

# Create collaborative and dynamic method content using Web 2.0

Skill Level: Intermediate

[John E. Boyer \(john.boyer@us.ibm.com\)](mailto:john.boyer@us.ibm.com)  
SOA Program Manager  
IBM

[Bertrand Portier \(bportier@ca.ibm.com\)](mailto:bportier@ca.ibm.com)  
IT Architect  
IBM

[Dr. Eoin Lane \(eoinlane@us.ibm.com\)](mailto:eoinlane@us.ibm.com)  
Senior Solution Engineer  
IBM

17 Apr 2008

Leverage Web 2.0 technologies to extend software development process content, which is typically published static as HTML. This article describes how you can develop the ability to collaboratively edit method content and have access to the latest dynamic content within a method context.

## Introduction

IT practitioners commonly use software development methodologies, such as the [IBM® Rational® Unified Process \(RUP®\)](#). Methods like this can be applied across a variety of software development disciplines and industry verticals. Software development methods like RUP and [IBM RUP Service-Oriented Modeling and Architecture \(SOMA\)](#) provide static process guidance that's published as HTML. [IBM Rational Method Composer](#) is a tool that process engineers can use to customize a process; however, you can publish the new process as just read-only Web pages.

For the method to be truly useful it needs to be augmented with context-specific assets. These are generally content, tooling, and people assets. The content assets

include a range of resources, such as documents, presentations, models, social bookmarks, and others. For example, if this method is being applied to aid in business modeling in a telephone company (telco) vertical account, then the method should provide guidance around specific tooling and content that can be leveraged.

Because the method content is frozen after being published, it's not extensible. Therefore, you can leverage Web 2.0 technologies to augment the static content with supplemental *wiki* pages that enable collaborative editing and dynamic Web feeds. These pages are referred to as *extension points* in the next section.

So why is the lack of method extensibility a problem? Because method contents:

- Become outdated; for example, guidance artifacts such as templates, assets, or tool mentors become obsolete.
- Lack specific context details without being customized. For example, due to the nonspecific nature of the off-the-shelf method content, it needs to be adapted for the situation in which it's being applied, such as the organization, industry domain, roles, activities, assets, and tools.
- Customizations require republishing.
- Lack the ability to be augmented by the user or practitioner and, thus, can't take advantage of the user's contribution to community content development and enhancement. Consequently, opportunities for feedback and collaboration by the user community is often lost.
- Lack the ability to leverage media-rich content, such as videos, podcasts, and flash demonstrations because they typically become outdated quickly.
- Tend to lack detailed commercial off-the-shelf tool guidance.
- Off-the-shelf, lack guidance for organizational assets or tools.

Another problem is that process engineering departments don't have enough resources to produce all of the method content needed in the field. For example, they usually can't provide content for different versions of the same tool. Hence, the method content remains unchanged, and the organization loses the collective knowledge and understanding of its practitioners who can revise the content based on real-world field knowledge.

**Note:** Some of the content is often priority to an organization; an example is the Insurance Application Architecture (IAA) model, which is proprietary to a company like IBM.

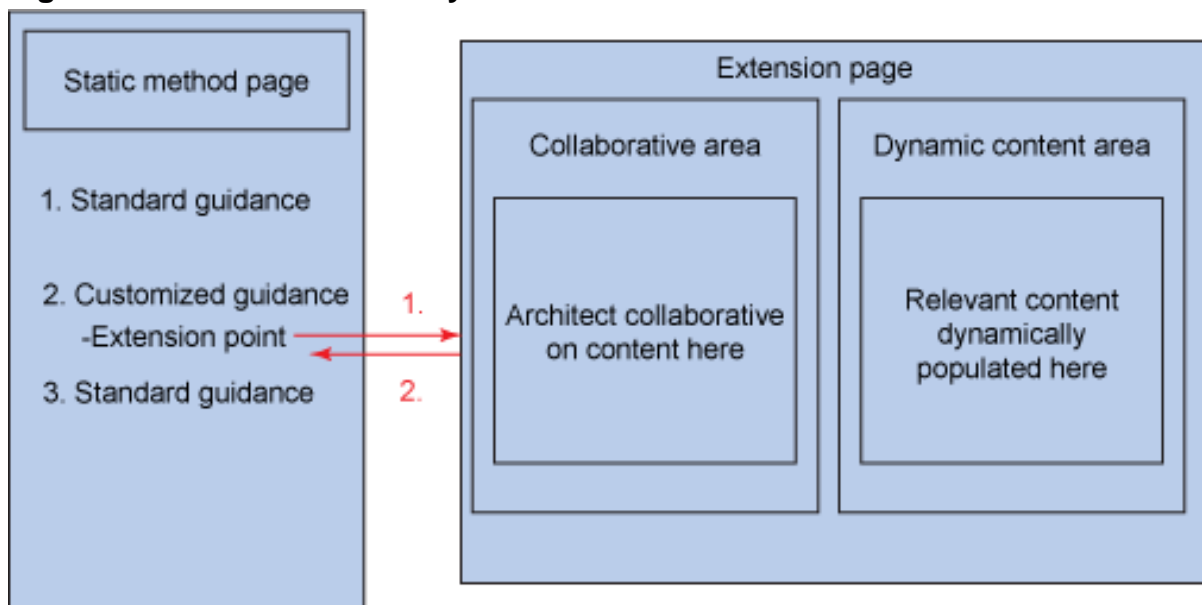
Other techniques have been used to solve the same problem. Consider the following: An administrator or process engineer can republish the static method on a

regular basis (for example, monthly, weekly, daily). However, a process needs to be established to incorporate practitioner feedback or contributions into the method. Also, the problem is aggravated by the fact that the method is published along with a tool, such as [IBM Rational Software Architect](#) and doesn't get updated until Rational Software Architect gets updated. Practitioners build their own pages with up-to-date information, but these pages are scattered and not integrated with method and process contents.

## Create extensible method content with Web 2.0

Ideally, you want to be able to extend software development processes and methods with extensions built collaboratively, and dynamically populated at the time practitioners use the method. Figure 1 shows you what this looks like.

**Figure 1. Collaborative and dynamic method overview**



There are several activities associated with this collaborative and dynamic content. Let's take a closer look at these activities.

### Extension point identification

The process engineer identifies extension points in a static method. Typically these extension points are in areas of the method that describe new or improved techniques, or both, and in areas for which content needs to be built with the help of community or will quickly become out of date.

### Extension page creation

After an extension point has been identified, the process engineer creates an

extension page for this extension point. The purpose of this extension page is to provide up-to-date guidance in addition to the contents of the static method. The extension page contains two areas:

- Collaborative guidance content area
- Dynamic content area

### **Collaborative guidance area**

The collaborative guidance content area provides up-to-date guidance on the method for this extension point. Typically the initial content in this page is populated by another process engineer skilled in the art of this particular extension point. The content in this area can be, for example, the latest information on tools and how to obtain them, and then used to execute this extension point more effectively. This collaborative area is also editable by the user (practitioner or architect) to allow for field-based lessons and input to be captured, thus keeping the best practices and field-based lessons learned about this extension point up to date. An example of a Web 2.0 implementation of this collaborative guidance area is a wiki.

### **Dynamic content area**

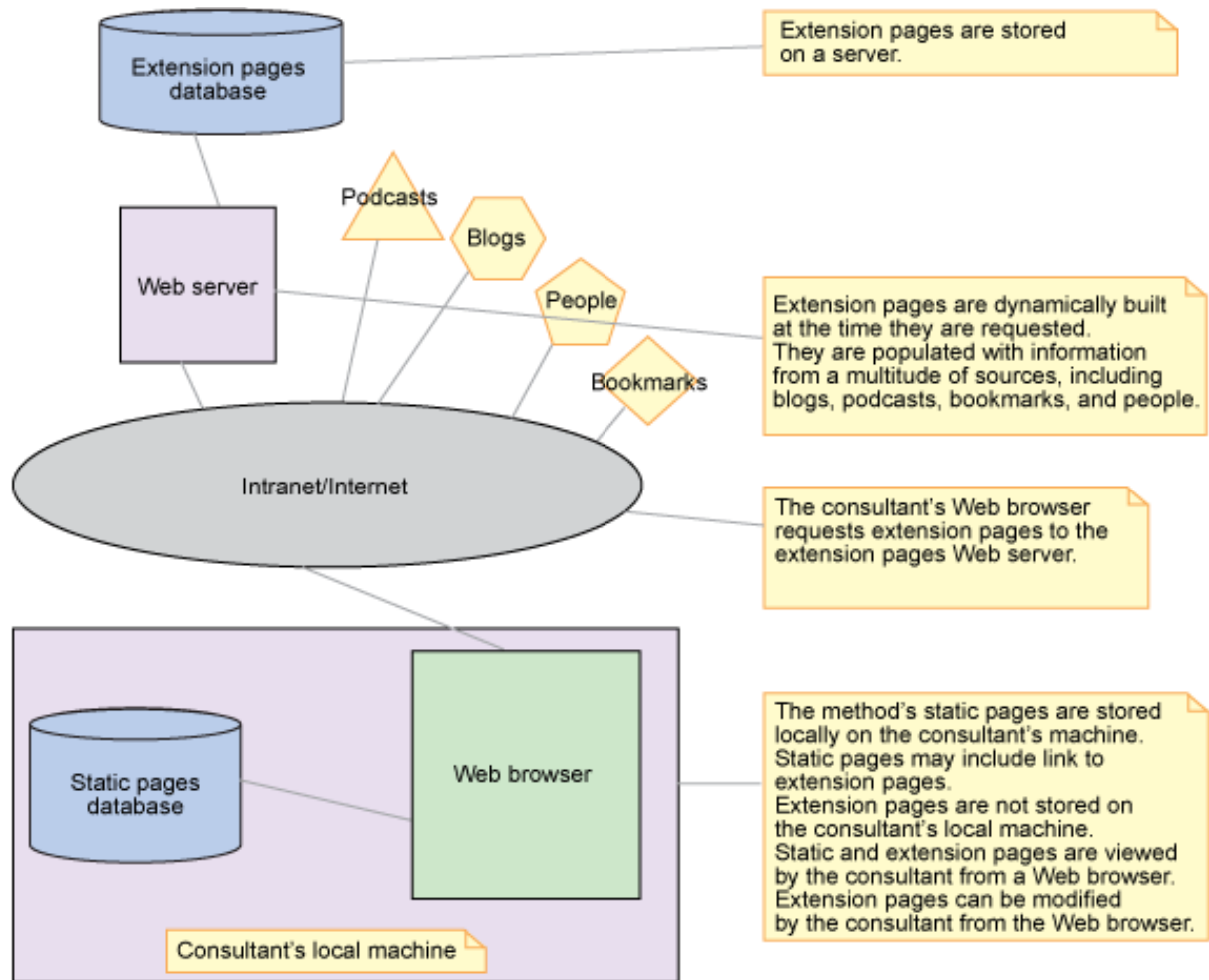
The dynamic content area dynamically provides assets and artifacts to the user or practitioner to help him or her execute the task described by this extension point. The kinds of assets and artifacts can include social bookmarks; subject matter experts (including their instant messaging statuses); documents; publications; presentations; media-rich syndicated content, such as podcasts and movies; and education materials, including blogs, online course material, and classes. Because the information in this area is dynamic, the system builds content at the time the practitioner requests the page so that he or she is always guaranteed of the latest information. An example of a Web 2.0 implementation of this dynamic content area is aggregated Web feeds.

The static method process engineer repeats the creation of extension pages for each extension point identified in the static method.

By adopting this technique for creating a method, the dynamic content is automatically built when a user requests a specific page and includes information from many different sources outside of the core method contents. This dynamic content can be provided by practitioners, not just method authors (process engineers). This dynamic content is accessible from the method, not scattered over the Internet or intranet.

Figure 2 illustrates where method contents are located (topology view). It's best to read the comments from bottom to top, starting from the consultant's local machine.

### **Figure 2. Collaborative and dynamic method structure**



The advantages of adopting this kind of approach to software method development are:

- Method content isn't restricted to what's being shipped by a method at a specific version.
- Method contents are always up to date.
- Practitioners, not just process engineers, can provide contents for the method.
- Practitioners don't need to download new versions of a frozen method.
- Contents are no longer restricted to what an engineering department can produce given their time, budget, and headcount constraints.
- Media-rich and innovative contents (for example, list of experts on a topic, podcasts, and flash movies) are easily integrated into method contents.
- Methods can contain both frozen (static) core contents and spots where

organizations or communities can extend and customize contents.

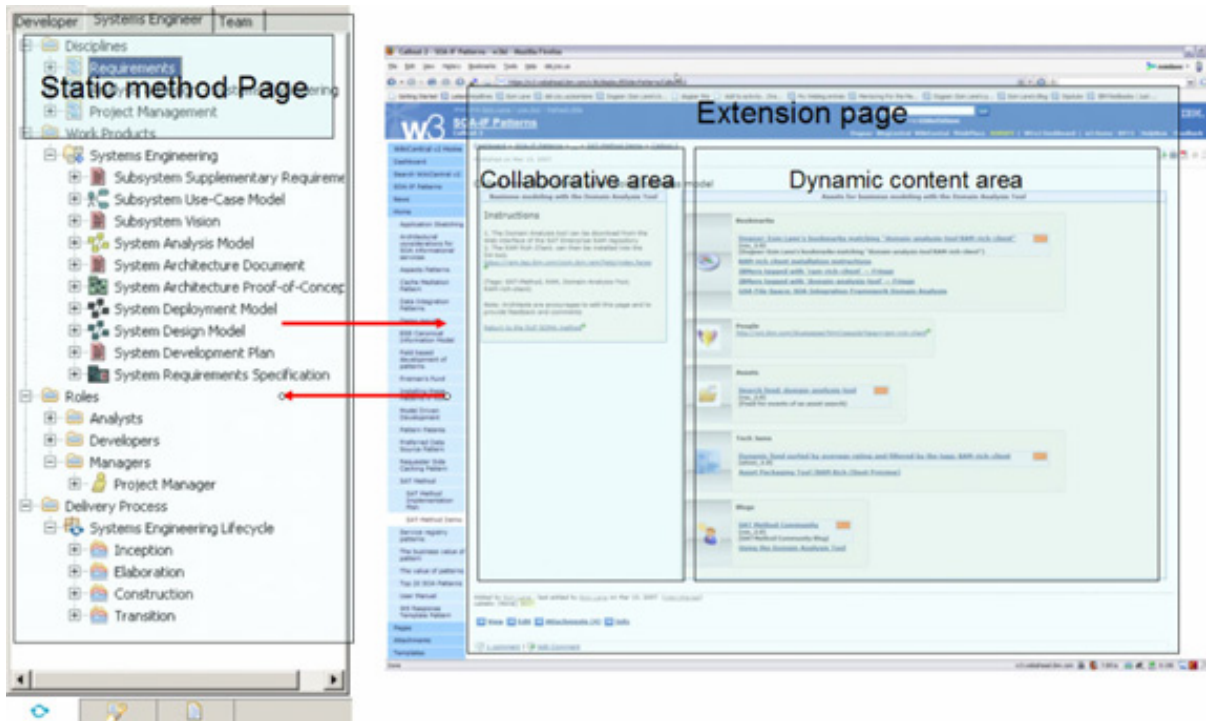
- Practitioners can easily find experts on a given topic.

An implementation of this kind of method:

- Identifies the points in the static method that can be extended.
- Provides links to collaboratively built dynamic contents from these extension points.
- For each link, provides a wiki page with two areas:
  - Up-to-date editable textual information
  - Dynamically populated Web feeds on social bookmarks, people, activities, blogs, or assets

Here's how the method works and is implemented for a particular Service-Oriented Architecture (SOA) analysis scenario (see Figure 3):

**Figure 3. Collaborative and dynamic method implementation**



An example of how it can be used is as follows: A software architect in an SOA engagement is currently involved in the analysis of an SOA system. In SOA, one of the core activities is called service identification. Because SOA projects are by their very nature complex, and because SOA is still in the process of being understood, SOA activities and the tools to support these activities are fully understood only by a small number of practitioners.

It's highly unlikely that a static method includes up-to-date content on the best practices around these activities or, indeed, the tooling that could make these activities more streamlined. To address this shortcoming, extension points are identified in the base method. Where the static method is rendered in HTML, these extension points are hyperlinks from inside the method. In this example, the SOA architect is doing service identification. The static method contains high-level guidance around service identification and a link to collaboratively built dynamic content on specific tools used for service identification. This link brings the architect to a collaborative Web site, such as a wiki dedicated to this extension point.

## Wiki implementation

This wiki site contains the following two areas.

### Collaborative content area

The first area contains current information on service identification tool offerings to make the activity more consistent and streamlined. Because this is a collaborative area, the SOA architect is encouraged to edit these instructions to keep them up to date and to provide the latest field-based development thinking around these tools.

### Dynamic content area

The second area of the wiki Web site provides dynamic content relevant to the architect for this particular extension point in the form of Web feeds. In the case of the service identification extension, the area is populated by dynamic content around service identification. This dynamic information is filtered based on a set of unique tags associated with this particular extension point (for example, tags for a service modeling activity might be: `services modeling` and `service-modeling`). Such syndicated dynamic content includes:

- Social bookmarks and subject matter experts of service identification (including their instant messaging status).
- Reusable assets from an asset repository (including documentation patterns and even tooling for services identification).
- Media-rich content (including technical presentations, movies, or podcasts).
- Educational content (including blogs, course material, other reading material, and IBM Redbooks®).
- Activities that the architect needs to follow to complete the extension point around service identification successfully.

This is made possible by being able to embed all of this dynamic content in a wiki

with syndicated Web feeds formatted as RSS or Atom. Each of the dynamic items above has its own Web feed, and these feeds are aggregated to provide all of the dynamic content. This is already possible, and by standardizing on Web 2.0 technology and tags, for example, all service identification-related content checked into an asset repository are tags with the service identification and dynamic-method keyword. After an item is checked into the Web feed-enabled asset repository with these keywords, the Web feed provided by the asset repository is automatically updated. As the extension page is refreshed for the service identification extension point, the updated content is now available to the method practitioner (architect). This can also be done with all of the content for the other dynamic feeds.

## Conclusion

This article demonstrated how to build collaborative and dynamic method content using Web 2.0 technology. This leverages the idea of sets of tags specific to an extension point of a method to provide dynamic content-to-context mapping.

# Resources

## Learn

- Read the article "[What is Web 2.0?](#)" by Tim O'Reilly.
- Visit the [IBM Web 2.0 home page](#).
- Check out the [IBM Rational Method Composer home page](#).
- The [SOA and Web services zone](#) on IBM developerWorks hosts hundreds of informative articles and introductory, intermediate, and advanced tutorials on how to develop Web services applications.
- Play in the [IBM SOA Sandbox!](#) Increase your SOA skills through practical, hands-on experience with the IBM SOA entry points.
- The [IBM SOA Web site](#) offers an overview of SOA and how IBM can help you get there.
- Stay current with [developerWorks technical events and webcasts](#).
- Browse for books on these and other technical topics at the [Safari bookstore](#).
- Check out a quick [Web services on demand demo](#).

## Get products and technologies

- Download [IBM Rational Method Composer](#).
- Innovate your next development project with [IBM trial software](#), available for download or on DVD.

## Discuss

- [Participate in the discussion forum for this content](#).
- Get involved in the developerWorks community by participating in [developerWorks blogs](#), including Eoin Lane's blog [Building SOA applications with reusable assets](#).

# About the authors

John E. Boyer

John Boyer is an SOA program manager for IBM Software Group. He has extensive experience designing and developing Java, J2EE, and C++ systems. He's currently focused on integrating social software into software development and learning activities.

---

### Bertrand Portier

Bertrand Portier is an IT architect with SOA Advanced Technologies, IBM Software Group. He works in the field on strategic SOA transformation projects and, based on these experiences, works with IBM Software Group development teams. His background is in J2EE and Web services, and he is now heavily involved with asset-based and model-driven development.

---

### Dr. Eoin Lane

Dr. Eoin Lane, senior solution engineer, is the lead for harvesting and developing of application pattern from key IBM SOA engagements and driving those patterns through the IBM pattern governance process to accelerate adoption. Eoin also specializes in model-driven development, asset-based development, and Reusable Asset Specification (RAS) to facilitate SOA development.

## Trademarks

IBM, the IBM logo, Rational, Redbooks, and RUP are registered trademarks of IBM in the United States, other countries or both.