at lower cost, enterprise IT must find ways to streamline development. One way that organizations are meeting this challenge is by implementing ways to drive successful software reuse initiatives. The upside to this approach is considerable: Devising an effective software reuse strategy can help enable software development teams to construct enterprise software systems through the assembly of reusable parts, and component-based development (CBD) serves as a critical enabler of realizing higher degrees of reusability. While CBD is not new, immature technologies and short-term approaches have hindered previous efforts and have soured some on the CBD journey. But that is changing, and many people are looking at CBD with fresh eyes.

Today, proven technologies and tools exist that allow teams to develop, share and manage the evolution of their software assets. With many organizations embarking on large-scale service-oriented architecture (SOA) initiatives, CBD stands to play a critical role in supporting higher degrees of reuse across enterprise services and applications. In fact, CBD is one of the few technologies to have successfully bridged the gap between commercial and open source software development. At present, the open source community thrives on componentization as a major means of achieving high degrees of reusability. Realizing maximum value and return on investment from your enterprise software development initiatives in today's diverse software ecosystem demands an effective reuse strategy that features CBD as its principal driver.

Component-based development and SOA

In many ways, service orientation is an evolution of the primary tenets of component-based development, and a stated SOA goal is to fulfill mission-critical business processes through an orchestration of reusable services.

Highlights

Service-oriented architecture has evolved naturally from CBD in that it strives to fulfill mission-critical business processes by orchestrating reusable services.

SOA helps IT and business align using services tied to specific business functions.

But SOA also represents a significant step forward, in that it offers the possibility of helping IT and business achieve greater alignment through services that expose discrete business functions. While SOA has the potential to increase alignment between IT and business, the services development team faces many of the same challenges that software development teams have dealt with for decades. One critical dilemma lies in preventing problems that arise from the inability to manage change and evolutionary growth. An effective software reuse strategy, achieved through componentization, can provide a significant benefit in realizing your SOA goals.

Let's illustrate this benefit with a simple example. Figure 1 shows a sample use-case diagram for an insurance company's claim management system (CMS). A customer service representative enters the claim upon receiving a notice of loss from the customer. After the claim has been entered into the system, workflow rules dictate to which claim adjustor the claim is routed. After receiving the claim, the adjustor can evaluate the loss, enter the claim into his or her electronic claim file, and eventually process the claim to settlement.

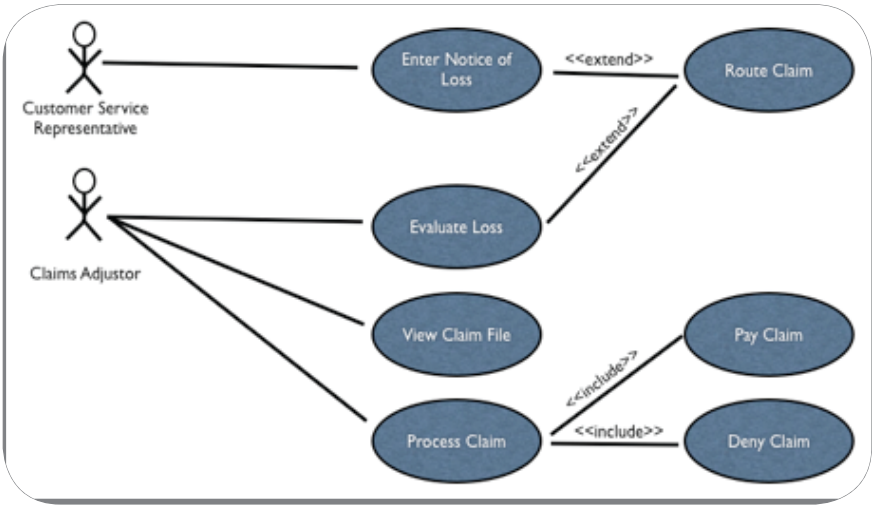

Figure 1: Use-case diagram for a claims processing system.

Highlights

Component reuse can help speed service and application development for multiple development efforts.

Figure 2 shows a component diagram for the claim management system illustrating two separate Web applications and a number of services and components. Each service fulfills a core business function and is composed of one or more reusable components that can be deployed and reused across multiple applications and services. The claim severity component is an example of a component that is deployed and used by both the workflow and adjudication service to help score the severity of the claim and to help ensure that the appropriate rules are applied during adjudication. In this example, component reuse helps speed the development of applications and services through a combination of custom development and component assembly.

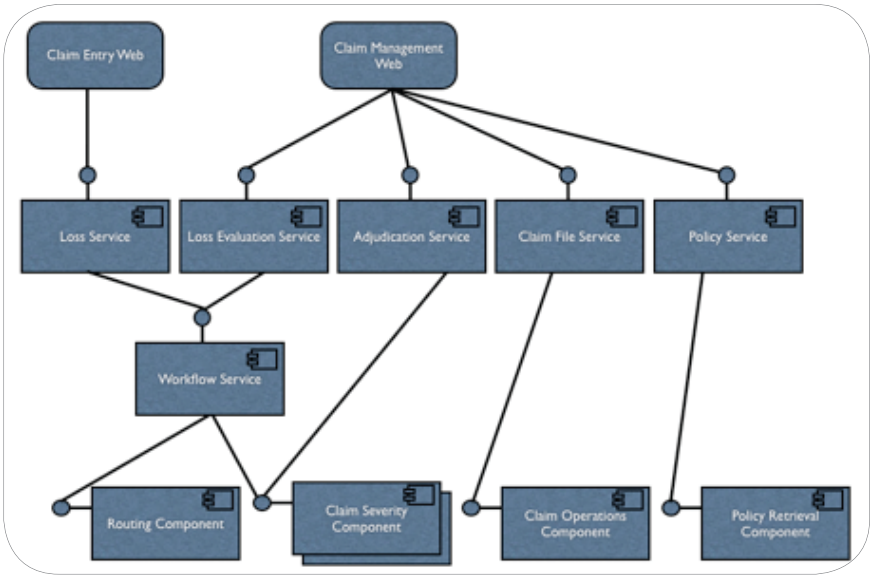

Figure 2: Component diagram illustrating the CMS services and components

Highlights

The inherent complexity of software development can make successfully managing the development of reusable components difficult.

Managing the development of these components is crucial to realizing the advantages of component-based development (CBD) and SOA. Yet managing this development is also difficult, not only because of complex technical challenges, but also due to the essential complexity surrounding software development. Often not considered a significant part of a CBD strategy, a robust software configuration and change management solution is a fundamental aspect to consider for project success. IBM® Rational® Synergy and IBM Rational Change offer solutions for change and configuration management that can help ease your pains surrounding enterprise CBD and SOA initiatives. Before delving into how Rational Synergy and Change software contribute to CBD, let's first examine the benefits of CBD followed by some common impediments.

IBM Rational Synergy and IBM Rational Change can help manage complexities around creating reusable software through their robust change and configuration management functions.

Benefits of software reuse

Software reuse has been considered the "holy grail" of software development for decades. The business value realized through successful reuse initiatives can offer substantial rewards, which explains why so many organizations are pursuing this goal. Below are a few of the significant advantages teams can potentially realize through a successful CBD initiative for achieving reuse:

- *Reduced time to market: CBD emphasizes the assembly of applications from prebuilt parts. Robust components have already undergone rigid testing cycles. Because components are modularized and independent units, they often can be developed in parallel. Given the proper collaboration and testing tools, teams can work effectively even when dispersed across the globe.*
- *Higher quality: Components are discrete, fine-grained units of behavior, and automated tests can be written to help ensure that each component undergoes a strict test lifecycle. As components are field-tested as they are consumed by an increasing number of applications, developers gain increased trust in CBD.*

Highlights

Businesses that successfully reuse software components can realize many potential benefits including increased quality and agility, decreased costs and faster time to market.

- *Decreased cost: Increased component modularity helps isolate software functions, providing an efficient mechanism to upgrade a component without impacting an entire application. Although the short-term cost of CBD is typically greater than the cost of developing applications in silos, as your component reuse increases, the potential benefits of reduced time to market and higher quality may offer substantial long-term savings.*
- *Increased organizational agility: The ability of IT to respond quickly to business needs is imperative in today's dynamic business climate. As the component inventory of your organization grows, teams will be able to enhance application functionality by leveraging prebuilt software components. Additionally, because change requests of the business stakeholders are isolated to discrete units of functionality, change can be made in isolation and released to all component consumers.*

Challenges faced in achieving software reuse

While component technologies have gone mainstream within the enterprise, CBD's benefits have not scaled systematically. Advances in technology are not enough, as teams continue to struggle with the many challenges surrounding management of the CBD process.

Companies that want to reuse software face numerous challenges with configuration, change dependency and version management.

- *Managing versions and upgrades: It is likely that many of your projects use a single component. Unlike reuse in an SOA world, where multiple consumers reuse the same instance of a service, single instances of a component are deployed with an application. When an upgraded version of the component is available, the component development team must determine the best way to make the component available for upgrade, while the teams using the component must determine the best upgrade path. Beyond managing the upgrade process, CBD initiatives must have processes in place to help ensure that component development teams and application development teams receive proper notification when a new version of a component is available.*

- *Component granularity: The behavioral evolution of a component impacts its reusability. As a component is used by more and more applications, there tends to be a push-and-pull effect as requests for expanded and reduced behavior influence the component. Component granularity must be carefully managed to achieve an optimal reuse threshold while ensuring that a valuable level of behavior is provided. Managing change surrounding the behavior of a component is a significant challenge in CBD.*
- *Dependency management: Components with excessive dependencies are more difficult to reuse because of the impact of including the component and each of its dependencies. Attempts to eliminate all component dependencies, however, often result in duplication of code, effectively negating the reuse benefits of CBD. Managing dependencies among components is critical, but understanding component dependencies and the granularity of dependent components is equally important.*
- *Configuration: Ideal components are adaptable to the environment containing them. Failing to create components that are flexible and configurable often means that a component's reuse is limited due to inflexibility. On the other hand, components that integrate easily into your environment are well accepted.*
- *Change management: Change is an inherent part of the software development lifecycle. As change occurs, the team must collect change from a variety of sources and collaborate on the most effective approach to component growth without having a detrimental impact on component reuse. A central repository for managing change with event-based triggers and bidirectional traceability to requirements, design, code and testing helps encourage effective change management.*

Highlights

It's important to have in place a strong foundation of communication and collaboration tools to facilitate component development and to deal with the social and cultural aspects of CBD, in addition to the technological aspects.

- *Communication and collaboration: Publishing a component as an organization's software asset seems like a logical step. However, ensuring that proper communication occurs between component developers and a component consumer is critical as a component evolves. Ensuring that a component repository is available, that your team understands how to obtain a list of published components, and that changes to components are made readily available, is imperative.*
- *Cultural challenges: Many reuse initiatives emphasize CBD's technology aspects but neglect addressing the cultural impact that this may have. Failure to address the people, process and social aspects of CBD can undermine even the most technically adept CBD teams. Since CBD is a reengineering initiative, obtaining strong upper-management support is critical to helping teams work cooperatively and effectively. Reuse initiatives may be disruptive, as key architectural decisions are often split between component developers and component consumers. Successful reuse initiatives require that trust be established between stakeholders and that developers feel empowered.*
- *Reuse without source: The "reuse without source" model of CBD is incredibly difficult because it is based on the assumption that the component will always work within your environment. Unfortunately, because component developers cannot predict all possible usage scenarios, environmental failure is to be expected. Component "reuse with source" is an effective CBD model because teams experiencing component failure can perform trial-and-error debugging techniques in search of the problem. In fact, reuse with source is a significant factor in the continuing success of open source software.*

Highlights

IBM Rational Synergy and IBM Rational Change offer change and configuration management solutions that integrate with other application lifecycle management (ALM) tools to create a robust CBD environment.

IBM Rational Synergy and IBM Rational Change together offer a centralized or distributed component repository, advanced configuration and version tracking capabilities, and robust reporting and security features.

Managing the CBD lifecycle

Overcoming many of the challenges surrounding CBD requires that we effectively manage the CBD lifecycle. Teams that have the discipline and maturity to adhere to a robust CBD process can potentially realize significant value from solutions like IBM Rational Synergy and IBM Rational Change software. These solutions offer software change and configuration management capabilities that integrate with other application lifecycle management (ALM) tools. Together, the Rational solutions provide a robust environment for managing CBD in a complex environment with multiple teams. Each is a first-class construct that supports numerous practices aiding the CBD lifecycle, including:

- *Choice of a centralized or distributed component repository that stores components as source or binary, allowing teams to set up multiple separate repositories and distribute the components across those repositories. IBM Rational Synergy supports high-performance wide area network access to a central server, while one has a topographical choice of setting up multiple, distributed repositories. Because components can be reused as source components, teams have the ability to review the source code when troubleshooting component configurations, and they can make critical changes to the component depending on an organization's CBD processes and guidelines. Because source components can be versioned and reused at many different levels of abstraction, source components can also be easily broken down into finer-grained components to achieve the desired level of component granularity.*
- *Advanced configuration and version tracking that allows teams to compose applications and services from a combination of components. Teams can also easily manage different variations of a component and can perform parallel development on multiple release streams at once. Teams have the ability to search the component repository and subscribe to the components they want to use.*

Highlights

The robust promotion scheme, management console and out-of-the-box support for best practices help make IBM Rational Synergy and IBM Rational Change first-class constructs for supporting the CBD lifecycle.

- *Task-based change and configuration management enables teams to manage coarse-grained change requests that are broken down into discrete tasks. Instead of change being managed at a file level, change is managed at the task level. Once a task is marked complete, all files that have undergone change as part of that task are released back to the repository. Using a shared repository with IBM Rational Change helps to ensure that a single instance of change requests and tasks propagate throughout the CBD environment. Change requests entered into the software by quality assurance engineers or customers can easily be traced through completion.*
- *Insulated environments and a robust promotion scheme, which can help enable teams both to work in isolation during heavy active development and to work together during intense periods of integration. For instance, separate environments can be set up such that a component must pass integration tests before being propagated to the others. Teams can also set up an integration area where the most recent changes to a component are built and tested.*
- *A robust management console that allows individual tasks or entire change requests to be excluded from component versions, branches and builds to help isolate component errors during component development, integration and build.*
- *Support for best practices and patterns for CBD available "out-of-the-box," with significant flexibility for managing variants. With support for a variety of reuse engineering rules, Synergy allows for very flexible component hierarchies that include control over what component is published, identification of all consumers of a given component, and techniques for managing the process of component sharing.*
- *Plug-ins for integrated development environments that allow developers to work within the context of a single consolidated environment.*

Highlights

Companies who can overcome the challenges of creating a software-reuse environment can realize numerous benefits. IBM Rational Synergy and IBM Rational Change can help companies succeed in their CBD initiatives and gain a competitive edge in the global marketplace.

- *Robust reporting capabilities that allow members of the team to clearly understand the evolution of component growth across versions, environments and component consumers.*
- *Robust security that supports the separation of roles between component developers and component consumers.*

Conclusion

Software reuse and its underlying fundamentals have evolved dramatically over the past four decades. The benefits of constructing mission-critical enterprise software applications through the assembly of reusable parts are immense. With today's complex development environments and ambitious SOA initiatives, effective CBD can play a critical role in realizing success. While component technologies have gone mainstream, many other factors are impeding CBD success. One of the biggest reasons is the inability of development teams to develop, share and manage components through an underlying framework of integrated and optimized process patterns and change management techniques.

The Rational Synergy and Rational Change solutions can offer mature teams a significant advantage with CBD initiatives. The robust features combined with strong integration capabilities offer a disciplined approach to CBD. The two solutions can help you reduce time to market, improve quality and accelerate the delivery of advanced software and systems by supporting best practices.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
June 2009

RAW14082-USEN-01