Designing a Web Services Project for Maximum Value: The 90 Day Challenge

Katherine Radeka
Hewlett-Packard Company
18110 SE 34th Street
Vancouver, WA 98683
360-212-0375
katherine_radeka@hp.com

ABSTRACT
The 90 Day Challenge team set out to deliver an end-to-end Web services solution to enterprise sales agents in 90 days, with immediate plans to extend the solution to other user groups. In order to meet their commitments, the team had to reuse the same component Web services in multiple contexts. By modeling the workflows and carefully managing scope at the component level, the team developed a suite of Web services that other teams could easily incorporated into multiple projects. As a result, twelve other programs to date have incorporated Web services from the 90 Day Challenge, saving weeks of development time.

Categories and Subject Descriptors
D.2.2 [Software Engineering]: Design Tools and Techniques D.2.13 [Software Engineering]: Reusable Software---reuse models

General Terms
Management, Design

Keywords
Web services, Services Oriented Computing

1. INTRODUCTION
Services oriented computing has received a great deal of press in the past two years as a new way of thinking about delivering value over the Internet. The basic idea is to encapsulate core functionality into a series of services that contract with each other to provide value to the end user. These services interact with each other using standards-based interfaces derived from XML, enabling them to operate transparently between a diverse set of platforms and across organizational boundaries.

In theory, this makes such tasks as enterprise application integration much easier than with older technologies because open standards allow for open communications. However, a team can implement Web services in the same way as legacy applications, thus making them just as brittle and unresponsive as these older systems. The promise of web services technology can only be realized by using best practices for modeling the desired service, and developing a common architecture for these services.

Last summer, I led a team to develop a Web services based end-to-end application within 90 days. Along the way, we discovered that scenarios, business process modeling, carefully scoped components, a flexible architecture and standards-based interfaces are all critical elements of a successful Web services project.

2. WEB SERVICES DEFINITION
Web services can be defined in a number of ways. For the purposes of our project, we define Web services as components that used standard Internet protocols, such as HTTP Post, and an XML based data exchange format to enable inter-component communication. Each Web service performs one step of a business process very well.

We took the definition one step further. We had a compelling need, from the beginning, to develop a suite of services to be shared among different sets of users. These component services could reside on different systems, even different platforms. An aggregator service was responsible for coordinating the inter-component communications.

Although our organization did not have an internal implementation of SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) or UDDI (Universal Description, Discovery and Integration) we kept these standards in mind as we designed the interfaces for the service components. Plans were underway to support these standards, which will play a key role in enabling Web services to coordinate communication across platforms and organizations.

3. PROJECT BACKGROUND
Our specific task was to develop a service for our company's external sales agents that would enable them to use wireless devices to access all the information needed to prepare for a meeting with a potential customer. They also needed the ability to send content to a nearby printer or fax machine, especially those reports which would not display on the small screen of a cell phone or a Personal Digital Assistant.

The service needed to retrieve information from our company's customer database, financial database, news services and stock

price services, to be aggregated into a single report in a format that would translate well onto a wireless device. The service also had to provide the ability to take action on the items in the report, such as send voicemails or emails to colleagues, directly from the service.

Finally, if successful, the intention was to immediately leverage the services to provide a similar solution to the company's financial analysts and line of business managers, with similarly aggressive schedules.

4. SHARED SERVICES

Since we had a very tight schedule and we knew that the work would be leveraged in future projects where time to market would have similar importance, we placed a major emphasis on developing services that could be shared.

The promise of Web services technology is that it solves a lot of the problems with reusing code libraries across an organization. For example, if DLLs and code libraries are reused, it is easy for multiple versions to spring up across an organization which are not compatible, making it difficult for the organization to roll out changes, enhancements or fixes. There is also the issue of trust, and 'not-invented-here' which any group that wants to encourage reuse must overcome. Finally, the development team has no incentive to create objects for reuse since there is no immediate benefit to the team.

Web services address the first issue by making the service available to external teams, but not the code itself. The service runs in an environment which is under the control of the original development team.

The second issue has been addressed in our organization by Service Level Agreements between organizations that specify what the service will do, its availability and load capacity, an escalation path to report problems and a billing process.

Teams have incentives to build services for reuse since they are measured, in part, on how many users are accessing the service. If they make their services work in multiple contexts, they increase their usage numbers. If they have a billing model in place, their organization receives revenues from the service. They also shorten their own time-to-market in later applications.

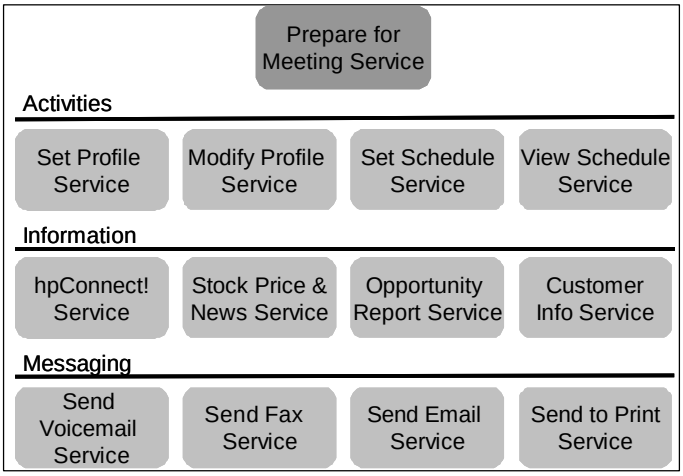
5. SCOPING SHAREABLE SERVICES

5.1 The Issue of Scope

The issue of scope is one of the biggest challenges of reuse which has yet to be solved, however. A component service will only be used by other groups if it does what the groups need the service to do, exactly as they need it to be done. Therefore, a critical issue in developing shared services is deciding how to scope them so that the target developer communities have their needs met. The question about how to scope any individual service is a tricky one – there is definitely overhead involved in processing all of the XML documents that are flying around if the services cannot do enough on their own. But scope the services too large and they lose generic applicability.

We found that it was easiest to think in terms of small chunks of business logic – enough to perform one logical step in the process. With this granularity, we created component services that could be combined and recombined at will to produce new aggregated services.

We found that the team needed to understand the multiple contexts within which an individual service might be used. Otherwise, there was a significant risk of developing a service that was too specialized within its initial context to perform the task in a different one. We needed a detailed understanding of the business processes of our various user communities to write the reusable component services.. Our team defined twelve different component Web services to perform the tasks that our first project required, with one aggregator function that tied them all together (Figure 1).



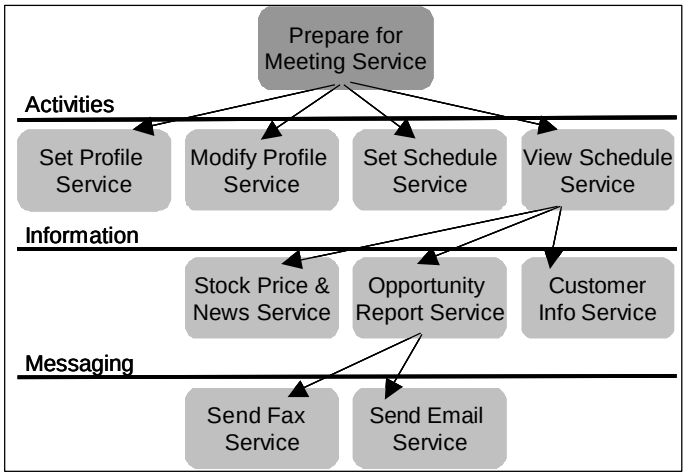


Figure 2 shows the 90 Day Challenge component services that are required to deliver the Prepare for Meeting Service. We created diagrams like this for each of the major pieces of the 90 Day Challenge to understand which of our services would be needed for each problem domain.

We were fortunate in that we already had three potential target audiences for our services. This helped us understand which services would need to be common for all, and which ones really needed to be customized. However, we were faced with a relentless deadline, so any process we followed had to be lightweight and flexible. For help in analyzing this problem, we turned to scenarios and business process modeling.

5.2 Scenarios

In order to understand what our business process would be, we wrote several scenarios that described what we wanted the user to experience. We wrote the scenarios as stories, in simple paragraphs, which discouraged us from taking up valuable time to create nice diagrams when a sentence or two would do. For example, here is the scenario that the Prepare for Meeting service is designed to fulfill:

Prepare for Meeting Scenario: Susan, an enterprise sales representative, is on her way to a meeting with her most important client. She arrives at her appointment a few minutes early, time enough to get an update on some critical information about the status of the customer's latest requests. She views the order status information, outstanding deals, the customer's stock price and the latest news about the company. She prints out portions of the report to show to her custome, by sending it to a nearby fax machine. She also forwards a copy of a report to a colleague using email.

The scenarios helped us get inside our users' heads, to understand what was likely to be most important to them. For the example scenario, speed was paramount – the sales rep just had time for a quick last-minute check to see if there was any news.

One advantage of writing scenarios in this informal style is that we could churn out a lot of them in a short period of time. This helped us to identify the touchpoints where services could be shared across problem domains without impacting our schedule.

5.3 Business Process Modeling

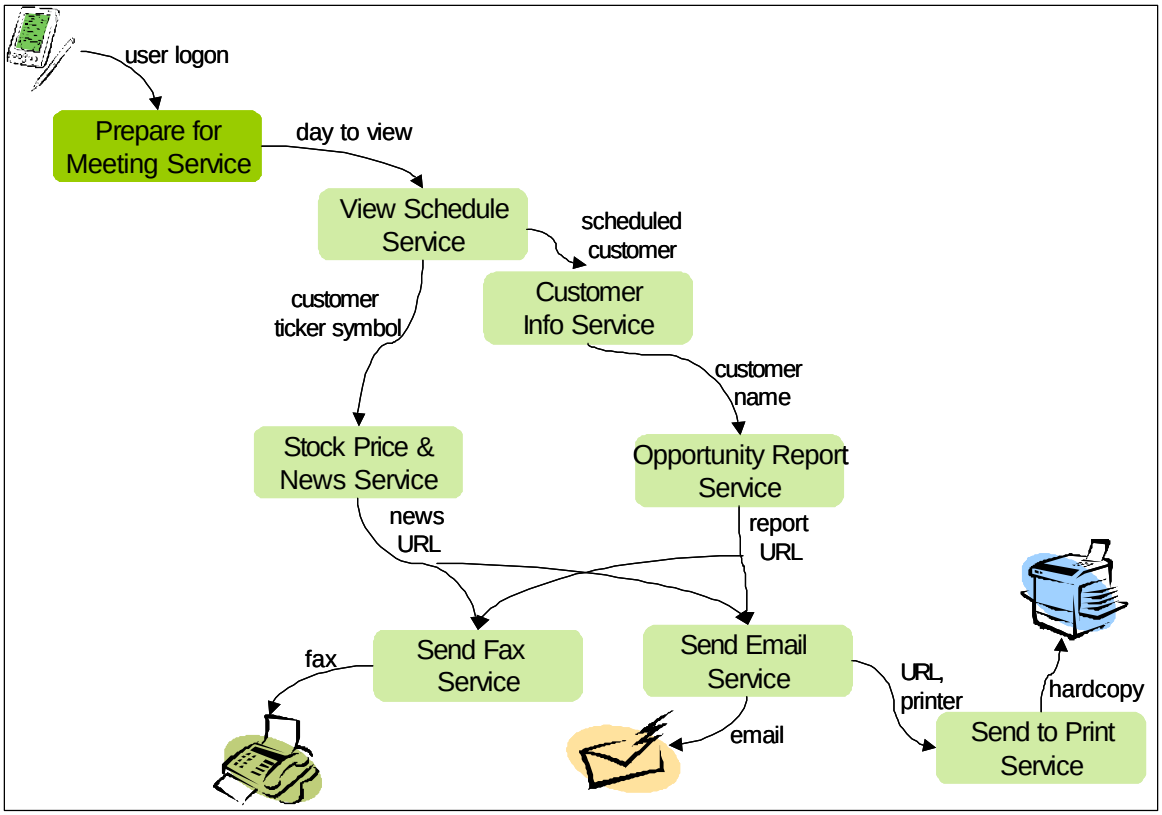
We used the scenarios to develop high level business process models. The models were simply drawings with a few basic conventions that helped the team understand the workflow. We chose not to use any formal methodology, such as IDEF0 or UML (Unified Modeling Language) because the diagrams were primarily for the benefit of the team, and they were not trained in those methods. We also did not want the team to get lost in the "method" and lose focus on the problem at hand.

To help this along, we set a strict time limit for creating each business model, and did most of the work on printable whiteboards which would be translated into simple diagrams. Our business environment was in a state of rapid change, so it was more important for us to be fast, and to create a drawing style that could be rapidly updated, than it was to be 100% accurate.

These process flows helped us describe the steps we would need to go through to deliver the service to the end user. We began with fairly low levels of detail in the models – about twenty steps to get through the example scenario above.

We then manipulated the levels of detail shown in the models to maximize the number of process steps that were reused across diagrams. That helped us identify opportunities for services that could be shared.

As we defined the services, we replaced the business process steps with the names of the services that would fulfill them.



Finally, we completed the drawings with the information that would need to flow from service to service. This gave us the basis for beginning to write the interfaces. The end result of the process is shown as Figure 3.

5.4 Early Interface Documentation

The next step in the process was to design the XML interfaces between the services. This would enable the teams to develop the actual services more or less independently. We were even able to integrate very quickly with an XML interface that had already been developed for accessing the customer and sales databases. This gave us an immediate two-week jump in our schedule for the Web services that used data from those systems. This two week savings enabled us to deliver other high priority features without impacting the schedule. Following our model, we kept our interface definition documents very simple – just example XML documents with a few lines text at the bottom to define each parameter.

5.5 Agility in Response to Change

By keeping the overall process lightweight and the documentation simple, we were able to ensure that our models kept sync with reality in the face of a lot of change.

About midway through the project, one of our major user groups was moved into a different organization which would require different information sources for some of the data in a key report. Since our models were easily updated, we were able to update them quickly and to come up with a solution to the new problem.

That process continues today. The original services are still in use, although some of them have been migrated to new platforms and others have been extended to deal with additional problem domains. As the services have evolved, it has been easy to evolve the documentation with it, which enables us to make quick decisions when we are faced with a new business need.

6. RESULTS

The services developed by the 90 Day Challenge are still in use today, and the messaging services have been applied in at least ten separate projects.

We delivered our original project on time, and we have impressed other business teams with our ability to recombine our services to meet unanticipated needs.

We were able to deliver more in our 90 day time frame than we would have been able to deliver had we not used shared services, and we were able to start delivering value as soon as the most important part of the application, the Prepare for Meeting service, was ready – two weeks earlier than scheduled.

The core Web services that the 90-Day Challenge team developed have continued to evolve In addition to on-going development to enhance the original top-level service, the component Web services have been incorporated into ten distinct projects for the other frequent users of the customer and sales databases.

7. CONCLUSION

Web services development teams can only realize the benefits of the technology – agility and speed – if they are able to define the scope of each component so that other groups are able and willing to use it in their own services.

By using a combination of quick scenarios, business process modeling and standards-based interfaces to create components that were scoped to deliver maximum value, our team was able to demonstrate that the power of Web services to create an agile IT applications infrastructure is real.

8. ACKNOWLEDGMENTS

Our thanks to Lougie Anderson for her editing assistance with this paper.

Designing a Web Services Project for Maximum Value: The 90 Day Challenge

Katherine Radeka

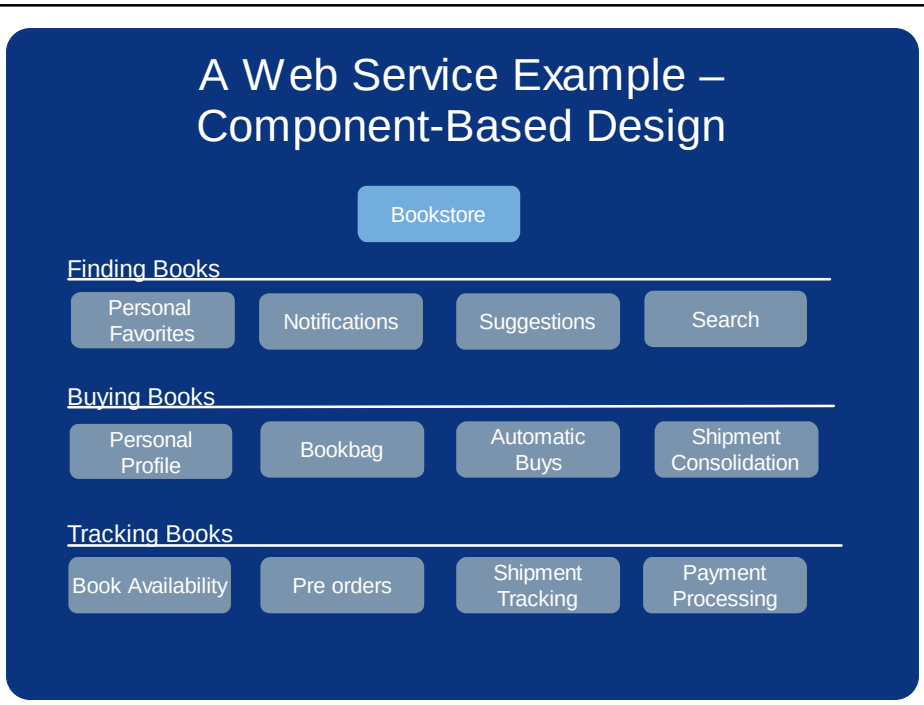
Hewlett-Packard Company

November 8, 2002

Topics

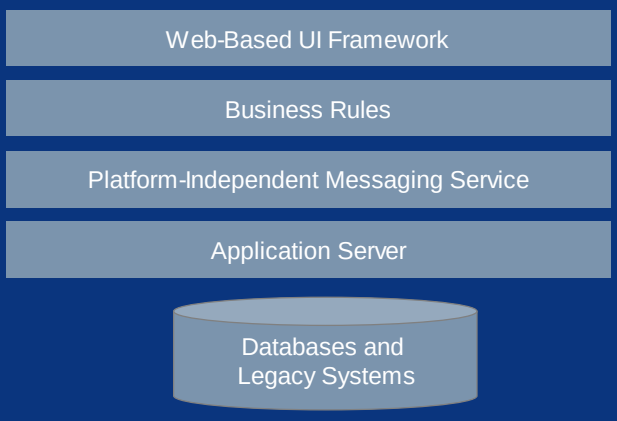
- What is a Web Service?
- Project Background
- Issues with Web Services
- Scenarios
- Business Process Modeling
- Early Interface Documentation
- Results

A Web Service Example – Component-Based Design



page 3

A Web Service Example - Architecture



page 4

Web Service Standards and Common Platforms

Protocols: XML, HTTP-POST

Dynamic Discovery: UDDI

Description: WSDL

Information Exchange: SOAP

Platforms: J2EE, .Net

page 5

Our Mission

- Develop a suite of services for external sales agents to access all the information needed to prepare for a meeting with a potential customer.
- Deliver the content using mobile devices.
- Complete the project in 90 days.
- Immediately leverage the services on other projects

page 6

Our Strategy: Build a suite of shared services

Opportunities

- Display output to a variety of devices from one set of XML docs.
- Enable rapid leveraging to other projects.

Challenges

- How do we scope a service so that it can be shared?
- How do we manage co-development of inter-related services?

page 7

Three Techniques to Address the Challenges

- Scenarios
- Business Process Modeling
- Early Interface Definitions

page 8

Scenarios

Short, one page descriptions of situations where our service would be used.

- Quick to write, quick to read and review.
- Described multiple situations where our services would come into play.
- Focused the team on the real-world experience of the people who would use the services.

page 9

Example Scenario:

Prepare for Meeting Scenario: Susan, an enterprise sales representative, is on her way to a meeting with her most important client. She arrives at her appointment a few minutes early, time enough to get an update on some critical information about the status of the customer's latest requests. She views the order status information, outstanding deals, the customer's stock price and the latest news about the company. She prints out portions of the report to show to her custome, by sending it to a nearby fax machine. She also forwards a copy of a report to a colleague using email.

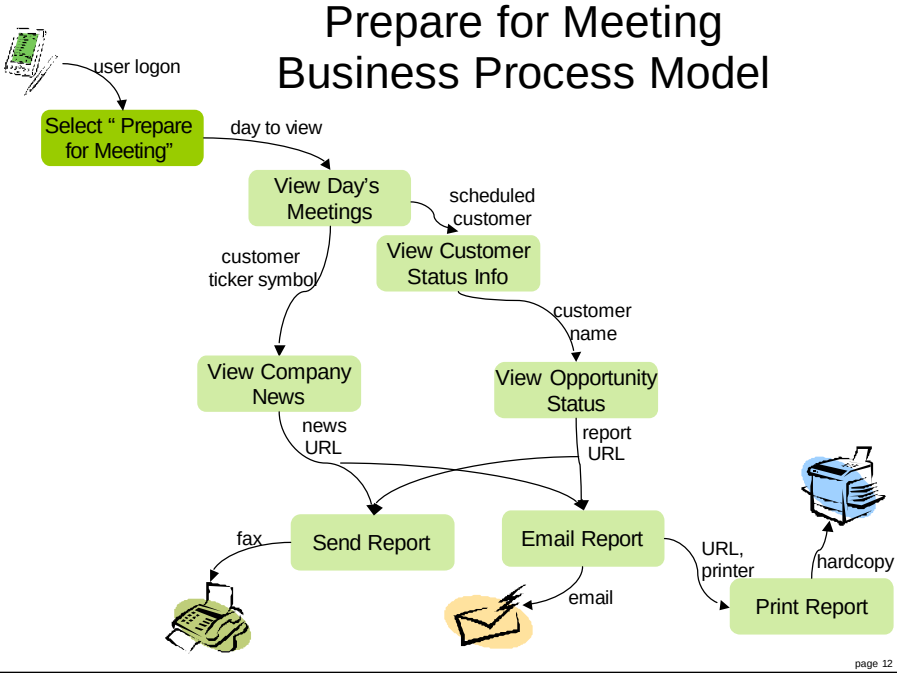
page 10

Business Process Modeling

- Each scenario became a simple business process model.
- Each model identified natural candidates for shared services.
- Manipulating levels of detail led to models of how services would interact in the scenarios.

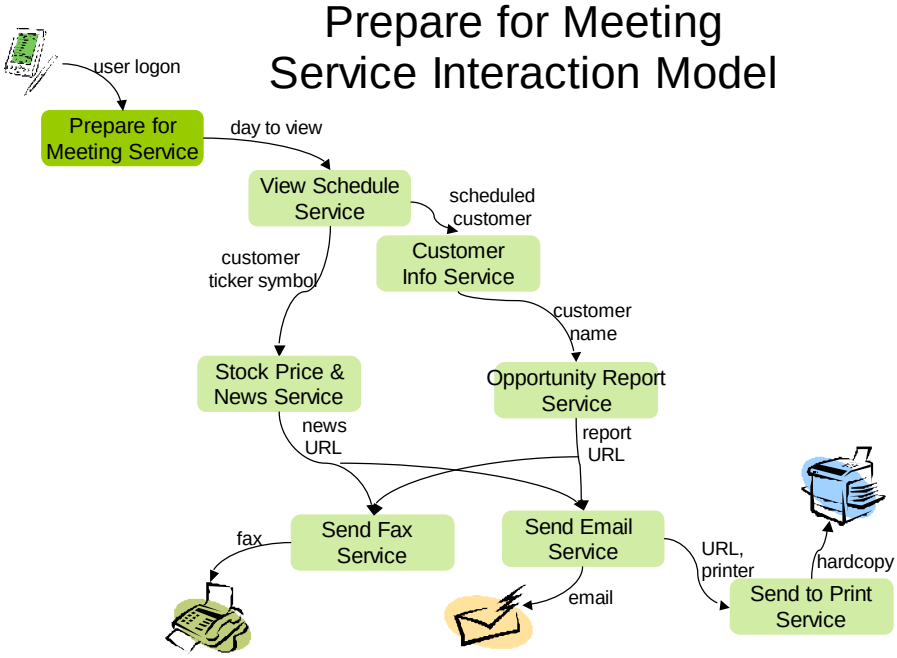
page 11

Prepare for Meeting Business Process Model

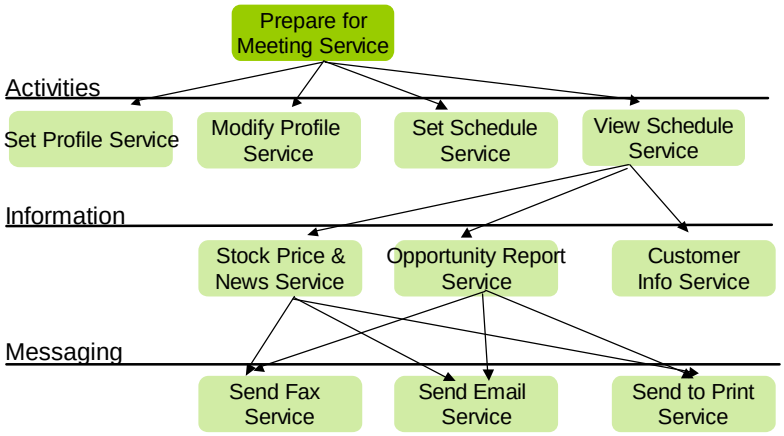


page 12

Prepare for Meeting Service Interaction Model



Prepare for Meeting Services



Early Interface Definition

Finally, we defined the interfaces for each service before development began on any of them.

- Validated the service interaction model.
- Minimized dependencies between developers working on interrelated services.
- Enabled us to publish the interfaces to other teams who would consume the services so that we did not hold them up.

page 15

Results

- We met our deadline.
- We delivered more value than anticipated, by combining the Web services into creative combinations.
- The Web services have been leveraged into more than a dozen projects in the last year.

page 16

Q&A

hp invent