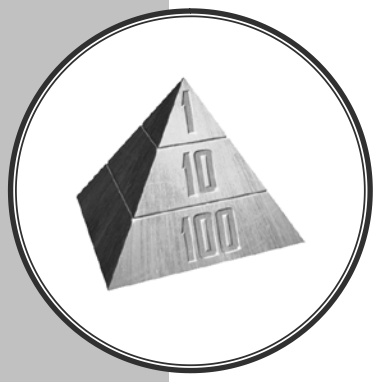
P

Applying Requirements Management with Use Cases



Roger Oberg
Leslee Probasco
Maria Ericsson

Technical Paper TP505

Roger Oberg is Vice President and General Manager for Rational Software's Requirements Management products and on the Board of Directors of two start-up software companies. He has 16 years of high technology marketing and engineering management experience in office automation systems, software development tools, and telecommunications, and a B.A. in Economics from the University of Michigan.

Leslee Probasco is the Product Manager for Rational University's Requirements Management professional education courses. She is a trainer and consultant with experience teaching requirements management, project management, TQM, CMM and ISO 9000.

Leslee has twenty years of software management and development experience in the areas of military and aerospace systems, telecommunications, oil and gas, and electrical engineering. Leslee has a M.S. in computer science from the University of Denver and a B.S. in Management Science and Operations Research from Colorado State University.

Maria Ericsson is Senior Technical Consultant for Rational Software, located in McLean, Virginia. Maria Ericsson is part of the team developing the Rational Unified Process, and the co-author of the textbook The Object Advantage: Business Process Engineering with Object Technology. Maria has a M.S. in Engineering Physics from the Royal Institute of Technology in Stockholm, Sweden

Applying Requirements Management with Use Cases

by Roger Oberg, Leslee Probasco and Maria Ericsson

If you are new to or somewhat familiar with requirements management and are interested in requirements process improvement, this paper offers a framework with which to develop your own approach.

USE CASES AND SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS)

To make requirements management workflows more meaningful to readers, the authors have chosen specific document types and other requirements management artifacts from Rational Software's Unified Process™ and the industry-standard Unified Modeling Language, both of which recommend a ~~U~~use-~~C~~case-driven software engineering process.

Therefore, a ~~U~~use-~~C~~case-driven approach for specifying software requirements is described here. These workflows may also be used with the more traditional Software Requirements Specifications (e.g., IEEE standards) in place of or in addition to ~~U~~use-~~C~~case ~~M~~models and ~~U~~use ~~C~~cases.

Software and System Development in the Age of Process

For most software and system* development teams, the 1990s have beenappearhave been process-intensive when compared to the more freewheeling days of the past. Standards for measuring and certifying effective software development process (e.g., SEI CMM, ISO 9000) have been introduced and popularized. Many books and articles on software development process and related material on business process, business process modeling, and business process re-engineering etc., have been published. IThere are increasing numbers of software tools have helped that defined and helped apply effective software development process. To what end? There is no doubt of some progress. The global economy's dependence on software accelerated in the decade, enabling development processes and improving s as system quality and enabling development processes improved.

So how do we explain the high incidence of the software project failure today? Why are many, if not most, of software the projects you have worked on this decade still plagued by delays, budget overruns, and quality problems? How can we improve the quality of the systems we build as our businesses, national economies, and daily activities become increasingly dependent on them?

The answers, as always, lie in the people, tools, and processes applied to our profession. Requirements management is often proposed as a solutionOne particular contributor to the ongoing problems of software development— the management of requirements — is cited often, yet relatively little attention has been focused on improving the practice of this disciplineit.

The purpose of this paper is to presents the elements of an effective requirements management process and highlights some of the obstacles to its successful implementation.

* Requirements management applies equally to software-only projects and to projects in which software is only a part of the end result or not included at all. For convenience, the paper will hereafter use the term "system" to mean any or all of these things. However, it is the abstract nature of software development, alone or in combination with hardware that complicates requirements management, and is therefore the primary focus of the paper.

Why Manage Requirements?

Simply put, ~~those~~ system~~s~~ – development teams who manage requirements do so because they want their projects to succeed. M~~, and m~~eeting their project's requirements defines success. F~~. F~~ailing to manage requirements de~~in~~creases the probability of ~~not~~ meeting ~~them~~these objectives.

Meeting project's requirements defines success.

Formatted

Recent evidence is supportive:

- The Standish Group's CHAOS Reports from 1994 and 1997 established that the most significant contributors to project failure relate to requirements.[1]
- In December 1997, Computer Industry Daily reported on a Sequent Computer Systems, Inc. study of 500 IT managers in the U.S. and U.K. that found 76 percent~~%~~ of the respondents had experienced complete project failure during their careers. The most frequently named cause of project failure was "changing user requirements."[2]

Avoiding failure should be sufficient motivation to manage requirements. Increasing the probability of a successful project and other benefits of managing requirements may be equally motivational. The Standish Group's CHAOS report further established that managing requirements well was the factor most related to successful projects.

What is a Requirement?

The first step towards understanding requirements management is to agree on a common vocabulary. ~~To that end,~~ Rational defines a requirement as "a condition or capability to which the system [being built] must conform." The Institute of Electronics and Electrical Engineers uses a similar definition.

Well-known r~~R~~equirements e~~E~~ngineering authors Merlin Dorfman and Richard H. Thayer offer a compatible and more refined definition that is specific – but not necessarily limited – to software:

A requ... a cond... capability to which the system must conform.

Formatted

"A software requirement can be defined as:

- A software capability needed by the user to solve a problem or achieve an objective.
- A software capability that must be met or possessed by a system or system component to satisfy a contract, specification, standard, or other formally imposed documentation."[3]

What is Requirements Management?

~~If~~Since requirements are things to which the system being built must conform, and conformance to some set of requirements defines the success or failure of projects, it makes sense to find out what the requirements are, write them down, organize them, and track them in the event they change.

Stated another way, ~~:~~ ***R***

~~R~~equirements Management is:

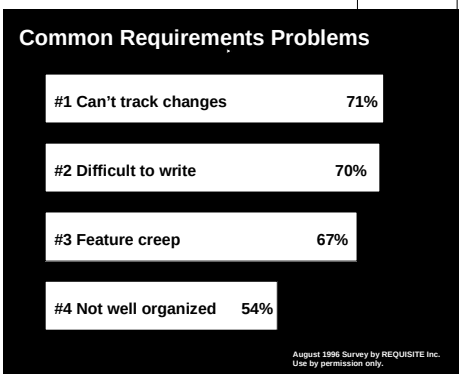
- ***a systematic approach to eliciting, organizing, and documenting the requirements of the system, and***
- ***a process that establish~~es~~ing and maintain~~s~~ing agreement between the customer and the project team on the changing requirements of the system.***

This definition is similar to Dorfman and Thayer's and the IEEE's "Software Requirements Engineering." Requirements Engineering includes elicitation*, analysis, specification, verification, and management[4]. All of these activities are incorporated in the definition of requirements management presented here and taught by Rational Software. The difference lies mainly in the choice of the word "management" rather than "engineering." Management is a more appropriate description of all the activities involved, and it accurately emphasizes the importance of tracking changes to maintain agreements between stakeholders and the project team.

The Problems of Requirements Management

So what might be difficult about a process intended to ensure that a system conforms to the expectations set for it?

Figure 1



When put into practice on real projects, difficulties come to light. Figure 1 displays the results of a 1996 survey of developers, managers, and quality assurance personnel. It shows the percentage of respondents who experienced the most frequently mentioned requirements-related problems.

A more comprehensive list of problems includes:

- Requirements are not always obvious and have many sources.
- Requirements are not always easy to express clearly in words.
- There are many different types of requirements at different levels of detail.
- The number of requirements can become unmanageable if not controlled.
- Requirements are related to one another in a variety of ways, and to other deliverables of the system development process in a variety of ways.
- Requirements have unique properties or property values. For example, they are neither equally important nor equally easy to meet.
- There are many interested and responsible parties, which means requirements need to be managed by cross-functional groups of people.
- Requirements change.
- Requirements can be time-sensitive.

When these problems are combined with inadequate requirements management and process skills, and the lack of easy-to-use tools, many teams despair of ever managing requirements well. Fortunately, Rational Software has developed the expertise to instruct teams in requirements management skills and process. In addition, Rational has created RequisitePro™, an accessible tool for automating effective requirements management.

* For those unfamiliar with the term "elicitation," it is defined as the set of activities that teams employ to elicit or discover stakeholder requirements.

Requirements Management Skills

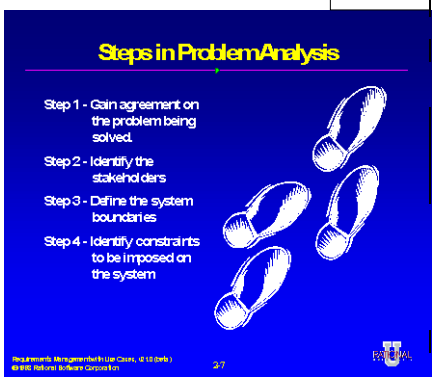
To resolve ~~t~~these problems mentioned above, Rational encourages the development of ***key skills***. ~~They are~~ These skills are presented below in what appears to be sequential order~~; however, that may be misleading. We refer to them as "skills" rather than as a process because~~ but in an effective requirements management process they are applied continuously in varied order. Here they are presented in the sequence one would likely apply to the first iteration of a new project.

Key Skill 1: Problem Analysis

Problem analysis is ~~done~~ conducted to understand business problems, target initial stakeholder needs, and propose high-level solutions. These ~~It is an~~ acts of reasoning and analysis ~~to~~ find "the problem behind the problem."

During problem analysis, agreement is gained on a statement of the real problem~~(s)~~ and the stakeholders are identified. Initial solution boundaries and constraints are defined from both technical and business perspectives. If appropriate, the business case for the project ~~should also be~~ analyze~~sd so that there is~~to ~~gain a good understanding of what~~ return on investment that is expected ~~on the investment made in~~from the system ~~being built~~.

Fig~~-~~ure 2



Key Skill 2: Understanding Stakeholder Needs

Requirements have many sources. They may come from anyone with an interest in the outcome of the project. Customers, partners, end users, and domain experts are some sources of requirements. Management, project team members, business policies, and regulatory agencies can be others.

~~-~~It is important to know how to ~~best~~ determine ~~what~~ who the sources should be, how to get access to those sources, and how to ~~and also how to best~~ elicit information from them. The individuals who serve as ~~provide the~~ primary sources for this information ar~~w~~e referred to as "stakeholders" in the project.

~~-~~If you are developing an information system to be used internally within your company, you may include people with end~~-~~user experience and business domain expertise in your development team. Very often you will start the discussions at a business-model level rather than at a system level. If you are developing a product to be sold to a marketplace, you may make extensive use of your marketing people to better understand the needs of customers in that market.

Techniques for eliciting requirements include interviews, brainstorming, conceptual prototyping, questionnaires, and competitive analysis. The result of requirements elicitation is a list of requests or needs that are described textually and graphically, and that have been given priority relative to one another.

Formatted

Formatted

Key Skill 3: Defining the System

To define the system means to translate and organize the understanding of stakeholder needs into a meaningful description* of the system to be built. Early in system definition, decisions are made on what constitutes a requirement, documentation format, language formality, degree of requirements, request priority and estimated

* We use the word "description" rather than "document" to avoid the perceived limitation inherent in the common use of the latter. A description may be a written document, electronic file, a picture, or any other representation meant to communicate system requirements short of the system itself.

effort, technical and management risks, and scope. Part of this activity may include early prototypes and design models directly related to the most important stakeholder requests.

The outcome of system definition is a description of the system that is both natural language and graphical. Some suggested formats for the description are provided in later sections.

> **Principle 55**
> **Write Natural Language Before A More Formal Model**
>
> If you write the formal model first, the tendency will be to write natural language that describes the model instead of the solution system. Consider the following examples:
>
> TO MAKE A LONG DISTANCE CALL, THE USER SHOULD LIFT THE PHONE. THE SYSTEM SHALL RESPOND WITH A DIAL TONE. THE USER SHOULD DIAL A "9". THE SYSTEM SHALL RESPOND WITH A DISTINCTIVE DIAL TONE…
>
> THE SYSTEM CONSISTS OF FOUR STATES: IDLE, DIAL TONE, DISTINCTIVE DIAL TONE, AND CONNECTED. TO GET FROM THE IDLE STATE TO THE DIAL TONE STATE, LIFT THE PHONE. TO GET FROM THE DIAL TONE STATE TO THE DISTINCTIVE DIAL TONE STATE, DIAL A "9."
>
> Note that in the latter example, the text does not help the reader at all.
>
> **- Alan M. Davis, 201 Principles of Software Development, 1995**

Key Skill 4: Managing the Scope of the Project

The scope of a project is defined by its requirements. Managing project scope to fit the available resources (time, people, and money) is a key to managing successful projects. Managing scope is a continuous activity made easier by that requires iterative or incremental development – which by definition breaks project scope into smaller more manageable pieces. – and the effective application of other requirements management skills.

For example, Using requirement attributes, such as (e.g., priority, effort, and risk,) as the basis for negotiating the inclusion of a requirement in an incremental release is a particular useful technique for managing scope. Focusing on the attributes rather than the requirements themselves helps desensitize negotiations that are otherwise often contentious.

Formatted

It is also helpful for team leaders to be trained in negotiation skills and for the project to have a champion in the organization, as well as on the customer side. Product/pProject champions, whether on the project team or not, should have the organizational power to refuse scope changes beyond the available resources or to expand resources to accommodate additional scope.

Key Skill 5: Refining the System Definition

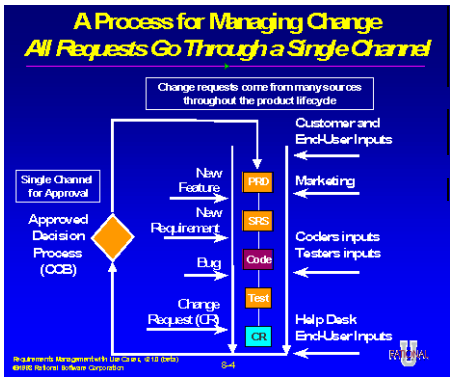
With Given an agreed upon high-level system definition and a fairly well understood initial scope, it is both possible and economical to invest resources in more refined system definitions. Refining the system definition includes two key considerations: developing both more detailed descriptions of the high-level system definition, and as well as descriptions to verifying that the system will comply with stakeholder needs and behave as described.

These descriptions are often the critical reference materials for project teams. Descriptions are best done with the audience(s) in mind. A common mistake is to represent what is complex to build with a complex definition, particularly when the audience may be unable or unwilling to invest the critical thinking necessary to gain agreement. This leads to difficulties in explaining the purpose of the project and the system to people both inside and outside the project team. Instead, you may discover the need to produce different kinds of descriptions for different audiences. This paper includes sSuggested formats for detailed natural language, and formal text, and as well as graphical descriptions are provided later in this paper. Once the description format is established, refinement continues throughout the project lifecycle.

Key Skill 6: Managing Changing Requirements

No matter how carefully you define your requirements, they will change. In fact, some requirements change is desirable! It means that your team is engaging your stakeholders. Accommodating changing requirements is a

measure of your team's stakeholder sensitivity and operational flexibility – team attributes that contribute to successful projects. Change is not the enemy, ***unmanaged*** change is.



~~What makes changing requirements hard to manage is not only that a~~A changed requirement means that more or less time has to be spent on implementing a particular ~~new~~ feature, and~~but also that~~ a change to one requirement may have an impact on other requirements ~~and on other items~~. ~~M~~anaging requirement change includes activities such as establishing a baseline, keeping track of the history of each requirement, determining which dependencies are important to trace, establishing traceable relationships between related items, and maintaining version control. As Figure 3 illustrates, it is also important to establish a change ~~–~~control or approval process, requiring all proposed changes to be reviewed by designated team members. Sometimes this single channel of change control is called a Change Control Board (CCB).~~.~~

Important Requirements Concepts

To apply requirements management skills ~~in~~ to a project, certain requirements management concepts are useful for everyone on the project to understand. They include:

Fig~~ure~~ 3

~~**Requirement Type.**~~

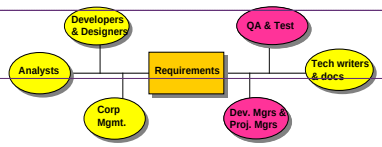
The larger and more intricate the system ~~developed~~, the more ~~expressions, or~~ types, of requirements appear. A requirement type is simply a class of requirements. By identifying types of requirements, teams can organize large numbers of requirements into meaningful and more manageable groups. Establishing different types of requirements in a project~~, or in a standard process,~~ helps team members classify requests for changes and communicate more clearly ~~(i.e., "this is a new Stakeholder Requirement" or "this is a change to an existing Use Case or Software requirement")~~.

~~U~~Usually, one type of requirement can be broken down, or decomposed, into other types. Business rules and ~~"~~vision~~"~~ statements can be types of high-level requirements from which teams derive user needs, features, and product requirement types. Use cases and other forms of modeling drive **design requirements** that can be decomposed to ~~**functional and non-functional** **software requirements**~~ and represented in analysis and design models ~~and diagrams~~. ~~**Test requirements** are~~ derived from the ~~above~~ software requirements and decompose to specific test procedures ~~and other testing assets~~. ~~In this simple description alone there are at least 10 different types of requirements to which the system is expected to conform.~~ When there are hundreds, thousands, or even tens of thousands of instances of requirements in a given project, classifying requirements into types makes the project more manageable.

~~**Cross-Functional Teams.**~~

Fig~~ure 43~~

Unlike other processes, such as testing or application modeling, which can be managed within a single business ~~function or even a subset of a function~~group, requirements management should involve everyone who can contribute their expertise to the development process. It should includes people who represent the customer and the business expectations. Development managers, product managers, analysts, systems engineers, and even customers should



Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted

participate. Requirements teams should also include those who create the system solution – engineers, architects, designers, programmers, technical writers, and other technical contributors. Testers and other QA personnel should be counted as are important team members.

Often, the responsibility for authoring and maintaining a requirement type of requirements can be allocated by functional area, further contributing to better large project management. The cross-functional nature of requirements management is one of the more challenging aspects of the discipline.

Traceability.

As implied in the description of requirement types, no single expression of a requirement stands alone. Stakeholder requests are related to the product features proposed to meet them. Product features are related to specific individual requirements that specify the features in terms of functional and non-functional behavior. Test cases are related to the requirements they verify and validate. R Some requirements may be are dependent on other requirements s or they may be or are mutually exclusive. In order for teams to determine the impact of changes and feel confident that assure themselves that their the system indeed conforms to expectations, as well as helping to determine the impact of changes, these traceability relationships must be understood, documented, and maintained. While traceability is one of the most difficult concepts to implement in requirements management, it is essential to accommodating change. Establishing clear requirement types and incorporating cross-functional participation can make traceability easier to implement and maintain.

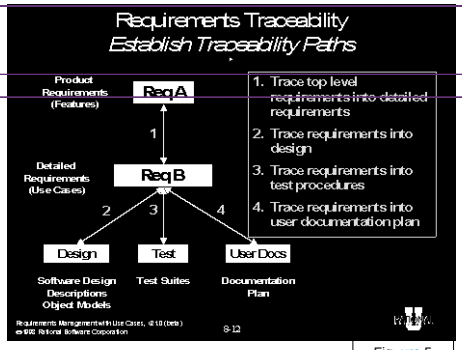


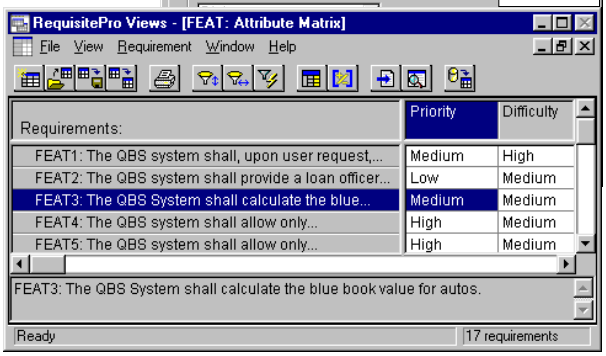
Fig ure 5

Multi-Dimensional Attributes.

Each type of requirement has attributes, and each individual requirement has different attribute values. These attributes and values are not readily apparent in the expression of the requirement itself. For example, requirements may be assigned priorities, identified by source and rationale, assigned priorities, delegated to specific sub-teams within a functional area, given a degree-of-difficulty designation, or associated with a particular iteration of the system. To illustrate, Figure 65 displays attributes for a Feature Requirement Type from a Learning Project in Rational's RequisitePro requirements management tool. As implied by the title of the screen, the requirement type and attributes for each type are defined for the entire project, ensuring usage consistency across the team.

In Figure 67, instances of feature requirements are displayed for a specific project in RequisitePro. Note that even without displaying the entire text for each requirement, in this view we can learn a great deal about each requirement from its attribute values. In this case, its priority and difficulty – no doubt assigned by different members of the team – – will help the team begin to scope the project to their available resources and

Fig ure 76



Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted

time, taking into account both stakeholder priorities and a very rough estimate of effort reflected in the difficulty attribute value. In more detailed types of requirements, the priority and effort attributes may have more specific values (e.g., estimated time, lines of code, etc.) with which to further refine scope. This multi-dimensional aspect of a requirement, compounded by different types of requirements – each with its own attributes – is essential to organizing large numbers of requirements and to managing the overall scope of the project.

Change History

Both individual requirements and a~~A~~ collection of r~~r~~equirements ~~and even each individual requirement~~ have histories that become meaningful over time. Change is inevitable and desirable to ~~maintain~~ keep pace with a changing environment and evolving technology. ~~Versioning~~ Recording the versions of project requirements ~~collectively~~ enables team leaders to capture the reasons for changing the project, ~~e.g.,~~such as a new system release. Understanding that a collection of requirements ~~at a given time~~ may be associated with a particular version of software allows you to manage change incrementally, reducing risk and improving the probability of meeting milestones. As individual requirements evolve, it is important to understand their history: what changed, why, when, and even by whose authorization.

Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted

Putting Requirements Management to Work

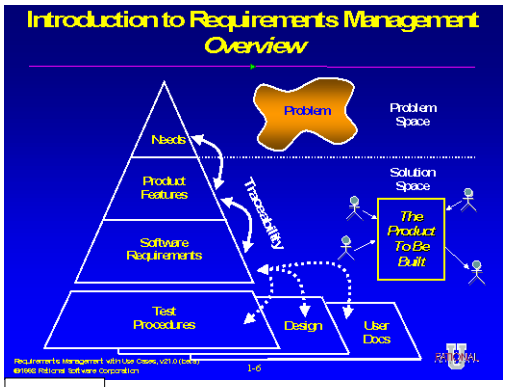


Fig-ure 87

Requirements management employs the key skills and concepts presented above to identify and resolve the identified problems successfully, as defined by the solution's conformance to some set of requirements.

To build a system that truly meets customers' needs, the project team must first define the problem to be solved by the system. Next, the team must, while identifying stakeholders from whom business and user needs are elicited, described, and prioritized. From this set of high-level expectations or needs, a set of product or system features should be agreed upon.

Detailed software requirements should be written in such a form as can be understood by both the customers and the development team. We have found that using the language of the customer to describe these software requirements (e.g., in use cases) is most effective in gaining the understanding and agreement that is necessary for the successful completion of the project. These detailed software requirements are then used as input for the system design specifications as well as for test plans and procedures needed for implementation and validation. Software requirements should also drive the initial user documentation planning and design.

To facilitate this, the project team should:

- Agree on a common vocabulary for the project.
- Develop a vision of the system that describes the problem to be solved by the system, as well as its primary features.
- Elicit stakeholder's stakeholders needs in at least five important areas: functionality, usability, reliability, performance, and supportability.
- Determine what requirement types to use.
- Select attributes and values for each requirement type.
- Choose the formats in which requirements are described.
- Identify team members who will author, contribute to, or simply view one or more types of requirements.
- Decide what traceability is needed.
- Establish a procedure to propose, review, and resolve changes to requirements.
- Develop a mechanism to track requirement history.
- Create progress and status reports for team members and management. Manage dependencies between requirements by using a carefully selected set of requirements attributes, as well as traceability dependencies between requirements.
- Delimit the boundaries of the system.

A Few Words about Documents

The decision to describe requirements in documents deserves some thought. On the one hand, writing is a widely accepted form of communication and, for most people, a natural thing to do. On the other hand, the goal of the project is to produce a system, not documents.

Common sense and experience teach that the decision is not whether but *how* to document requirements. With an effective requirements management process that establishes what to document, document templates that show how to write them, and document-and-database-centric tools such as Rational's RequisitePro to facilitate the management of requirements contained within, documents are invaluable vehicles for requirements capture and communication. Document templates provide

Formatted

Formatted

Formatted: Bullets and Numbering

Formatted

- ~~Detail your understanding of the requirements by:~~
  - ~~Detailing their specifications and make them unambiguous and verifiable.~~
  - ~~Structuring the requirements to make them easier to comprehend.~~
  - ~~Developing a user-interface model or specification.~~
  - ~~Developing a user-interface prototype.~~
- ~~Prioritize and develop requirements in increments based on risk, architectural significance, and architectural coverage.~~
- ~~Continuously feedback results to stakeholders and other team members by conducting reviews of the results.~~

~~To support these development activities you need to have the following in place:~~

- ~~Guidelines for what types of requirements to use, what requirements attributes to use, and to what detail traceabilities should be managed.~~
- ~~Templates and samples for how requirements of different types should be described.~~
- ~~A procedure for managing change requests.~~
- ~~Tools or mechanisms for tracking requirements and their history, and for producing progress reports on the development activities.~~
- ~~Procedures for establishing clear responsibilities within the development team: who will author, contribute to, or simply view a particular set of requirements.~~

These essential requirements~~ ~~management activities are independent of industry, development methodology, or requirements tool~~s~~. They are also flexible, enabling effective requirements management in the most rigorous and the most rapid application development environments.

Requirements Management Workflows

Requirements management can follow an infinite number of domain-specific paths. The following approach prescribes six detailed workflows that apply to each of the key requirements~~ ~~management skills~~, yet are configurable enough to~~but can be applied to any domain.[*]

Workflow: ~~The~~ Problem Analysis ~~Workflow~~

In the **Problem Analysis** workflow, the primary activity is ***vision development***. Output from this activity is a ***vision document*** that identifies the high-level user or customer view of the system to be built. The vision expresses initial requirements as key features the system must possess in order to solve the most critical problems. The ***system analyst*** has the primary role in this workflow. The system analyst should have problem domain expertise and an understanding the problem, and should be able to describe a process that he or she believes will solve the problem. Active involvement from various project stakeholders is required.

[*] The following workflow diagrams are from the Rational Unified Process~~.~~ Requirements Workflow. The workflows are expressed in terms of workers, activities and artifacts (input or output). The accompanying text in this paper describes ~~the detail of~~ each workflow briefly, in the hopes of stimulating your thoughts and interest in improving your requirements management process. More information on the Rational Unified Process can be found at www.rational.com.

Formatted: Bullets and Numbering

Formatted

To begin managing dependencies, features should be assigned attributes such as rationale, relative value or priority, and source of request. As the vision develops, the analyst identifies users and systems (the actors) of possible use cases. Actors are the first element of the use-case model, which will define the system's functional and non-functional technical requirements.

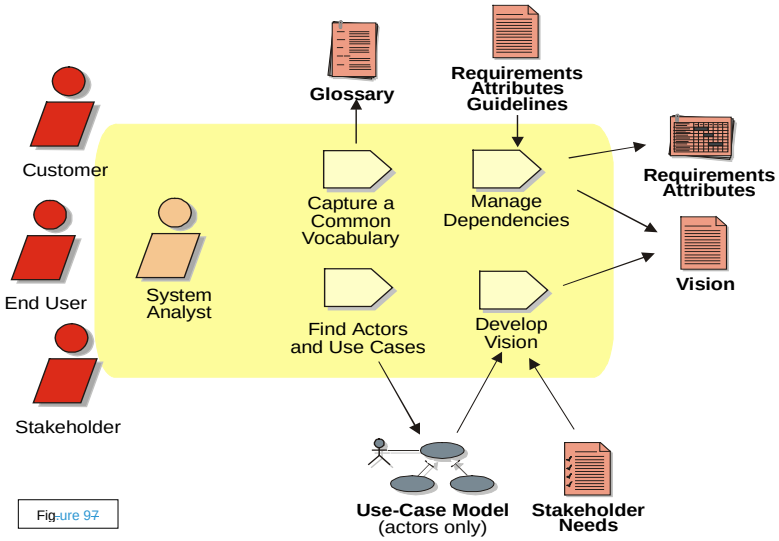


Figure 97

Overview

Use-Case Model Introduction

A use-case model consists of actors, use cases, and relations among them. *Actors* represent everything that must exchange information with the system, including what are usually typically called users. When an actor uses the system, the system performs a *use case*. A good use case is a sequence of transactions that yields a measurable result of value for an actor. The collection of use cases is the system's complete functionality.

Jacobson I., Christerson M., Jonsson P., Overgaard G.,
Object-Oriented Software Engineering – A Use Case Driven Approach, Addison Wesley – ACM Press, 1992

Formatted

In the Problem Analysis Wworkflow, the primary activity is *Develop Visionvision development*. Output from this activity is a *vVision document* that identifies the high-level user or customer view of the system to be built. In tThe vision, initial requirements are expressesd as initial requirements as key features the system must possess in order to solve the most critical problems. The *main worker or role involved is the sSystem aAnalyst* has the primary role in this workflow. The right person for the job should have, who is someone with problem domain expertise, should who understands the problem, and should be able tocan describe a process that he or she they believes will solve the problem. Active involvement from various project stakeholders to the project is required.

Formatted

Formatted

Formatted

To begin managing dependencies, he features should be assigned attributes such as rationale, relative value or priority, and source of request and so on, so that dependencies can begin to be managed. As the vVision develops, the analyst identifies users and systems – the "actors." of possible uUse cCases. Actors ***Actors*** are the first element of the uUse- cCase mModel, which that, when completed, will define the system's functional and non-functional technical requirements.

Formatted

Formatted

Formatted

The Activities in Problem Analysis

Initiation: One or more stakeholders who perceive a problem will initiate the workflow.

One or ~~a few of the~~more system analysts ~~thoin se within the~~a development team ~~that act as system analysts~~ conduct a session ~~where they~~ to help the initial stakeholders describe the problem they want solved. The elements of the vision document are organized in~~you work with here is the problem statement, often organized as in~~ the following table:

~~The p~~P~~roblem of~~	(~~describe~~ define the problem~~)~~
Affected Stakeholders	(list the stakeholders affected by the problem~~)~~
~~I~~I~~The i~~mpact ~~of which is~~	(~~describe what is~~ the impact of the problem~~)~~
~~SA s~~uccessful Solution ~~would~~	(list some key benefits of a successful solution~~)~~

The problem statements succinctly explain~~capture~~ the purpose of the project. Problem analysis stimulates further investigation into all stakeholder needs and the initial business case including compelling benefits and roughly estimated costs.

In parallel with defining problem statements, you should also compile a glossary by keeping ~~keep~~ track of commonly used terms and ~~and~~ agree~~e~~ing on their definitions. ~~This is captured in a ***glossary*** for the project~~.

Problem analysis also identifies the ma~~M~~ain system actors ~~are identified during problem analysis~~. ~~A~~Actors are users of the system ~~and~~ or any other system that ~~thing else that~~ will exchange information with it~~, including other systems~~.

Use-Case Model Introduction

A use-case model consists of actors, use cases, and relations among them. *Actors* represent everything that must exchange information with the system, including what are ~~usually~~ typically called users. When an actor uses the system, the system performs a *use case*. A good use case is a sequence of transactions that yields a measurable result of value for an actor. The collection of use cases is the system's complete functionality.

Jacobson I., Christerson M., Jonsson P., Overgaard G., Object-Oriented Software Engineering – A Use Case Driven Approach, Addison Wesley – ACM Press, 1992

~~Some of the obvious ways that the actors will interact with the system are identified in problem analysis.~~ At this stage, problem analysis should briefly identify some obvious ways that the actors will interact with the system~~these are~~. Descriptions should be ~~described very briefly, perhaps with a name and description only, and are more~~ oriented towards business process rather than system behavior. For example, a budgeting program may allow one type of actor to "Create departmental budget~~,~~", while another actor will be able to "Consolidate departmental budgets.". ~~When these use cases are described in more detail later t~~The system analyst may later break them into additional use cases that align more meaningfully with specific system behavior. For example, "Create departmental budget" could result in system use cases such as~~,~~ "Import spreadsheet information" and "Create budget views.".

The problem analysis session described above is often performed more than once, maybe with different stakeholders, and intermingled with ~~sessions "more "~~internal~~"" to the~~ development team sessions. The system analyst who conducted the ~~"external" session will lead a meeting~~meeting with the stakeholders will lead a session with members~~teammates from~~ of the development team to envision a technical solution to the problems, ~~deriving~~ derive features from the initial stakeholder inputs, and draft~~ing~~ the vision description, the first definition of the system to be built. To facilitate understanding of the proposed solution among the initial stakeholders, the system analyst may use modeling tools or manual drawing techniques to complement the vision description.

The initiating stakeholders are consulted at multiple points to help refine the problem description and constrain the number and scope of possible solutions. ~~Constraining scope at the vision level of requirements is~~

Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted
Formatted

~~represented by *the Manage Dependencies activity* in the workflow.~~ Stakeholders and system analysts manage dependencies in this workflow by negotiating the priority of key features and gaining a general~~, rough~~ understanding of the resources and effort needed to develop them. While priority and effort/resource estimates inevitably change ~~significantly as the stakeholder needs are better understood and systems are defined in more detail~~, managing dependencies early establishes an important pattern that continues throughout the development lifecycle. It is the essence of the ~~managing~~ scope management ~~w~~workflow and an early predictor of project success.

~~As with any collaborative workflow, the resulting artifacts will evolve perceived availability of time, people, and money, constrain the size of the problem to be tackled and the solution to be built.~~ After several drafts, ~~There are several drafts of the vision. In parallel, as~~ the vision reaches a point when the team must decide whether to ~~state at which the decision to~~ invest in additional requirements elicitation. By the same time, ~~can be made,~~ the business case approval process has been ~~is~~ initiated separately. ~~The~~ Although not addressed further in this paper, the business case describes:

- ~~the context~~ (product domain, market, and scope),
- the technical approach,
- the management approach (schedule, risk, objective measure of success),
- and the financial forecast.

~~Business case development is not a subject of this paper.~~

Workflow: Understanding Stakeholder Needs ~~*Workflow*~~

If the initial vision justifies additional investment, the **Understanding Stakeholder Needs** workflow begins in earnest. The key activity is *eliciting stakeholder needs.* The primary outputs are collections of *prioritized stakeholder needs*, which enable refinement of the vision document, as well as a better understanding of the requirements attributes. Also, during this workflow you may start discussing the system in terms of its use cases and actors. Another important output is an updated glossary of terms to facilitate common vocabulary among team members.

Formatted

Formatted

Formatted

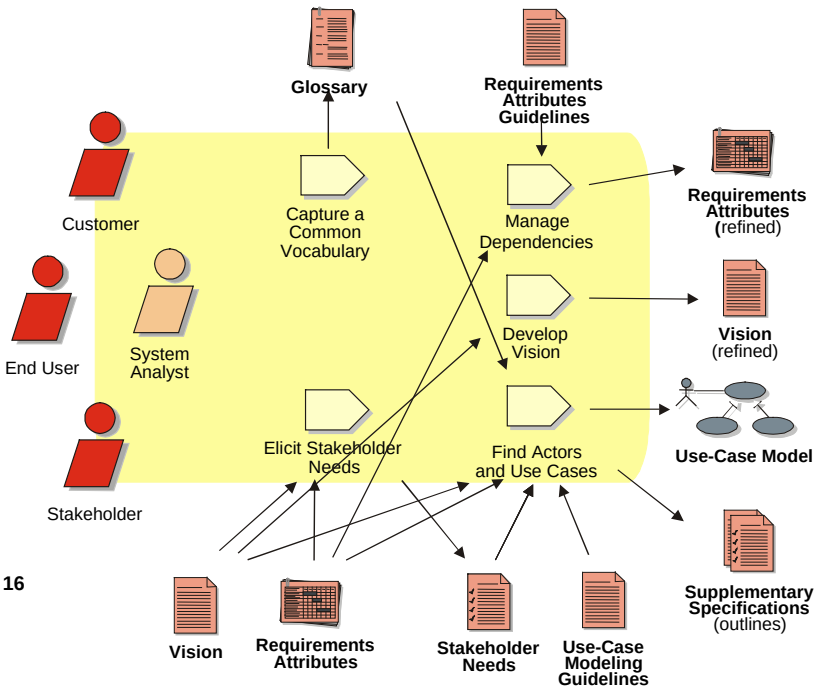
Formatted: Bullets and Numbering

Formatted

Formatted

Formatted

Figure 10


Glossary
Requirements Attributes Guidelines
Customer
Capture a Common Vocabulary
Manage Dependencies
Requirements Attributes (refined)
End User
System Analyst
Develop Vision
Vision (refined)
Elicit Stakeholder Needs
Find Actors and Use Cases
Use-Case Model
Stakeholder
Supplementary Specifications (outlines)
Vision
Requirements Attributes
Stakeholder Needs
Use-Case Modeling Guidelines

~~Overview~~

~~If the initial vision justifies additional investment, the Understanding Stakeholder Needs W~~workflow begins in earnest. The key activity is ~~e***Elicit ing stakeholder needs***~~.~~ The primary outputs are collection~~(s)~~ of ~~***prioritized stakeholder needs***~~, which enable refinement of the vision document, as well as a better understanding of the requirements attributes. Also, during this workflow you may start discussing the system in terms of its use cases and actors. Another important output is an updated glossary of terms to facilitate common vocabulary among team members.~~

Formatted

Formatted

Formatted

Formatted

Formatted

Activities in Understanding Stakeholder Needs

Understanding Stakeholder Needs: Where "Delighting Customers" Begins

Stakeholder nNeeds are a type of proposed requirement captured as much as possible in the language and format of the submitting stakeholder. Unlike subsequent requirement types that are usually authored by process-educated and technically proficient project team members, sStakeholder nNeeds are often expressed poorly. They are~~,~~ duplicated or overlap.~~,~~ ~~t~~They ~~and~~ can be expressed on anything from slips of paper to enhancement~~ ~~-~~request databases.

The analyst (or team representing the analyst role) must review them all, interpreting, grouping, perhaps re-typing (without re-writing), and translating them into features in the vVision description. Depending on the rigor applied in your development and the availability of tools, traceability between some or all stakeholder needs and features can be applied to help stakeholders understand how their needs were taken into account.

Demonstrating serious concern for eliciting and satisfying stakeholder needs by applying the Understanding Stakeholder Needs Workflow can be critical to establishing stakeholder confidence in your team's abilities.

Formatted

The system analyst and key stakeholders identify additional stakeholders and elicit their needs via interviews, workshops, storyboards, business process use cases, and other ~~elicitation~~ techniques. One or ~~a few individuals acting as~~more system analysts facilitate these sSessions ~~of this kind are facilitated by one or a few individuals acting as system analysts~~. Requirements are among the most useful elicitation techniques. ~~-~~The process includes uUsers, help~~-~~desk personnel, business owners, testers, and others who have a stake in the outcome of the proposed project ~~are involved~~. ~~Their requests are captured as they are made.~~ Stakeholder needs are often ambiguous, overlapping, and even extraneous. In addition to formal elicitation results, ~~they~~ stakeholder needs may ~~arrive~~ be expressed in~~as~~ well-formatted documents, defect and enhancement requests from databases, or e-mail and groupware threads. System analysts~~As stakeholder~~ record, ~~and~~ categorize, and prioritize stakeholder needs ~~and the priority of each need~~~~Needs are captured and categorized, system analysts also capture their relative priority as represented by the submitting stakeholder, and their source~~.

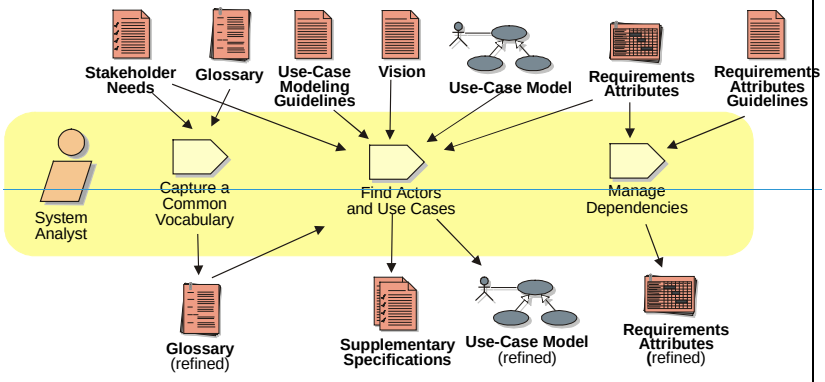
Based on a better understanding of stakeholder needs, the system analysts in the development team ~~will~~ refine the vision document, paying special attention to ~~. An important part of the refined vision is~~ the product position statement. In two or three sentences, t~~T~~his statement establishes ~~in only two or three sentences,~~ the compelling value of the project. The statement should include ~~: i~~for whom it is ~~i~~ntended users, the problems it solves, the benefits it delivers, and the competitors it replaces~~the alternative it is better than.~~. All team members should understand this ~~It is the~~ project theme ~~all team members should understand~~.

Formatted

~~The s~~System analysts also updates the glossary to facilitate common understanding of terms.

Key stakeholders are consulted at multiple points to negotiate priority of new features derived from understanding stakeholder needs and gain a current understanding of resources and effort needed to develop them. As with problem analysis, managing dependencies in this workflow helps manage scope. ~~When appropriate, it further~~It also establishes traceability between stakeholder needs and ~~derived~~ vision features ~~in the vision~~, so ~~that stakeholders~~stakeholders can be sure their inputs were considered.

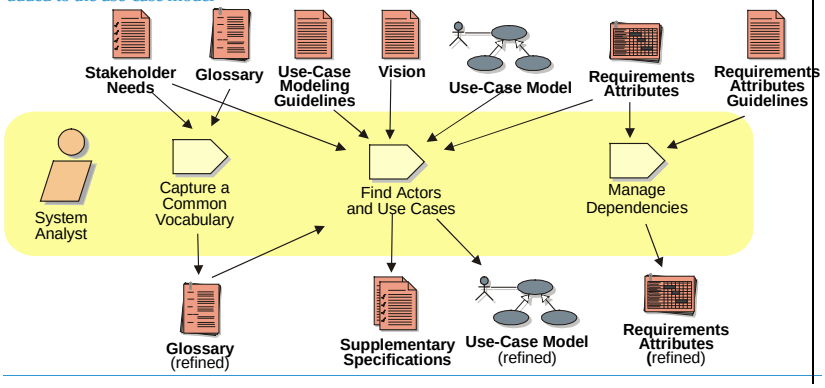
Workflow: Defining the System ~~Workflow~~



~~Overview~~

The **Problem Analysis** workflow and the **Understanding Stakeholder Needs** ~~w~~workflow~~s~~ create early iterations of key system definitions, including the vision document, a first outline to the use-case model, and the requirements attributes. The **Defining the System** ~~w~~workflow *completes the description* of the system-level requirements with the addition of new actors, ~~and~~ use cases, and supplementary specifications.

~~added to the use-case model~~



Fig~~-~~ure 11~~7~~

Formatted
Formatted
Formatted
Formatted
Formatted

Activities in Defining the System

The glossary is updated to reflect current understanding about the terms used to describe features and the use-case model.

The system analyst uses the refined ~~Vision~~vision to derive and describe the ~~Use Case~~use case~~s~~ that elaborate on the system's expected behavior.

The ~~Use-Case Model~~use-case model serves as a contract between the customer, the users, and the system developers. It ~~, a~~allows~~ing c~~defines expectations for system developers and helps customers and users to validate that the system will ~~become what they expected and allowing~~ allows system developers to build what ~~is expected~~meet these expectations.

The ~~System Analysts~~system analyst describes requirements that do not fit well in ~~Use Case~~use cases in a supplementary specification. Usability, reliability, performance, ~~-~~and supportability requirements often end up here. It should be noted that many non-functional requirements of these types are specific to a single use case. It is better for use~~-~~ case authors to place these ~~use-case specific non-functional~~ requirements in the use case specification itself (see the **Refining the System** workflow), leaving the supplementary specification for ~~*more*~~ *global* non-functional requirements.

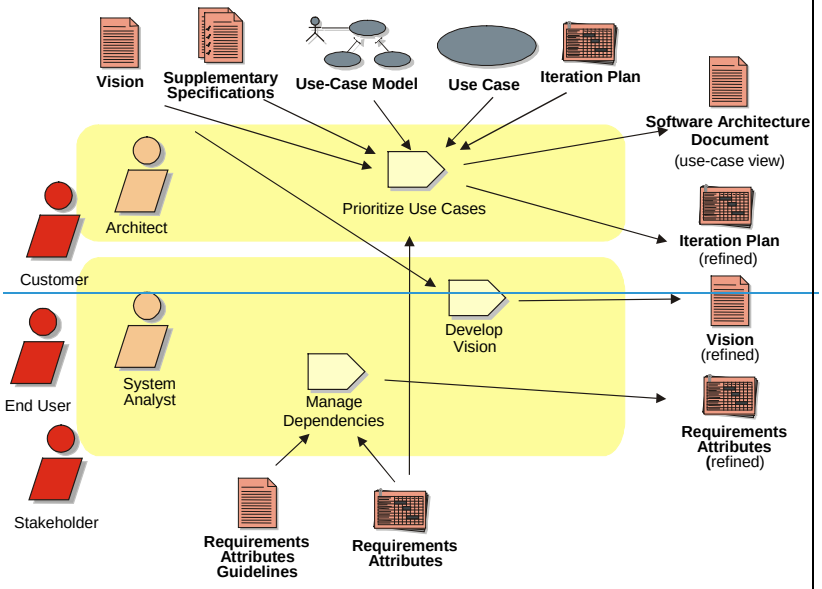
In this workflow, t~~Activities also include c~~~~C~~~~reating~~he system analyst creates~~e~~ attributes for the supplementary requirements (such as priority and ~~, including priority,~~ related use cases). In addition, the system analyst~~,, etc.,~~ adds and updat~~esing~~e attribute values for the initial and new use cases~~, and adding values to the attributes for new use cases~~.

Finally, the system analyst m~~M~~anages dependencies by tracing important user needs and critical features to related use cases and supplementary specifications.

Formatted

Formatted

Workflow: Managing Scope ~~Workflow~~



~~Overview~~

~~Although project scope should be managed continuously, the better understanding of system functionality obtained from~~ Identifying most actors, use cases, and supplementary specifications allows the ~~System Analyst~~system analyst to apply priority, effort, cost, and risk values ~~etc.,~~ to requirements attributes more accurately. This better understanding also~~, and~~ enables the architect to identify the architecturally significant use cases.

The ~~*An input to Managing Scope not seen in other requirements workflows is the I*~~*iteration p*~~*P*~~*lan*, developed in parallel by project and development management, first appears in the **Managing Scope** workflow. Also known as a development plan, the i~~The I~~teration ~~P~~plan ~~or development plan~~ defines the number and frequency of iterations planned for the release. ~~The scope of the project defined in Managing Scope will have a significant impact on the iteration plan~~T~~ as t~~he highest risk elements within scope should ~~will~~ be planned for early iterations.

Other important outputs from the **Managing Scope** workflow include the initial iteration of the *software architecture document*[*] and a *revised ~~Vision~~vision* that reflects analysts and key stakeholders' ~~better~~ increased understanding of system functionality and project resources.

[*] Like the business case earlier and first issue of the ~~i~~Iteration plan, the software architecture document is not an artifact of requirements management workflows, although it is related and is part of the Rational Unified Process. It is not the subject of this paper.

Experience teaches that the keys to managing scope successfully are the well-considered attribute values ass… sta… use… supplementary specifications; and regular, open, and honest interaction with representative stakeholders

Formatted

Formatted

Formatted

Formatted

Formatted

Formatted

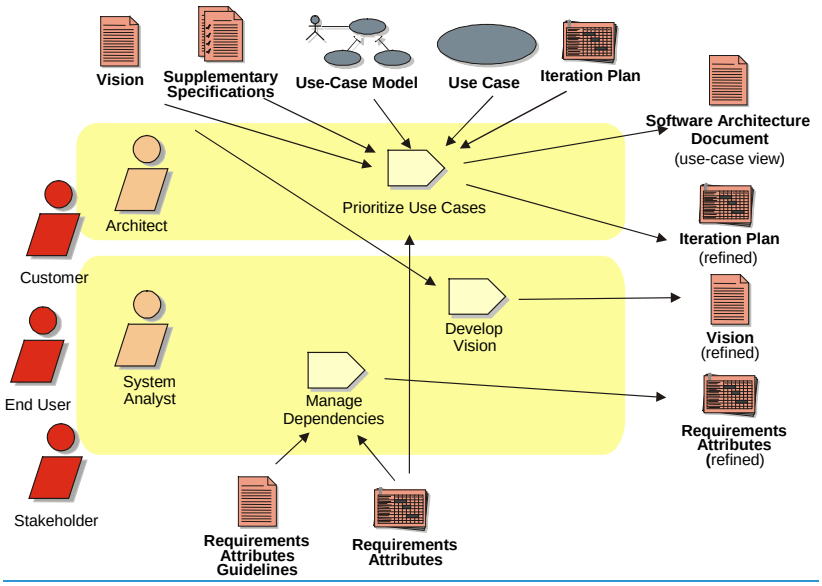

Vision
Supplementary Specifications
Use-Case Model
Use Case
Iteration Plan
Software Architecture Document
(use-case view)
Architect
Prioritize Use Cases
Iteration Plan
(refined)
Customer
Develop Vision
Vision
(refined)
End User
System Analyst
Manage Dependencies
Requirements Attributes
(refined)
Stakeholder
Requirements Attributes Guidelines
Requirements Attributes

Figure 127

Activities in Managing Scope

Architects prioritize use cases for their risk coverage, architectural significance, and architectural coverage. While the system may be defined with many use-case and supplementary specification requirements, only a subset of use cases are usually critical to good system architecture. With prioritized use cases, architects refine the iteration or development plan and model a use-case view of the system architecture in tools such as Rational Rose.

~~System Analyst~~System analysts manage dependencies by refining requirements attributes for features in the ~~Vision~~vision. They also refine ~~and~~ requirements in use cases and supplementary specifications. System analysts ~~and~~ ensure that appropriate traceability* exists for ~~agreed to~~ stakeholder needs, features, use-case requirements, and supplementary specification requirements.

~~System Analyst~~System analysts negotiate revised project scope and ~~Vision~~vision with key stakeholders. This step is among the most important in the entire project. For the first time the **breadth** of knowledge about the proposed system is available to make serious commitments on requirements, project resources and delivery dates. At the same time, it must be understood that these requirements ~~y~~ will change as the **depth** of knowledge increases. ~~This step is much more easily done and future changes will be more easily accommodated if~~ If dependencies have been managed in the previous three workflows, this step is much easier, and future changes will be easier~~problem analysis, understanding stakeholder needs, and defining the system workflows~~.

~~Successfully m~~Managing scope to match available resources is ~~probable~~ successful only if stakeholders and development team members view this step as a natural progression ~~–, and~~ not an ambush on users' expect~~ed system functionality nor~~ations or~~, from the stakeholders' untrusting view,~~ an attempt to blackmail the organization for more time and money. This workflow will need to be repeated ~~at least at~~at major milestones in the project to assess whether ~~your better understanding of~~new insight into the system and its problems requires ~~that you~~ change to the scope. While committed requirements, budgets, and deadlines are hard to change, an in-depth understanding of prioritized use cases, ~~and~~ supplementary specifications, a~~nds well as~~ early system iterations, ~~–~~ inevitably lead to scope reconsideration.

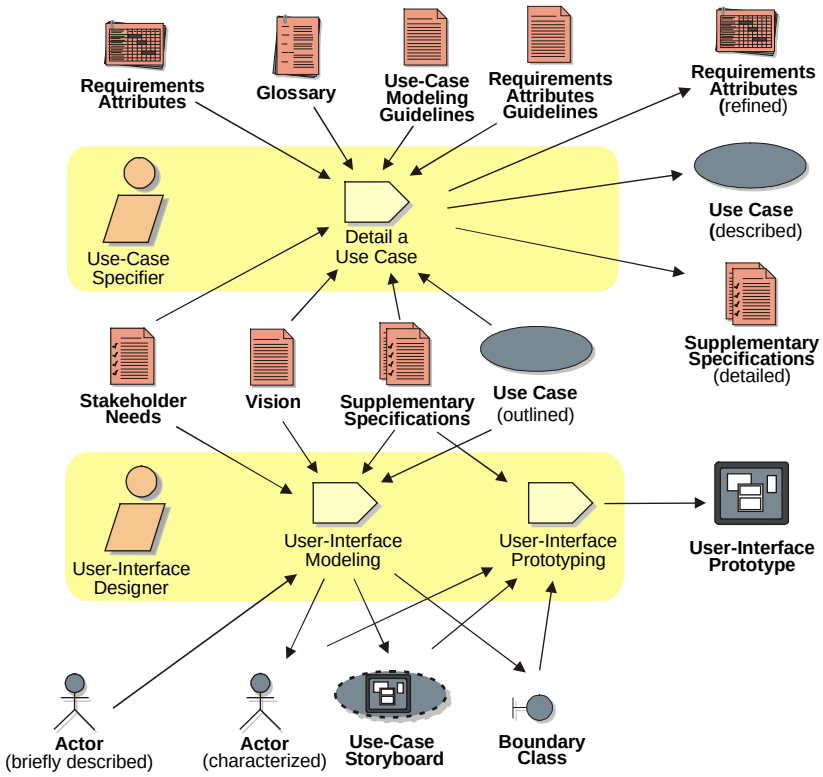
Once again, ~~As mentioned earlier,~~ it is critical that the project team engages in habitual scope management before reaching the refinement stage. R~~, and that r~~epresentative stakeholders must understand and trust that their priorities and interests are taken seriously during increasingly difficult scope negotiations. By the time system requirements are refined, ~~there~~ only ~~are~~ ~~**ONLY**~~ important requirements remain to be negotiated ~~out~~ or modified. Unless effective scope management habits ~~trust~~ ha~~s~~ve been established ~~through effective scope management habits~~, your project may be doomed as a ~~it is at this point that~~ "death march~~es~~" – a ~~projects~~ hopelessly over-scoped project moving inexorably towards delays and cost overruns ~~— begin~~.

Formatted

* The term "appropriate traceability" is deliberate. See the inset text on ~~the *tao* of~~T~~t~~raceability later in this paper.

Workflow: Refining the System ~~Workflow~~

The **Refining the System** workflow assumes that system-level use cases have been outlined and actors have been described, at least briefly. Through managing the project scope, the features in the vision have been re-prioritized and are now believed to be achievable by fairly firm budgets and dates. The output of this workflow is a more in-depth understanding of system functionality expressed in **detailed use cases**, **revised supplementary specifications**, and **early iterations of the system** itself[*].



Fig~~-~~ure 13~~7~~

~~Overview~~

~~The **Refining the System** workflow begins with system-level use cases outlined and , actors described at least briefly., and a revised understanding of project scope reflected in r~~Re-prioritized features in the ~~Vision~~vision ~~that is believed to be achievable by fairly firm budgets and dates—reflect a revised understanding of project scope. The output of this workflow is a more in-depth understanding of system functionality expressed in~~

Formatted

[*] Obviously, not all systems will have user interfaces and not all early iterations will include GUI elements ~~if they are not considered risky~~. We use them~~it~~ here only as an example of an early iteration. ~~A partial list of o~~Other examples includes prototypes, models, and storyboards.

detailed Use Caseuse cases**, revised supplementary specifications**, and **early iterations of the system** itself represented in the workflow example above by user-interface elemitselfnts*.

Activities in Refining the System

The ***use-case specifier*** details the use cases by detailing the definition of their flow of events, pre- and post-conditions, and other textual properties of the each use case. To minimize the effort and enhance the readability, it is advisable to use a standard document format or template, a use -case specification template, to capture textual information about each use case. Creating well-thought-out use -case specifications is critical to the quality of the system. T, and the act of creating them the specifications usually forces the author tois specification development requires a thorough understanding thoroughly of the stakeholder needs and features requirements related to the use case they are specifying. Because Since it takes some effort to create good use cases, and because and creating them is so valuable to an in-depth understanding of the system to be built, iIt is both common and desirable to have several members of the project team (such as, for example software engineers), create themparticipate in creating the use cases.

In parallel, the use -case specifier revises the supplementary specification with additional non-use-case-specific requirements that emerge with a better understanding ofare note specific to the use case.

The ***user-interface designer*** will work to models and prototypes the user interface of the system, as well as building a prototype of the user interface. This work is highly correlated to how the evolution of the use cases evolve.

The use-case specifier and the System Analystsystem analyst revise the effort, cost, risk, and other attribute values for each requirement that is understood better.

The result of refining thethis system refinement is submitted towill go through be input to another round of the **Managing Scope** wworkflow. Once you know more about the system, you may realize you want to change the priorities.

Workflow: Managing Requirement Change Workflow

Like the **Managing Scope** workflow, the **Managing Requirements Change** workflow should be applied continuously. The output of -this workflow can cause modification to every artifact, which requires effective communication with all project team members and stakeholders.

In this workflow we ***introduce additional artifacts*** that are affected by requirements workflows. Changes to requirements naturally affect the system models that represent them in the analysis and design workflows. Requirement changes also affect tests created to validate the proper implementation of the requirements*. Traceability relationships identified in the process of managing dependencies are the keys to understanding these impacts.

The *tao of* Traceability

Much is made of traceability in the requirements field. Many promote the virtue of tracing individual customer requirements to each related specification, test, model element, and ultimately source code files. It is an attractive and achievable vision with the right suite of integrated tools. Certainly, some traceability is the key to successful requirements change management.

Be forewarned, however, that all forms of traceability require an investment to set up and maintain during the life of the project. Like all investments, traceability has diminishing points of return depending on your specific situation. This paper emphasizes the value of tracing between types of requirements. It This is a good place to start and wellcan be -automated by tools such as Rational's RequisitePro. We believe you will find some level of requirements

* Obviously, not all systems will have user interfaces and not all early iterations will include GUI elements if they are not considered risky. We use it here only as an example of an early iteration. A partial list of other examples includes prototypes, models, and storyboards.

* As in earlier examples, these artifacts are part of the Rational Unified Process but are not the subjects of this paper.

Formatted

Formatted

Formatted

Formatted

Formatted

Formatted

Another important concept for Managing Requirements Change Workflow is ***requirement history tracking***. By capturing the nature and rationale of requirements changes, reviewers (anyone on the software project team whose work is affected by the change) receive the information needed to respond to the change properly.

Another important concept for Managing Requirements Change Workflow is ***requirement history tracking***. By capturing the nature and rationale of requirements changes, reviewers (anyone on the software project team whose work is affected by the change) receive the information needed to respond to the change properly.

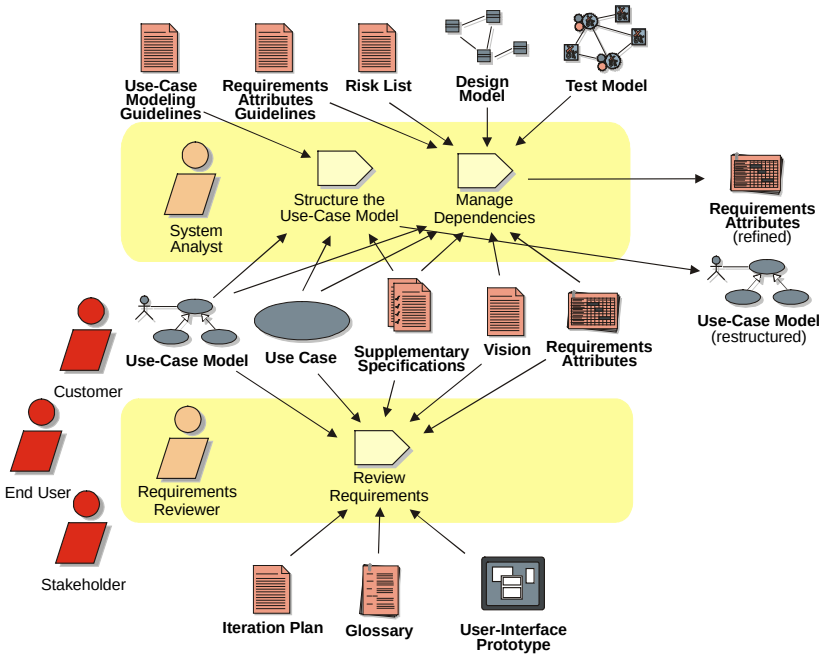


Fig.ure 14.7

~~Overview~~

~~Like the Managing Scope W~~~~w~~~~orkflow, the Managing Requirements Change W~~~~w~~~~orkflow should be applied continuously. The output of the Managing Requirements Change Workflow can be modification to every artifact, which requires~~ing~~ effective communication with all project team members and stakeholders.~~

~~In this workflow we ***introduce additional artifacts*** that are affected affected by requirements workflows. Specifically, changes to requirements naturally impact affect the system models that represent them in analysis and design workflows of software engineering. Requirements changes , and also affect~~quite naturally ~~impact tests created to validate the proper implementation of the requirements in automated testing~~

The *tao* of Traceability

Much is made of traceability in the requirements field. Many promote the virtue of tracing individual customer requirements to each related specification, test, model element, and ultimately source code files. It is an attractive and achievable vision with the right suite of integrated tools. Certainly, some traceability is the key to successful requirements change management.

Be forewarned, however, that all forms of traceability require an investment to set up and maintain during the life of the project. Like all investments, traceability has diminishing points of return depending on your specific situation. This paper emphasizes the value of tracing between types of requirements. It is a good place to start and well-automated by tools such as Rational's RequisitePro. We believe you will find some level of requirements traceability to be a good investment…and you, too, may decide to pursue the *tao*!

Formatted

Formatted

requirementworkflows*. Traceability relationships between requirements identified in the manage dependencies activity of this workflow and others are the keys to understanding these impacts.

Another important concept for Managing Requirements Change Workflow is ***the tracking of requirement history tracking***. By capturing the nature and rationale of requirements changes, reviewers (in this case the role is played by anyone on the software project team whose work is affected by the change) receive the information needed to respond to the change properly.

Formatted

Activities in Managing Changing Requirements

Changes to requirements are initiated by any stakeholder or project team member for an infinite number of reasons.

The System Analystsystem analyst initiates a review activity, assimilating all change requests, and classifies them as:

- defects in implementation that do not affect requirements,
- modifications to existing requirements of some type, and
- new stakeholder needs or enhancement requests.

Formatted: Bullets and Numbering

Once classified, the proposed changes to requirements are assigned attributes and values as described in other requirements workflows.

In reviewing changes, the System Analystsystem analyst presents proposed prioritized requirements changes to a ***change control board*** comprised of representative stakeholders and/ or representative project team members empowered to approve changes. Scope modifications that exceed resources should be rejected or elevated to stakeholder representatives that are empowered to approve required changes to date and budget commitments.

Formatted

The change control board approves or rejects changes to requirements.

The System Analystsystem analyst communicates requirements changes to requirements authors or makes changes directly to requirements in the Visionvision, Use Caseuse cases, or supplementary specification documents.

The ***requirements reviewers*** (developers, testers, managers, and other project team members) evaluate the impact of changes to requirements on their work by reviewing requirement history. Finally, they and implement the change, including making and make appropriate changes to related requirements for which they have authority.

Formatted

Summary

The need to manage requirements is not new. So, what makes the preceding information worth considering now?

First, if your projects are not regularly satisfying customers, meeting deadlines, and staying within budget, you have reason to re-consider your development approach. If in so doing so, you determine that some of the requirements-related problems cited in this paper contribute to the unsatisfying state of affairs inare undermining your development efforts, you have reason to consider better requirements-management practices.

Second, the requirements -management practices summarized by the authors in this paper embody the collective experience of thousands, and are t. T, and the well-considered opinions of a smaller group of peoplenumber of individuals who have spent years working with customers in the field of requirements management*. We suggest that this overview of their contributions, -- and the more thorough presentation of

* As in earlier examples, these artifacts are part of the Rational Unified Process but are not the subjects of this paper.

* The authors would like to acknowledge the direct and in-direct contributions of Rational Fellow Ivar Jacobson, and those of Dean Leffingwell, Dr. Alan Davis, Ed Yourdon, and Elemer Magaziner, who helped define early iterations of Requirements Management key skills presented here. Most importantly, we appreciate the Rational Software customers who have applied and improved these practices in hundreds of development projects.

them made in Rational's Unified Process[5] ~~and complementary course on requirements management with Use Cases~~ –, represent a "best practice" in requirements management. You will not find more proven advice on requirements anywhere.

Finally, in the past two years Rational Software, a ~~credible~~ leader in the business of ~~supplying customers with~~producing effective software development solutions~~, Rational Software~~, has taken on the challenge of requirements management ~~challenge~~ and emerged with the tools to automate this difficult task. The chronic, pervasive problems of requirements management are solvable. And that, ultimately, may be the best reason to start practicing~~is what makes ity it's time for you to consider doing begin practicing~~ excellence in requirements management ~~better, today~~ today.

2þ

~~Bibliography~~End Notes

[1] *CHAOS*, The Standish Group International, Inc., Dennis, MA, 1994, 1997

[2] Computer Industry Daily, December 12, 1997

[3] Dorfman, M. and R. Thayer, Software Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1997
pp. 79

[4] Dorfman, M. and R. Thayer, Software Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1997 pp. 80

[5] Rational Unified Process™, Rational Software Corporation, Cupertino, CA, 1998

1

R

Corporate Headquarters
18880 Homestead Road
Cupertino, CA 95014
Toll-free: 800-728-1212
Tel: 408-863-9900
Fax: 408-863-4120
E-mail: info@rational.com
Web: www.rational.com

Australia +61-2-9419-0100
Belgium +32-16-46-24-11
Brazil +55-11-829-7585
Canada 613-599-8581
France +33-1-30-12-09-50
Germany +49-89-628-38-0
India +91-80-553 8082/9864
Japan +81-3-5423-3611
Korea +82-2-556-9420
The Netherlands +31-23-569-4300
New Zealand +64-4-568-9591
South Africa +27-12-663-5677
Sweden +46-8-703-4530
Switzerland +41-1-445-36-00
Taiwan +886-2-720-1938
UK +44-1344-462-500

Lit No. TP505 Rev. 10/98. Subject to change without notice.

Formatted

Formatted