Creating anything completely from scratch is always a difficult proposition: you can't benefit from something that's been created before; there's no "platform" on which to build; and there's nothing available to use as a model or blueprint. So it is when creating e-business Web applications. The first time a company constructs the type of application that's meant to run solely across the Internet, it may draw on experiences gathered on a different network platform, but the business ramifications of always being live and not being able to predict the volume of user traffic make this a new situation.

A portal — a sort of information clearing-house for a specific collection of Web users having similar needs or interests — represents just such a new situation. A company building and operating this type of software product faces a project that has no clear sponsor to resolve disputes about requirements. This leads to a challenging requirements definition process, which is important for appropriately narrowing the scope of the portal and staying focused on the original business objectives. Using a requirements framework can help.

A requirements framework enables the same set of requirements to be used for different Web application projects. Using such a framework can significantly reduce the amount of time needed for the initial elicitation of requirements and subsequent enhancements to them. In this article, I'll discuss the requirements that emerged from the development of a local community portal and describe how that requirements baseline constitutes a framework for additional Web application development projects.

Whereas this article is based on actual experiences in constructing a requirements framework, other Rational Developer Network articles on this subject take a slightly different approach. The other articles (listed later under "Related Resources") discuss frameworks from the standpoint of use-case models and software architectures.

The Business Vision

A project I was recently involved in focused on creating an online community portal. This portal was to provide access to local business directories, include multiple content feeds containing local news and information, and give local shops a chance to build inexpensive online stores where customers could purchase various items.

Because this was a new business activity, there was no legacy application to turn to for initial system requirements, UML models, or test plans; everything had to be done from scratch. The business plan emphasized building an initial community and then reusing that existing code base to deploy the resulting "framework" into a second and subsequent communities. This effort involved a number of activities: site design, user interface prototype definitions, development, and finally testing and deployment.

Rollout of the initial portal was critical to the success of the overall plan. It was essential that the systems engineering team develop a set of reusable artifacts that would greatly shorten the development cycle of subsequent portals. The initial artifact that was targeted for reuse was the Project Requirements Specification (PRS).
It was anticipated that although all the community portals were going to be similar in design and functionality, each community would have a specific set of requirements that would be deemed essential to portal acceptance. In this sense the business vision was that of a franchise model. This model is prevalent in many industries but is perhaps most visible in the fast-food industry, where a company such as McDonald’s has rigorous quality control standards to enforce an overall corporate look and product delivery, but also allows for regional variations in restaurant decor as well as some regional menu variations. This approach ensures that core functional requirements are defined and established within a basic framework yet offers individual storeowners options for customization. Note that in the end, it is the management of expectations that is being subtly achieved; in the fast-food industry, customers desire and expect a consistent experience with every restaurant they enter, regardless of location.

**Initial Site Definition**

Defining the initial portal site required resolving marketing and engineering judgments on issues such as "look and feel," site navigation, and URL display. As the site matured during its development, each change order became increasingly difficult and time-consuming to implement.

A complicating issue during this process was that the initial site became a blend of off-the-shelf purchased components and products on the one hand and, on the other hand, in-house custom-developed software. Much discussion centered on how the purchased products should be represented in a requirements baseline. Since the features that already existed in purchased software were obviously not going to be redeveloped, there was a discussion about how to account for that set of functionality. Ultimately we decided that requirements documents were intended to capture functionality, whereas architecture documents were intended to capture product design and implementation specifics. After all, even existing product features needed to be tested for applicability and execution in a deployment environment.

As for the technical structure, the site was built on a Microsoft foundation: Windows NT as the operating system, Internet Information Server (IIS) as the Web server, SQL Server as the database store, and Active Server Pages as the Web page delivery mechanism.

**High-Level Design Goals**

The starting point was to come up with the high-level design goals for the project. Initially captured as general project features, these would lead to the elicitation of more detailed actual requirements.

The high-level design goals for the site were:

- to provide maximum flexibility
- to be database-driven
- to provide multiple content feeds
- to categorize user content into a logical structure
- to provide consistent site navigation

These became our functional directives; when design decisions were proposed, they were used to select an appropriate option.
Requirements Elicitation

Since there was no "customer" in the traditional project sense, we decided to work with the small set of core functional directives listed above and to formally establish a project requirements baseline (again, to shorten the requirements analysis process for the subsequent portals) only after the initial portal site was deployed. This had the advantage that we didn't need to conduct a formal requirements acceptance test until the second portal was deployed.

The first step in eliciting requirements was to identify key product stakeholders. These stakeholders consisted of personnel from the marketing, sales, user interface, content generation, software development, quality assurance, and systems engineering teams, all of whom were polled for input. The marketing group represented the viewpoint of an end user in the community and so became the de facto "customer."

A key question that arose during this process was how the Project Requirements Specification (PRS) could be structured to both reflect operational reality and be logical enough that the technical team making functional or other enhancements could isolate their changes (and not affect the overall project). In other words, the primary question became how to build an initial set of well-structured requirements. Let's take a look at the approach that was taken.

Initial Requirements Organization

The core high-level requirements were organized quite simply, into three sections:

- **presentation requirements** — how the information is to be displayed to the user
- **functional requirements** — actions the system must be able to perform
- **nonfunctional requirements** — other system requirements, such as usability, reliability, performance, and supportability

There were a number of reasons for choosing this approach. For one thing, it aligned with how our technical resources and organization were structured. The graphical design team (charged with meeting the presentation requirements) was composed of Web designers who used a different set of tools than the site developers building Active Server Pages. Grouping the functional requirements together, separate from the network, security, and other engineering disciplines, matched the structure of our engineering organization.

This approach also aligned with the groups of people who were responsible for approving the resulting requirements (see Table 1).
This initial high-level structure was then decomposed further, each section in turn. The requirements analysis team was responsible for ensuring that the degrees of uniqueness in each section were reasonably consistent — that is, that one group didn’t come up with a detailed listing of 100 requirements while another peer group listed only ten. Some variations are to be expected, but we decided that if such variations did emerge we would reorganize the overall requirements structure.

The presentation requirements were further refined into 11 distinct areas, the functional requirements into 19 areas, and the nonfunctional requirements into 10. The individual areas are all listed in the table of contents in the next section.

**Requirements Document Structure**

Upon its completion, the PRS was structured according to the following table of contents.

<table>
<thead>
<tr>
<th>1. Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Purpose</td>
</tr>
<tr>
<td>1.2 Scope</td>
</tr>
<tr>
<td>1.3 Definitions</td>
</tr>
<tr>
<td>1.4 References</td>
</tr>
<tr>
<td>1.5 Assumptions</td>
</tr>
<tr>
<td>1.6 Design Constraints</td>
</tr>
<tr>
<td>1.7 Applicable Standards</td>
</tr>
<tr>
<td>1.8 Document Overview</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Presentation Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 General</td>
</tr>
<tr>
<td>2.2 Navigation</td>
</tr>
<tr>
<td>2.3 Home Page</td>
</tr>
<tr>
<td>2.4 People</td>
</tr>
<tr>
<td>2.5 Community</td>
</tr>
<tr>
<td>2.6 News</td>
</tr>
<tr>
<td>2.7 Sports</td>
</tr>
<tr>
<td>2.8 Entertainment</td>
</tr>
</tbody>
</table>

**Table 1: Groups and responsibilities**

<table>
<thead>
<tr>
<th>Group</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marketing</td>
<td>Presentation/functional requirements</td>
</tr>
<tr>
<td>User interface</td>
<td>Presentation requirements</td>
</tr>
<tr>
<td>Business domain experts</td>
<td>Functional requirements</td>
</tr>
<tr>
<td>Engineering</td>
<td>Nonfunctional requirements</td>
</tr>
</tbody>
</table>
The above may be a good start, but have we now built a requirements framework? Given all of the preceding discussion, exactly what constitutes the framework?

Based on the business domain, or to reflect the product mix, different organizations may create different types of frameworks. A typical framework, however, consists of a Rational® RequisitePro® project populated with one or more requirements document templates along with defined requirement types and a set of specific requirements under each type.
Consider that the initial purpose of the framework is to enable a requirements analysis team to more quickly focus in on changes (differentials) to the requirements baseline and to easily replicate key functionality of a previously constructed system. So, to create a framework, a subset of each top-level requirement type was chosen to represent that grouping within an organization; this subset, when put into RequisitePro format, becomes a template to use when creating subsequent requirements specifications.

We decided that a set of 8 to 12 unique requirement types constituted a framework, and that each of these types could have from 5 to 10 requirements, for a total baseline of 40 to 120 requirements in the framework. These numbers could change depending on the specific business domain of the system in question, and so may vary for your particular situation.

As the process of creating a requirements framework matures, a greater number of actual detailed requirements can be archived and reused. In fact, upon system deployment, a useful task for system architects to perform is to review the requirements framework used for that system and refine the framework, or perhaps create a new one.

Over time, a library of requirements frameworks can be established, leading to the ultimate "holy grail" of component-based development: reuse. Among the key factors determining the success of this effort are the number of projects in the library and the comprehensiveness of the framework — for example, whether both functional and nonfunctional requirements are included.

Choosing the RequisitePro Requirement Types

Given the requirements structure shown earlier and the framework structure just described, we then faced the question of just how many — and which — requirement types to create for working within RequisitePro. One extreme would be to create a requirement type for each subsection within a major section of the requirements structure, but that would result in 40 different requirement types, which could become an administrative nightmare. The other extreme would be to have only one requirement type for each requirements section, which would result in only three types for the entire project - too few to have on a project with multiple concurrent teams working.

The answer was (as is so often the case) a compromise between the two extremes. Given the important nature of the presentation aspect of the portal, we gave each of its 11 subsections its own requirement type. The functional requirements, however, were all grouped together under the type of "FUN" and numbered sequentially based on creation; this reflected a blend of clarity and expediency, which was important given the fluid nature of these requirements early on in the project. Finally, the subsections of nonfunctional requirements were also each given their own type, resulting in an additional 10 types. The total was thus 22 different types for the entire project. Although more than our planned maximum of 12 types, this number seemed to strike the appropriate balance between too few and too many requirements types while still being practical from a project implementation perspective.

Another factor to consider is the use of color for requirement types when they're being viewed within the RequisitePro project. With 22 different types, we weren't able to use an easily distinguishable color for each type. At first we decided to instead assign a unique color to the presentation requirements section, to the functional requirements section, and to each of the nonfunctional requirement types, which would have meant using 12 colors in all. However, as the project unfolded we actually used only six colors, with each one being repeated (but separated) in the PRS. Our intent was simply to make the document easier to view during development by having each requirements grouping be a different color than the previous
one. Once a requirements baseline was established, the final "approved" PRS used only black, since not everyone had access to RequisitePro (or a color printer).

Sample RequisitePro Requirements

Finally, let's look at some samples of requirements in each of the three main areas, as they appeared in RequisitePro.

Sample Presentation Requirements

In the General presentation category, the following requirements were displayed:

- GEN1 The entire Web site shall be constructed of data content maintained in a database.
- GEN2 The entire Web site structure shall be maintained in a database.
- GEN3 The entire Web site structure shall be hierarchical in nature.
- GEN4 The same category structure shall be used for all community portals.

The Navigation section included items such as the following:

- NAV7 A user shall be able to return to the home page from whatever page they happen to be on at the time.
- NAV8 Navigation shall be context-sensitive, and navigation options shall change whenever a user enters another category of the Web site.
- NAV9 A navigation path shall be displayed on every page.

Sample Functional Requirements

The functional requirements section included items such as these:

- FUN67 The marketplace category section shall be used to display information on local businesses, merchants, and vendors providing goods and services in the community.
- FUN68 In the marketplace section, the Web site shall provide the following capabilities:
  - Search for a business by name
  - Search for an organization by name
  - Search for a business by function
  - Search for an organization by function
  - View business directory
  - Shop online
  - Browse business directory
  - Post a classified ad

We decided not to individually number each item in the above list of capabilities, to enable subsequent modifications to be made without having an impact on the existing requirements. From a testing perspective, it was important to know simply that there were capabilities to be tested; the actual list of capabilities could (and did) change many times during system design and development.
In the Banner Management subsection within the functional requirements area, some sample requirements were:

- **FUN200** The Web site shall provide a mechanism to manage banner ads.
- **FUN201** Banner ads shall be database-controlled.
- **FUN202** All banner ads shall be compressed to minimize the display time of the image.
- **FUN203** The Web site shall be able to rotate banner ads according to a predefined schedule.
- **FUN204** The Web site shall be able to count the number of times a banner ad is displayed to a system user.

Although all the functional requirements were grouped together when viewed from a traceability standpoint or from a requirements matrix view in RequisitePro, you can see here that they were also grouped by topic within the "FUN" type ("FUN2nn" for those related to banner ads).

### Sample Nonfunctional Requirements

In the Usability nonfunctional category, the following requirements were displayed:

- **USE1** The Web site shall require no special training to be used by an individual in the community.
- **USE2** The Web site shall display a consistent look and feel to every system user.
- **USE3** Generally accepted GUI standards shall be used in the site construction.

The following sample is from the Portal Metrics category of the nonfunctional requirements. Note that just as for the list of Web site capabilities shown earlier, each individual metric in the list of metrics wasn't assigned its own requirement number.

- **MTRC1** The system shall provide metrics to support business ROI (Return on Investment) calculations.

Individual metrics to be provided include the following:

- Number of home page views
- Number of unique visitors
- Total number of page hits
- Visitor profile by location
- Total dollar volume generated by all e-stores
- Comparison of site performance versus industry averages
- Number of site registered users

The Security section included items such as the following:

- **SEC1** Security policies shall be documented and maintained.
- **SEC2** The system shall operate properly behind a firewall designed to provide protection in an Internet environment.
- **SEC3** Site security information useful to the general community shall be provided online.
- **SEC4** The system shall require users to obtain a user ID and password before being given access to restricted services.
Finally, here are some sample requirements from the section for legal, copyright, and other notices:

- LGL2 The Web site shall take care to appropriately mark all content with the necessary copyright notices and trademarks.
- LGL3 The Web site shall maintain an online section describing any necessary legal disclaimers.
- LGL4 Information on legal disclaimers shall be accessible from every page of the Web site.

Summary

This article has presented an approach that was used on a community portal development project to establish an initial requirements baseline, which led to a fully populated Rational RequisitePro requirements database.

A requirements framework can serve to jump-start a development project and be a great labor saver, particularly in the inception phase of a Rational Unified Process® (RUP) project. A typical framework is composed of a set of requirements document templates along with defined requirement types and an initial set of specific requirements for each requirement type. This avoids the "blank sheet of paper" problem when you’re beginning a new initiative.

As this process is repeated, greater reusability can be achieved. Over time, the closer the approach resembles that of a franchise model — a basic framework with options for customization — the more reuse a project manager can expect to see on subsequent project iterations.

Related Resources

- "Requirements Frameworks Cut Development Costs and Time to Market" by Jim Heumann — Frameworks from the standpoint of use-case models


About the Authors

Robert Michalsky has over 20 years of professional computer industry experience with both government and commercial clients. His technical experience includes extensive requirements elicitation and definition, systems performance analysis, visual modeling, software development, and both technical and project management. He is currently certified as a consultant and a trainer for various Rational Software tools and methodologies. Robert can be reached via e-mail.