An Integrated Approach to Documenting Requirements with the Rational Tool Suite

by Kirsten Denney
Advisor
Black Diamond Software

Rational® RequisitePro® is the tool that most development teams associate with the Requirements discipline in the Rational Unified Process® (RUP®). Using it with other integrated tools in the Rational Suite® yields big advantages: more streamlined and efficient requirements artifacts; a better way to address logical data requirements; diagramming capabilities and appropriate management of requirements artifacts, and more. This article presents a start-to-finish process for requirements documentation that takes full advantage of the Rational tool suite and helps ensure that project documentation is complete. It also suggests modifications to the typical RUP approach that improve and augment both the workflows and the final deliverables of the requirements discipline.

To follow the recommended process, you should have authorized access to the following tools as well as basic knowledge of how to use them.[1]

- *A recent version of Microsoft Word*
- *Rational RequisitePro*
- *Rational® ClearCase®*
- *Rational Rose®*
- *Rational® SoDA®*
- *The Rational Unified Process Web site (for reference purposes)*

Advantages of this Approach

Before we walk through the process itself, let's take a closer look at some advantages of combining RequisitePro with other tools in the Rational Suite to document requirements.

Easy-to-Use Interface to Capture, Maintain, and Trace Requirements

This approach advocates the use of Rational RequisitePro as the database repository for the various types of requirements (e.g., Stakeholder Requests, Features, etc.) required for a project. RequisitePro provides an easy-to-use mechanism for entering and maintaining requirements along with their attributes and relationships.

An important aspect of capturing the relationships between requirements is establishing traceability from requirements of one type to requirements of another type. Traceability is critical to managing the scope of the system and ensuring that all system elements have been appropriately addressed. RequisitePro provides an extremely easy-to-use interface for tracing requirement types.[2]

Easy Way to Maintain and Publish Up-to-Date Deliverable Artifacts

Most projects use Microsoft Word to capture requirements in a publishable format; the documents are then imported into Rational RequisitePro using the Import function. Requirements in the document are maintained via the dynamic linkage between RequisitePro and the Word document. While this approach works well, it might not be optimal, especially when the project's documents are managed under source control. In that case it is far easier to enter and maintain the requirements information directly within RequisitePro. But then, how can you publish them for distribution?

Rational SoDA for Word addresses this challenge. SoDA provides the ability to embed reports within Word documents so that the data can be automatically extracted directly through RequisitePro (and other Rational tools). For example, the Vision document can include SoDA commands that enable you to extract the Stakeholders, Stakeholder Requests, and Features directly through RequisitePro.

Robust Configuration Management Capabilities

Use of a configuration management (CM) tool should be a requirement for all projects as a means of safeguarding and tracking project artifacts. Rational ClearCase is one of the more comprehensive CM products on the market and can be implemented to support all of your project artifacts, including those for the requirements discipline.

Versioning of Database-Stored Requirements

Because RequisitePro is built on a database, it does not lend itself particularly well to configuration management software. This disadvantage can easily be overcome by using ClearCase to manage the SoDA-generated documents. When the SoDA documents, as standard project artifacts, are added to the ClearCase repository, they provide an historical snapshot of the requirements data at the time that they were generated. This ability to use ClearCase to version and baseline the requirements is especially important in federally regulated environments.

Ability to Include Use-Case Model Diagrams in Artifacts

The RUP instructs teams to derive features from Stakeholder Requests and then model them into use cases. Rational Rose provides an easy-to-use interface for creating diagrams of the use case models that can be copied into project artifacts through the use of SoDA commands. These models provide a helpful, high-level inventory of the system's use cases and their relationships to one another.

Centralized Access to Complete Use-Case Information

After the use cases are modeled, they need to be fully described (including flows and constraints) and also tracked in RequisitePro (for traceability). Use-Case Specifications are Word documents that describe the use cases. Rational Rose allows you to attach these documents to the use-case model; when you do so, you implicitly create a link from Rose to the corresponding use-case requirement in RequisitePro. Consequently, the model provides convenient, centralized access to full documentation of the use cases for the project's analysts and designers.

Attaching the documents to the model also creates a direct link from RequisitePro to the Use-Case Specification. Because use cases are the only type of requirement maintained in RequisitePro that is not fully described in RequisitePro (we'll revisit this in a later discussion about requirement types), this feature provides a handy way to directly access complete information about the use cases from RequisitePro.

SoDA makes it possible to pull the model and use-case specifications directly into the appropriate project documents (specifically, the Software Requirements Specification).

Complete Data Requirements

The RUP does not discuss in detail the gathering of logical data requirements, yet these requirements represent critical input for system designers and analysts. The process we describe below ("The Process" section of this article) helps ensure that these requirements are appropriately included in the requirements documentation.

Logical data requirements can be represented in Rational Rose either as Class Diagrams (in the Logical View) or Data Model Diagrams; how you intend to treat design versus analysis within the system will determine which is best suited to your needs. Each type of diagram allows you to create high-level depictions of your logical data entities that provide an ideal means for specifying the multiplicity and direction of relationships between the entities. Each also allows you to capture all of the details about the data, in a structured format, so that these diagrams can serve as a single repository for the project's logical data requirements.

SoDA commands are used to extract the models and data details into the appropriate distributable documents (specifically the "Data Requirements" document).

Our approach also includes *information nicknames*, a term that Alistair Cockburn[3] uses to describe chunks of related information. For example "Customer Name" may be a nickname for the combination of First Name, Middle Initial, and Last Name from a Customer entity. Customer Address and Customer Billing Information may be additional nicknames. By using nicknames, rather than specifying actual data fields in our use cases, we are able to separate the tasks of detailing the functional requirements and detailing the data requirements. This also allows us to avoid specificity in use cases that can complicate the development and maintenance of the actual logical data requirements.

Complete Business Rules

In the RUP, the activity of gathering business rules is located within the Business Modeling discipline, which is not treated as a "core" discipline and does not include detailed guidance. Business rules, however, are critical to designers and analysts whose primary inputs are requirements artifacts, so our approach treats them as a critical component of documenting system requirements.

The Process

This section details the actual steps required for this approach. Although we will present them sequentially for simplicity's sake, with the exception of the first step, these steps are very likely to occur in iterations and even in parallel within iterations.

Configure the Requirements Environment

The first step of this process calls for configuring RequisitePro and establishing your project templates so that later steps can make appropriate use of them. The tasks within this step are as follows:

1. **Have your Rational Administrator configure RequisitePro so that your project documents are saved as Word documents.**[4]

   In my opinion, the simplicity of using standard Word documents outweighs the benefits of using the proprietary RequisitePro format.

2. **Configure the Requirements Management Plan (RMP).**

   The RMP documents the requirement types, attributes, and traceability that are to be applied to this project; it also provides the blueprint for how RequisitePro should be configured for a particular project.

   If you do not already have an RMP template, use the one provided in RequisitePro.[5]

Within the template, define each of the following requirement types in addition to any others that your project requires. Suggested attributes for some of these requirement types may be found in the RUP Vision document and the RequisitePro default template; however, specific note has been made of attributes required for this approach. In addition, the traceability required for each requirement type is noted and should be documented in the RMP.

Stakeholder. This requirement type allows us to record and trace the ultimate source of any given requirement and provides information about the source so that we may better qualify the request.

Include an attribute called "Actor," with a value of Yes/No so that Stakeholders may be identified as Actors (users) where appropriate. This distinction is necessary within the Vision document, and is useful for filtering when reporting on and tracing to other elements, such as use cases.

Stakeholder Requests. This requirement type allows us to track the various needs of the Stakeholders. Consequently, this type of requirement traces back to the stakeholder requirement type.

Feature. Features are the high-level capabilities of the system derived from Stakeholder Requests.

An attribute to distinguish types of features will facilitate the inclusion of this information in documents described later. The Vision document provides some guidance for this. For example, you might want to include an attribute of Type, with values of Functional, Non-Functional and Documentation. Or you might decide to decompose Non-Functional into System, Standards, and Performance. Or you might want to use the RUP "FURPS" types: Functionality, Usability, Reliability, Performance, and Supportability.[7] Use the categories that work best for your project.

Features trace back to Stakeholder Requests; this helps to ensure that there are no superfluous features or unaddressed stakeholder requests.

Use Case. Use cases provide a context for the features of the overall system by providing scenarios for their usage. Unlike RUP, this approach does not recommend putting the use-case specifics, such as pre- and post-conditions and flows of events, into RequisitePro, as this can make the use cases very unwieldy and does not yield great benefit.[7] Consequently, only an inventory of the use cases -- name and short description -- is included in RequisitePro, along with any other standard attributes, such as Priority, Status, and so forth.

Use cases trace back to the features from which they are derived, and to the Actors (Stakeholders with the Actor attribute) that will be users of the use case.

Supplementary Specification. Supplementary specifications, the detailed non-functional requirements of the application, are derived from the non-functional features identified for the application. So they, too, trace back to Features.

You may wish to provide an attribute similar to the Type attribute for features, in order to allow similar distinctions between specifications. As with features, this will be useful for documents described later.

Ultimately, all of the features should be traced to either one or more supplementary requirements and/or one or more use cases; this check will verify that all of the system features have been addressed by the requirements process.

You may also choose to include requirement types for data entities and/or business rules, specifically for the purpose of traceability to the use cases that invoke them. In the process described here, we will not include such types.

3. **Check the RMP into Rational ClearCase.**

   All project artifacts should be checked into ClearCase on a regular basis as a means of safeguarding, tracking, and controlling them.

4. **Have your Rational Administrator configure RequisitePro to reflect the requirements defined in the RMP.**

   The Administrator should use the Composite or Use-Case Model template as a starting point for the project if a standard template is not already defined.

5. **Tailor the document templates required for this approach.**

   Tailor the following templates as described, and ensure that they are documented in your project's Development Case[8]:

   Vision Document. A default template is provided in RequisitePro; however, you will need to convert the template into a SoDA report format so that the information entered into RequisitePro will be automatically retrieved into the report.

   Replace the following sections of the RUP Vision template with the appropriate SoDA commands (a table format is recommended for these sections):

   - **Stakeholder** -- Replace with a summary list of the Stakeholders with Actor=No. Rename this section "Stakeholder Summary -- Non-Users." Leaving the name as-is implies that Stakeholder are not users, which runs contrary to an important RUP concept.

- **User Summary** -- Replace with a summary list of the Stakeholders with Actor=Yes. Rename this section "Stakeholder Summary -- Users."
- **Stakeholder Profiles** -- Replace with detailed information about each Stakeholder with Actor=No. Rename this section appropriately.
- **User Profiles** -- Replace with detailed information about each Stakeholder with Actor=Yes. Rename this section appropriately.
- **Key Stakeholder or User Needs** -- Replace with a summary list of the Stakeholder Requests identified in RequisitePro.
- **Product Features** -- Replace with Features that have the Type attribute "Function" in RequisitePro.
- **Other Product Requirements** -- Replace with Features typed as "Non-Functional" (or System, Performance, Standards, etc.) in RequisitePro. Structure the headings and group by type, as appropriate for your project.
- **Documentation Requirements** -- Replace with Features typed as "Documentation" in RequisitePro. If more elaboration is required for this section than is feasible to capture in RequisitePro, augment this section as appropriate.

You may also choose to include a section or an Appendix to capture the traceability attributes of the requirements; SoDA provides the commands to do this. Tracing requirements -- and the ability to use traceability to determine the impact of changes -- is critical to every project; the need to publish them is perhaps less so.

Use-Case Specification (UCS). A default template is provided in RequisitePro; however, we recommend that the Description be replaced with the Goal of the use case. The description often ends up repeating the flow or the title of the UCS, whereas a Goal provides something against which we can clearly measure the success of a flow within the use case. This helps to clarify the delineation between primary, alternate/optional, and exceptional flows.

SoDA provides the option of automatically incorporating the Use-Case Specifications into the Software Requirements Specifications. This is recommended, unless you feel that the volume or complexity of the use cases would be better managed by keeping them separate. If you do include them in the SRS, do not include a title page or Table of Contents in the individual Use-Case Specifications.

Software Requirements Specification. A template for this document is provided in RequisitePro (if the Composite project template was used to create the project) under the title "Modern Software Requirements Specification"; the RUP also provides a template. However, you will need to convert the template into a SoDA report format so that the use cases and supplementary specifications entered into RequisitePro will be automatically retrieved into the report.

The following are necessary modifications to this document:

- Add a section titled "Use Case Model Diagram" within the Use Case Model Survey section. Use SoDA commands to incorporate the use-case model from Rational Rose.
- Add a section called "Actors" that precedes the Use-Case Model Survey. Use SoDA commands to extract the Name and Text (description) of the Stakeholders in RequisitePro with Actor=Yes.
- Replace the following three sections with SoDA commands (in a table format):
    1. Use-Case Model Survey (or Use-Case Hierarchy, depending upon the template) -- replace with a summary list of the use cases, as specified in RequisitePro.
    2. Use-Case Reports (or Use-Case Specifications) -- replace with the Use-Case Specification documents themselves.
    3. Supplementary Requirements (or the corresponding sections in the Modern SRS format) -- replace with the supplementary specifications in RequisitePro. Group and subtitle by Type as appropriate for your project.

Data Requirements Document. There is no template for this document in RequisitePro or the RUP; however, you can use any other RUP artifact as a starting point. Include the usual sections: title page, Table of Contents, and Introduction. The content specific to this document includes the following:

- A section named "Logical Data Model" (or "Logical Class Diagram," depending upon whether you use the Data Modeler or the Logical View Class Diagram). In this section, use the appropriate SoDA commands to extract the Data Model/Logical Class Diagram from Rose.
- A section named "Additional Attribute Information" to include information about the entities that is contained within the model but not displayed in the image. These properties are extracted from the model using SoDA commands.
- A section called "Information Nicknames." This section will map the information nicknames in use cases to actual entities and attributes in the logical data model. Use a table format, with columns for the following:
    - Information nickname.

        - Name of logical entity to which the mapped attributes belong.
        - Names of attributes that correspond to the nickname.

        This section is completed manually; no SoDA commands are necessary.

    - A section for lookup tables. These indicate valid values for any static lists that are used in the application. (Actually, the lists may not be truly static, because they can be updated via administrative functions or database interfaces). If an existing database table or input file is to be used, indicate the name and location of the file instead of its contents.

In many projects, some or all of the data may be extracted from a legacy system or dictated by feeds to or from other systems. In this case, it may not be necessary to do further analysis of the data within the Requirements discipline. Instead, the location of this information (for example, the pathname of a sample file feed or the appropriate datastore on a server) is sufficient and can be included in the SRS; the Database Designer can take it from there.

Business Rules Document. A template for this document is provided in the RUP.[9] Depending upon the volume and/or complexity of the business rules, you may choose to incorporate the business rules directly into the SRS. This approach assumes that you will use a separate Business Rules document.

6. **Check the templates into ClearCase.**

Document the Vision

At this point, the tools and templates required to capture the Requirements information have been configured. The following tasks describe how to use those tools and templates to capture the information required to describe the project Vision.

The Vision document in the RUP defines the Stakeholders' view of the product to be developed. It typically identifies the Stakeholders, their requests, and the high-level system features that are derived from those requests. This document is a critical artifact early in the Inception phase of the project because it allows the Stakeholders to verify that their vision is being properly captured by the requirements gathering process, and it gives the project team the information necessary to start formulating project plans and other project artifacts.

Following are the tasks necessary to complete this step:

1. **Add the requirements information to Rational RequisitePro.**

As you elicit stakeholder information and identify features, capture the following in RequisitePro. Use the RequisitePro attributes of each requirement type as a guideline to ensure complete descriptions.

- Stakeholders
- Stakeholder Requests
- Features

2. **Establish traceability.**

   Indicate the following traceability in RequisitePro:

   - Stakeholder Requests to Stakeholders
   - Features to Stakeholder Requests

3. **Create the Vision document.**

   Complete the sections of the Vision that will not be automatically populated from RequisitePro (e.g. Introduction, Positioning, Product Overview). Once you've done that, generate the SoDA report and update the Table of Contents in the resulting report; this will provide an up-to-date Vision document.

4. **Add the Vision document to Rational ClearCase.**

   Add the SoDA-generated version of the Vision document to the ClearCase repository. This will result in a consolidated and baselined version of the raw requirements.

Supplementary specifications, business rules, and data requirements are also likely to be uncovered during this process. These should be addressed as discussed in the following steps.

Document the Functional Requirements

At this point, the system features are identified in RequisitePro and the Vision document. Now the challenge is to derive use cases from these features.

Use cases describe units of system behavior in terms of a sequence of interactions between the system and the Actor. Each use case provides an observable benefit to the Actor; for example, use cases may have names such as "Add Customer," "Add Order," or "Ship Order." Use cases provide a context for the individual features of the system and are therefore very helpful to the project's reviewers, analysts, and designers. Here are the activities you must perform to derive use cases from features.

1. **Model the use cases.**

   Use Rational Rose to structure the use-case model appropriate to the requirements information captured in the previous step.[10]

2. **Create the Use-Case Specifications.**

   Within Rose, right click on each use case within the model that is to be elaborated upon. Click on **Use Case Document -> New**. This will open up RequisitePro (if it is not already open) and a new Use-Case Specification document[11], which will allow you to elaborate the use case.[12]

   When you save the Use-Case Specification using RequisitePro's Save Requirements Document feature, the name of the use case will be automatically added to RequisitePro as a Use-Case Requirement. Consequently, by saving each document in RequisitePro, you will create an inventory of use cases in RequisitePro. This inventory can then be used to trace the use cases to the features that spawned them. This functionality also results in a direct link from Rose into the RequisitePro requirement for the use case.

   A few conventions come in handy when elaborating use cases[13]:

   - Use boldface type to indicate the names of use cases that are invoked by another use case. This will highlight the relationship between use cases.
   - Use italics to indicate the names of business rules that are invoked by a use case. This will allow you to write simple sentences such as "Display all *approved* deals," in which the italics indicate there is a business rule describing what constitutes an approved deal. As the user can refer to the Business Rules document for details, you don't need further explanation in the sentence.
   - Avoid including data details in the use cases, because data requirements are highly susceptible to change. Instead, wherever possible, use information nicknames that will be mapped to actual logical data entities in the Data Requirements document. (You might want to create nicknames even for standalone attributes so that they may change names or entities without affecting the use cases that invoke them.)
   - Use Activity Models if they will help illustrate the use cases. [14]

   If you do choose to use any of these conventions, explain them within the highest-level document (either the SRS or the individual Use-Case Specifications) in which the use cases are published.

   As with the previous step, non-functional requirements are also likely to be uncovered during this process. These are addressed in the following steps.

3. **Update the use-case attributes in RequisitePro.**

   Since the use-case requirements were automatically added to RequisitePro when each of the Use- Case Specifications was created, you need to update the corresponding attributes directly in RequisitePro.

4. **Establish traceability.**

   Indicate the following traceability in RequisitePro:
   - Use Cases to Features.
   - Use Cases to Actors.

     To do this, you might want to use a RequisitePro query to display only Stakeholders that are designated as Actors.

5. **Add the Use-Case Specifications to ClearCase.**

At this point, the functional requirements for the system are diagrammed in Rose, inventoried in RequisitePro, and fully elaborated in the Use-Case Specifications. The Use-Case Specifications and requirements are accessible from either Rose or RequisitePro.

You may generate the SRS at this point, as a check against the SoDA template and commands; however, for simplicity's sake, we will not discuss this until we come to the next step, when all the information to be included in the SRS is complete.

Document the Non-Functional Requirements

Supplementary specifications are the detailed non-functional requirements for the system, including such things as performance requirements, system constraints, and legal restrictions. They represent the fine detail of the non-functional features that have been identified for the project.

Supplementary specifications are fairly straightforward. Like the features, they are entered directly into RequisitePro. However, the supplementary specifications complete the content of the SRS, and that document is therefore generated at the end of this step, which you can perform as follows:

- **Add the supplementary specifications to RequisitePro.**
- **Trace the supplementary specifications to the features that spawned them.**
- **Generate the SRS.**

- Now that all of the Actors, use cases, and supplementary specifications for the system have been entered into RequisitePro, make sure that any other non-RequisitePro information required in the document has been updated. Then, generate the SRS SoDA report and update the Table of Contents.
- If the SRS was defined as described earlier, it will pull in the Use-Case Model from Rose, the Actors and Use-Case Inventory from RequisitePro, and the Use-Case Specifications.
- **Check the SRS into Clear Case.**

Document the Data Requirements

Data requirements are typically uncovered during the requirements gathering and elaboration processes, so this step happens largely in parallel with the previous steps. However, the finer details of the data requirements may still need to be determined once the use cases are fully elaborated.

The data requirements include the names of *logical* or *conceptual* entities in the system, such as Customer, Order, and Item, as well as details about the entities (attributes, type, length, validation rules, etc.) and the relationships between the entities (multiplicity and direction). The goal is to capture this information in a formal way that may then be used during analysis and design activities. Here's how to perform this step:

1. **Create the Logical Data Model or Class Diagram.**

   Both the Rational Rose Data Modeler and the Logical View provide the ability to fully model and describe the logical data entities in your system. Theoretically, you would create a diagram of the logical data in the Data Modeler, and the Database Designer would use that to create a diagram of the physical data. The Designer would use the physical model to create the Class Diagram in the Logical view. However, depending upon the size and roles of the project, those steps may partially or fully collapse into each other. For example, you may have only a single Data Model and a Logical Class Diagram, or even just a Class Diagram. Use the tool that applies best to your project's needs.

   Once you've decided which tool to use, describe the data that is required for the project. Use the model or diagram to indicate the multiplicity and types of relationships between entities; use the specification pop-ups to describe any constraints (e.g., type, domain, default value, etc.) for the attributes. When describing constraints, it is recommended that you use domains for common constraints (e.g., phone number formats) and a standard language (e.g., OCL) for describing the rest.

2. **Map Information Nicknames to logical entities and attributes.**

In the Information Nicknames section of the Data Requirements document, map the Information Nicknames that have been identified in the use cases to the appropriate logical entities and attributes.

3. **Generate the Data Requirements document.**

   Generate the Data Requirements SoDA report and update the Table of Contents.

   If this document was defined as described earlier, it will pull in the Logical Data Model and additional attribute information from Rose, and include the mapping of the information nicknames that you completed in step #2 above.

4. **Check the Data Requirements into ClearCase.**

Document the Business Rules

Business rules declare policies, calculations, and logic that are relevant to an application. They might relate to determining status or calculating fees, for example. Business rules are also typically uncovered in parallel with the previous steps.

Because business rules often include pseudocode, which does not lend itself well to RequisitePro, they are documented in the Business Rules document. You may, however, wish to inventory them in RequisitePro for the sake of traceability. In any case, here is how to proceed:

1. **Document the business rules directly in the Business Rules document.**

   When describing the logic encapsulated by the business rule, consider the use of pseudo-code, the UML Object Constraint Language (OCL),[15] and/or flow diagrams in order to provide a concise description of the necessary logic.

2. **Check the Business Rules document into ClearCase.**

   After this step is completed, you will have a complete and fully traceable set of requirements that can be easily maintained and published for use by your project team and reviewers.

Additional Considerations

Below we describe a few more tools and documents you might want to add to your requirements management process.

Rational ClearQuest

Rational ClearQuest is the change management component of the Rational tool suite. Bug reports in ClearQuest can generate requirements for a project that is already under development or already deployed, and you can easily migrate these new requirements to RequisitePro through a ClearQuest / RequisitePro integration. This ensures that RequisitePro remains the single, consolidated repository of project requirements. For more information about this integration, see the RUP Web site.[16]

Glossary

The Glossary is a requirements artifact in RUP, so there is a template for it on the RUP Web site.[17] However, we recommend a tabular format that is easier to read and sort. We do not recommend that you store the terms in RequisitePro unless, perhaps, you intend to publish them via a SoDA report format. There is no discernible benefit otherwise.

Risk List

Although a Risk List is not part of the requirements discipline in the RUP, it is essential that those involved in the requirements process actively contribute to such as list. Requirements-related activities are most intense during the Inception phase of the project, so they provide a unique opportunity to identify and manage risks at the earliest stages.

Although the RUP does provide a template for a Risk List, we recommend an Excel format. It is far simpler to work with, most importantly in terms of ranking the risks by magnitude. Another approach is to use RequisitePro to track risks; this will afford you the ability to trace risks to the use cases and supplementary specifications that they impact.

Summary

The goal of this article is to provide readers with the following:

- The direction you need to establish and use an integrated and effective Rational tool suite environment for your requirements documentation effort.
- Guidance for augmenting the standard RUP artifacts to ensure that designers and analysts receive a complete set of requirements information.

By following this guidance, you'll benefit from an easy-to-use interface for capturing, maintaining, and tracing requirements; a more automated method for updating and publishing deliverable artifacts; robust configuration management capabilities, centralized access to use-case information, and inclusion of use-case model diagrams, data requirements, and business rules in your artifacts. Even more important, you'll enjoy the greater benefits that well-documented requirements bring to a project: an architecture that reflects Stakeholders' real needs; a foundation for effective communication among development team members and between team members and other Stakeholders; a basis for evaluating and filtering future changes requests; and much more.

Notes

1 Note that this article does not provide step-by-step instruction for using Rational tools; you can find that information in Rational user manuals and online Help. Also, although we will discuss how to document requirements information, we will not discuss how to gather that information or how to manage it throughout a project lifecycle. Refer to the RUP Web site and the online version of RUP for guidance on these topics.

2 For more information about the benefits of traceability, refer to the online version of RUP: RUP -> Disciplines -> Requirements -> Concepts -> Traceability

3 Alistair Cockburn, *Writing Effective Use Cases*. Addison-Wesley, 2001.

4 RequisitePro -> File -> Project Administration -> Properties -> Documents tab -> uncheck the box marked "Save documents in RequisitePro format"

5 RequisitePro -> File -> New -> Document -> select the Requirements Management Plan document type

6 RUP -> Disciplines -> Requirements -> Concepts -> Requirements

7 This statement may not be entirely true, depending upon how you design and trace your test cases; consult with your test designers before making a final decision. If you do put the specifics into RequisitePro, you will need to use SoDA to create the Use-Case Specifications, which are described later. However, you should make sure your team is comfortable with this more complex approach. This article assumes that the specifics are not included in RequisitePro.

8 RUP -> Artifacts -> Environment -> Development Case

9 RUP -> Artifacts -> Business Modeling -> Business Rules

10 Refer to the online version of RUP for more information about the purpose of, and means for, use-case modeling: RUP -> Disciplines -> Requirements -> Guidelines. This section has a variety of relevant and helpful documents. In addition, the use of patterns may be helpful, especially for large or regulated projects. See RUP -> Disciplines -> Business Modeling -> Concepts -> Business patterns.

11 RUP -> Tool Mentors -> Rational Suite AnalystStudio Tool Mentors -> Managing Use Cases Using Rational Rose and RequisitePro will provide more information about associating Rational Rose and RequisitePro, should you have any problems with this task.

12 See RUP -> Disciplines -> Requirements -> Guidelines. There are a variety of relevant and helpful documents in this section.

13 You can add Word format styles to your use-case templates to help ensure consistent use of these conventions.

14 RUP -> Disciplines -> Business Modeling -> Guidelines -> Activity Diagram in the Business Use Case Model

15 RUP -> Disciplines -> Business Modeling -> Guidelines -> Business Rules

16 RUP -> Tool Mentors -> Rational Suite AnalystStudio Tool Mentors -> Managing Stakeholder Requests Using Rational ClearQuest and Rational RequisitePro

17RUP -> Artifacts -> Requirements -> Glossary

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!