

EFFECTIVE
REQUIREMENTS
DEFINITION AND
MANAGEMENT

IMPROVES SYSTEMS AND COMMUNICATION

Contents

Executive summary	3
Requirements and rework	3
Key requirements issues	5
Requirements processes	6
Strategies for better requirements	7
Requirements elicitation	7
Requirements analysis	9
Requirements specification	9
Requirements validation	10
Requirements management	10
Assessing return on investment	11
Requirements definition and management: the Borland solution	13
Summary	15

Executive summary

By now, it is well known that shortcomings in requirements definition and management lead to excessive rework on software projects and products that fail to achieve full customer satisfaction. A closer look at their own software organizations can help managers at all levels identify pain points related to how requirements are elicited, analyzed, specified, validated and managed.

Software requirements engineering is a communication-intensive activity, at a minimum involving analysts, developers, business stakeholders and end users. Effective communication demands skilled requirements analysts, effective practices for requirements definition and management and tools to assist with these critical activities. The many human interfaces involved with requirements can lead to a variety of communication problems, including miscommunication between users and analysts, misunderstandings between analysts and developers, ineffective decision-making and inaccurate tracking of requirements status, all of which add time and cost to software projects.

This paper describes some of the key requirements issues that affect nearly every software and systems development project. The paper also outlines practical strategies and an effective solution to help address many common problem areas.

Requirements and rework

Software development involves perhaps 50 percent computing and 50 percent communication. Unfortunately, most teams are better at the computing part, and requirements are almost entirely about communication. There are many links in the requirements communication chain, as illustrated in Figures 1 and 2. A breakdown in any of these links leads to significant problems. For example, if an analyst misunderstands stakeholder input about requirements, if important requirements information does not surface or if an analyst and developer do not share the same understanding about requirements, the resulting product will not satisfy customers.

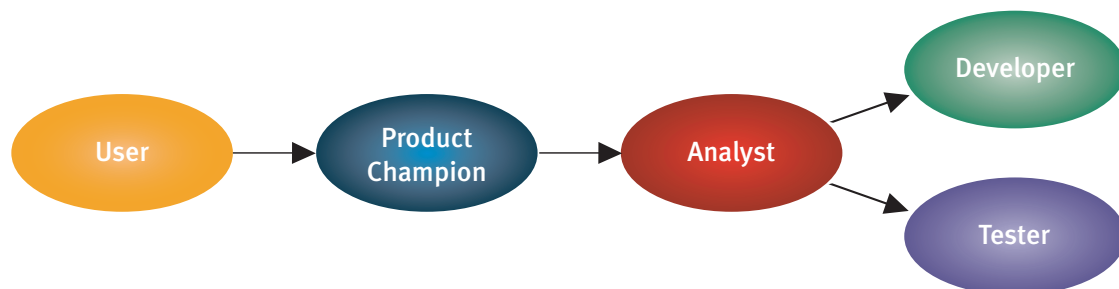


Figure 1. Typical requirements communication links in an information systems environment

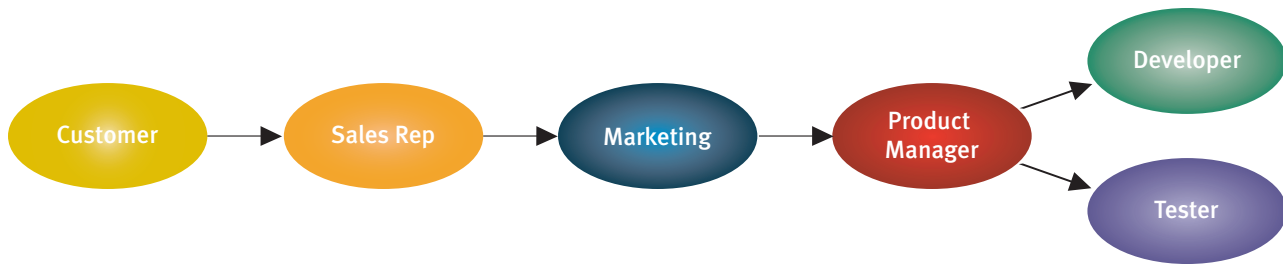


Figure 2. Typical requirements communication links in a commercial software environment

The inevitable outcome of requirements errors is time consuming and costly rework. Analysts report that rework can consume 30 to 40 percent of the total effort expended on a software project. Multiple studies have indicated that roughly 50 percent of the defects identified on software projects can be traced back to errors in the requirements. One analysis of the potential return on investment from better requirements suggests that requirements errors can consume between 70 and 85 percent of all project rework costs.¹ As Figure 3 illustrates, it can cost up to 110 times more to correct a requirements defect found in operation than it would if that same defect had been discovered during requirements definition.²

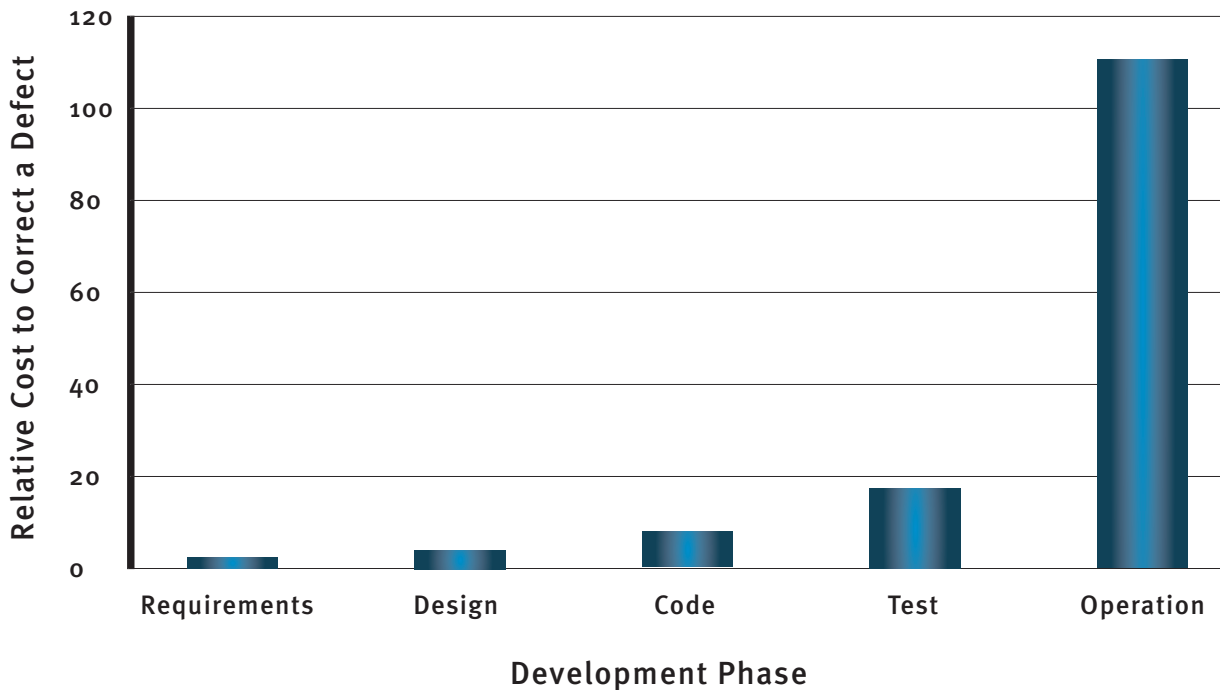


Figure 3. Relative cost to correct a requirement defect depending on when it is discovered

1 Leffingwell, Dean. 1997. "Calculating the Return on Investment from More Effective Requirements Management." *American Programmer* 10(4): 13–16.
 2 Grady, Robert B. 1999. "An Economic Release Decision Model: Insights into Software Project Management." In *Proceedings of the Applications of Software Measurement Conference*, 227–239. Orange Park, Fla.: Software Quality Engineering.

Good requirements practices may have always been challenging, but the Internet has made the problem worse. A 2001 survey “confirms that e-projects are time-compressed, intensive and mission-critical efforts with poorly defined requirements.”³

According to survey respondents, the top three risks threatening the success of the e-projects surveyed were:

- Unstable, constantly changing requirements (66 percent)
- Poor requirements specification (55 percent)
- Client behavior, such as approval delays, requirements changes and poor communication (42 percent)

Even though project teams often do not think they have the time to effectively elicit and capture requirements, they always seem to find the time, people and money to fix problems found in a delivered product. Therein lies the leverage for improving any organization’s requirements engineering approaches. If teams use better practices and better tools to facilitate requirements communication, they will introduce fewer defects attributable to requirements errors. As a result, they will reduce rework, thereby improving delivery time and quality with products that better meet business objectives.

Key requirements issues

Over the years, five universal truths have surfaced about software requirements engineering. The following statements are helpful for software development organizations to keep in mind as they consider how best to improve the requirements—and communication—for their projects.⁴

If teams don’t get requirements right, it doesn’t matter how well they execute the rest of the project.

The goal of every software development project is to build a product that provides value to customers. Effective requirements definition enables teams to determine the mix of product capabilities and characteristics that will best deliver this customer value. An understanding evolves over time as stakeholders provide feedback on the early work and refine their expectations and needs. Adequately exploring and crafting requirements into a set of product features and attributes helps to ensure customer needs are being met throughout the project lifecycle.

Requirements definition is a discovery and invention process, not just a collection process.

Teams often talk about “gathering requirements.” This phrase suggests that requirements are just lying around waiting to be picked like flowers or to be sucked out of users’ brains by an analyst. In reality, requirements definition is an exploratory activity, and requirements elicitation is a more accurate description than requirements gathering. Elicitation includes some discovery and some invention, as well as recording those bits of requirements information that various stakeholders present to an analyst. Elicitation demands iteration. Participants in a requirements elicitation discussion will not think of everything they will need up front, and their thinking will change as a project progresses. Teams that prepare to iterate most often elicit the most accurate requirements.

Customer involvement is the most critical contributor to software quality.

Various studies confirm that inadequate customer involvement is a leading cause of the failure of software projects.⁵ The development team will get the customer input it needs eventually—even if it is after a project ships. However, it is much cheaper—and much less painful—to get customer input earlier, rather than after product release. Customer involvement requires more than a workshop or two early in the project. It involves input from customers early and often in the

³ Rodrigues, Alexandre. 2001. “Project Goals, Business Performance, and Risk.” Cutter Consortium e-Project Management Advisory Service Executive Update 2(7).

⁴ Wiegers, Karl E. 2006. *More About Software Requirements: Thorny Issues and Practical Advice*. Redmond, Wash.: Microsoft Press.

⁵ The CHAOS Report, The Standish Group International, Inc., 1995.

requirements process. Ongoing engagement by suitably empowered and enthusiastic stakeholders is a critical success factor for software development.

Change happens; managing change is critical.

It is inevitable that requirements will change as business needs evolve, new users or markets are identified, business rules and government regulations are revised and operating environments change. The objective of a change control process is not to inhibit change. Rather, the objective is to manage change to ensure that the project incorporates the right changes for the right reasons. Teams that anticipate and accommodate changes minimize disruption and cost to the project and its stakeholders.

Teams are never going to have perfect requirements.

Requirements are never finished or complete. There is no way to know for certain that teams have not overlooked some requirement, and there will always be some requirements that are not in the specification. It is also folly to think teams can freeze the requirements and allow no changes after some initial elicitation activities. Rather than declaring requirements “done” at some point, effective teams define a baseline then follow a sensible change control process to modify requirements once a baseline is established.

Requirements processes

Good requirements practices can accelerate software development. The process of defining business requirements aligns the stakeholders with shared vision, goals and expectations. Substantial user involvement in establishing and managing changes to agreed upon requirements increases the accuracy of requirements, ensuring that the functionality built will enable users to perform essential business tasks.

Software requirements engineering encompasses the two major sub domains of requirements definition and requirements management:

Requirements Definition is the collaborative process of collecting, documenting and validating a set of requirements that constitute an agreement among key project stakeholders. Requirements definition is further subdivided into the critical process areas of elicitation, analysis, specification and validation processes.

From a pragmatic perspective, requirements definition strives for requirements that are good enough to allow the team to proceed with design, construction and testing at an acceptable level of risk. As discussed, the risk is the threat of having to do expensive and unnecessary rework.

Requirements Management involves working with a defined set of product requirements throughout the product’s development process and its operational life. It also includes managing changes to that set of requirements throughout the project lifecycle.

In practice, requirements management includes selecting changes to be incorporated within a particular release and ensuring effective implementation of changes with no adverse impact on schedule, scope or quality.

An effective requirements definition and management solution creates accurate and complete system requirements, while helping organizations improve communications in an effort to better align IT with business needs and objectives. It includes a set of industry best practices for each category, as well as tools to enable and accelerate requirements activities.

Strategies for better requirements

A variety of practices can help software teams bridge communication gaps and do a better job of understanding, documenting and communicating customer needs. Figure 4 illustrates, and process areas below describe, several best practices in the categories of requirements elicitation, analysis, specification, validation and management.⁶



Figure 4. Best practices for each requirements category

Requirements elicitation

Define the product vision and project scope.

The product vision is the long-term strategic concept underlying the ultimate purpose and form of the new system. The vision could also describe the product's place among its competition in its market or operating environment. The project scope is the portion of the product vision that the current project will address. The scope draws the boundary between what is in and what is out for that project. Without a clearly defined project scope, the project is issuing an open invitation to scope creep. Prior to eliciting requirements, teams should have a clear understanding of both the product vision and the project scope.

⁶ Wiegers, Karl E. 2003. Software Requirements, Second Edition. Redmond, Wash.: Microsoft Press.

Identify stakeholders, customers and users.

To software development groups, customers are a subset of stakeholders and users are a subset of customers. Teams can further subdivide user communities into multiple user classes that have largely distinct needs. Unrepresented user classes typically will be disappointed with the project outcome. It is essential to gain commitment from key stakeholders for their participation throughout the requirements definition. Customer engagement is necessary during requirements management, as well. The customer perspective is required when making change decisions, assessing the impact of proposed changes and adjusting requirement priorities. Every software project should identify its key requirements decision makers and its decision-making process early on to ensure that the right people can make important and timely decisions.

Select product champions.

Teams must determine who will serve as the literal voice of the customer for each user class. These representatives are called product champions. Ideally, product champions are actual users who represent their peers in a particular user class. Sometimes, though, surrogate product champions are needed. The product manager often performs this role in a commercial software development organization. When engaging product champions, it is important to agree on the level of involvement. It is one thing to participate in a workshop or two, but sustained engagement with frequent contact points between product champions, analysts and developers adds far more value. Teams should take care to not only choose a representative, but to choose the right representative. Product champions must understand the business requirements expressed in the product vision and project scope which define the boundaries for the work the product champions do. The best product champions are collaborative partners in the requirements process and communicate with other members of the team to request input, solicit feedback and resolve conflicts.

Choose elicitation techniques.

The methods an analyst uses for requirements elicitation depends on the extent of stakeholder involvement and how much access the analyst has to the stakeholders. Workshops to explore scenarios and use cases work well when user representatives are locally available. Questionnaires and surveys might be necessary if there are many, geographically distributed stakeholders. Individual interviews with subject matter experts are also useful for capturing information, as are analysis models and building interactive prototypes. These various elicitation techniques are not mutually exclusive. When the analyst can use multiple communication channels with the sources of requirements information, the chance of getting high quality and complete information increases. Therefore, teams should be trained and proficient in a variety of elicitation techniques.

Explore user scenarios.

Requirements analysts have learned that elicitation discussions that focus on users and how they use a product or system generally yield the best results. Users find it more natural to describe their business tasks and usage goals than to define all of the functionality they expect to see in a product. Use cases, scenarios and user stories are related terms that are used frequently to describe a system's user requirements. Use cases do not replace the need to define the detailed functional requirements that developers will implement. However, teams should explore use cases or user scenarios to help ensure that the requirements their developers implement will allow users to accomplish their goals.

Requirements analysis

Create analysis models.

The natural language requirements found in most specifications are full of ambiguities, redundancies and gaps. In most cases, it is desirable to represent requirements in multiple ways, giving readers a richer, more holistic understanding of them. Analysis models present requirements information visually, in graphical diagrams. They allow reviewers to immediately spot missing requirements by examining diagrams, rather than uncovering missing requirements by reading a dense textual specification. Teams may have better results using models that communicate at a higher level of abstraction, so readers can get the big picture without getting mired in all of the details.

Build and evaluate prototypes.

A prototype is a partial, preliminary or possible solution to the requirements. Prototypes provide opportunities for product champions to interact with a simulation or portion of the final system. Prototypes are more tangible than written requirements specifications; they are a way to bring use cases to life. When creating a prototype, the development team is taking a tentative step into the solution space, which is a valuable way to validate and refine requirements, but it is not a substitute for documenting detailed functional requirements. A prototype or set of screen designs does not show the business logic happening behind the scenes. It does not illustrate the complete behavior of the product when the user takes certain actions under certain conditions. A prototype also does not indicate how all exceptions and error conditions are going to be handled, although this information is critical if teams want to build robust software. When used with the right audiences, prototypes are an effective way to help teams analyze existing requirements for a new system.

Prioritize requirements.

Every software development organization has limited resources and time. Therefore, every team needs to determine which of its allocated requirements are most important and which are most urgent. Prioritizing requirements allows teams to implement the right sets of user functionality in the right sequence. Prioritization should be a collaborative process that involves both a customer and a technical perspective to balance customer value against cost and technical risk.

Requirements specification

Look for ambiguities.

Requirements written in natural language are fraught with ambiguity. Negative requirements, use of vague and subjective terms, complex logic, omissions, synonyms and adverbs lead to multiple interpretations by different readers. It is cheaper to correct ambiguities in the requirements than to deal with disappointed customers. As a result, teams should establish an agreed upon lexicon prior to writing requirements.

Store requirements in a database.

By storing requirements in a commercial requirements management tool, such as Borland® Caliber® Analyst, teams can overcome many of the limitations imposed by textual documents. Requirements management tools make it easy to store additional information—attributes—about different classes of requirements information. They facilitate tracking requirements status. They provide a mechanism for retaining requirements that have been proposed, rejected or approved, and later deleted from a baseline. The tools also make it easier to work with groups of requirements that are intended for multiple future releases. In addition, organizations that keep requirements in a shared online database can better facilitate communication and collaboration among distributed teams.

Trace requirements into design, code and tests.

It is valuable to be able to link each software functional requirement back to its origin, possibly a use case or business rule. Teams should embrace traceability information that connects functional requirements to associated design elements, code segments and tests to accelerate debugging and software maintenance. Requirements management tools are a great aid to managing traceability data.

Requirements validation

Review the requirements.

The highest leverage quality practice available to software teams is formal peer review of requirements documents. Peer review provides an indication of how well others understand the requirements. The requirements analyst should document requirements so as to ensure that they communicate clearly, effectively and efficiently to the various stakeholders. The analyst should create complementary views of the requirements selected—from textual requirements, analysis models, scenarios, prototypes, tests and other representations—all of which can be reviewed concurrently. All project stakeholders should review the requirements to ensure accuracy, completeness, and the optimal level of detail to deliver software that truly meets the business needs.

Create test cases from requirements.

Teams should begin “testing” as soon as they have some requirements in hand. Deriving test cases from use cases and scenarios is a valuable way to find problems in the use cases themselves, in functional requirements derived from the use cases and in analysis models created from the requirements.

Requirements management

Manage requirements versions.

As requirements evolve during the course of a project, it is important to track the different versions of requirements specification documents and even individual requirements. Teams that use version tracking help to ensure that all team members are working from the most current requirements baseline.

Adopt a change control process.

Once requirements have been baselined, proposed modifications in them should follow an established change control process. This process provides consistency in the way requirement changes are proposed, evaluated, approved or rejected, communicated to stakeholders and implemented in affected work products. Teams should have formal written change control processes in place before eliciting requirements.

Perform requirements change impact analysis.

To help the change control board (CCB) make appropriate business decisions about which proposed changes to approve, developers should assess the potential impact of each proposed change before committing to implement it.

Store requirements attributes.

Requirements attributes provide a richer understanding of each individual requirement. Potential attributes to track include priority, status, author, origin, rationale, validation method, risk and version number. Teams should store attributes with the requirements to ensure the detail necessary to communicate and prioritize requirements.

Track the status of each requirement.

Tracking project status is facilitated if the team can report on the status of individual functional requirements in the baseline. There are a number of possible requirements checkpoints, including proposed, approved, implemented, verified, rejected and deleted. Teams that track the status of each requirement can easily assess the health of the project and avoid unnecessary status meetings.

Assessing return on investment

Software development managers want to know what return on investment (ROI) they can expect from the money they spend on training, process improvement and tools for requirements definition and management. Although there are too many variables to predict a specific ROI for a given organization, the following are some factors to consider when estimating the ROI organizations can expect from better requirements.

In general, to determine the ROI from any new initiative, compare what was invested in the activity with the experienced benefits—such as reduced costs, accelerated schedules or increased sales. For example, track what would be spent on the following process improvement activities to determine your investment:

Assess current practices.

All process improvement should begin with an appraisal to learn how teams are dealing with requirements issues today and whether current approaches are successful or struggling. Use the Borland Requirements and Definition Management self assessment as an opportunity to quickly assess your organization's requirements definition and management maturity and see an action plan for more effective and sustainable results.

Develop new processes and templates.

Once team members have identified specific requirements engineering practices for improvement, identify practices that will work better. This may involve writing new processes, modifying current processes and selecting or adjusting the templates for key requirements deliverables.

Train the team.

All analysts and others who deal with project requirements should receive basic training in requirements concepts and practices. Team members should also receive training in the effective use of specific organizational processes and templates.

Acquire books and other resources.

Analysts will benefit from having reference books and articles on hand to refresh their knowledge and uncover new ideas for handling requirements challenges they encounter.

Employ external consultants.

Many software organizations require assistance from experienced consultants who have worked with a variety of companies. Consultants can help team members solve problems much faster than they might on their own.

Invest in requirements definition and management tools.

As requirements engineering activities become more sophisticated, teams should store requirements in a database, rather than in traditional word processing documents. Requirements management tools such as Borland CaliberRM® have been commercially available for several years. However, requirements management tools do not help teams scope projects, identify

and talk to the right users or write good requirements. More recently a new class of requirements tools has begun to appear, such as Borland® Caliber® Analyst, that support not only requirements management, but also requirements definition. Some of these tools assist with representing requirements in the form of use cases or scenarios, prototypes, graphical analysis models or simulations. Other tools analyze requirements for quality by scanning for vague words or through linguistic analysis to identify possible omissions in ambiguities.

Define and manage project requirements.

The best investment is the investment in time that team members spend defining and managing the requirements for the products they are creating. Multiple studies have confirmed that spending more time on requirements definition actually accelerates the project by reducing late-stage rework.

Estimate your return.

To estimate the payback that better requirements can bring to your organization, consider these questions:

- What fraction of your own development effort is expended on rework (due to miscommunication)?
- How much does a typical customer-reported defect cost your organization? A defect found in system testing?
- What fraction of user-reported defects and what fraction of defects discovered during system testing originate in requirements errors?
- How much of your organization's maintenance costs—such as defect correction and unplanned enhancements—can you attribute to missing requirements or other requirement defects?
- How much do you think your organization could shorten its delivery schedules if your project teams could reduce requirement defects by 50 percent?

Besides the obvious cost benefits, improving requirements approaches leads to other valuable—but less tangible—outcomes. Experiencing fewer miscommunications on a software project reduces the overall level of chaos. Less chaos lowers unpaid overtime, increases team morale, boosts employee retention and improves the team's chances of delivering on time. Best of all, these benefits lead to higher customer satisfaction.

Requirements definition and management: the Borland solution

Borland helps global organizations maximize the business value of software and accelerate the path to Software Delivery Optimization (SDO) by transforming software development into a managed business process to increase control, predictability, visibility and efficiency over the entire software delivery process. One of the core processes critical to achieving SDO is effective requirements definition and management.

The Borland Requirements Definition and Management Solution is the only scalable, integrated enterprise requirements definition and management solution to appropriately align people skills, process improvements and award-winning technology to deliver essential capabilities that significantly increase the probability that a project is delivered right the first time. It is the most comprehensive enterprise solution available in all five of the key areas of requirements definition and management, including:

Elicitation: reduce rework by capturing better information.

The process of capturing requirements in context, requirements elicitation includes identifying key business and technical stakeholders, getting commitment to stakeholder involvement, selecting appropriate elicitation techniques and capturing requirements and scenarios in a simple to understand form. To reduce rework, Borland helps organizations mature their existing requirements elicitation process by helping to define responsibilities and stakeholders, identify appropriate elicitation techniques and train team members to use the right techniques. Borland also helps teams leverage Borland Caliber Analyst technology to capture user scenarios in a simple, visual form that users readily understand. With the Borland Solution, organizations enhance communication and collaboration among distributed teams throughout the project lifecycle. Teams also improve alignment between business expectations and project deliverables, which increases end user satisfaction and reduces rework from incorrect or incomplete requirements.

Analysis: reduce time to market with more effective collaboration.

Requirements analysis involves verifying, estimating and prioritizing newly captured requirements for remaining application lifecycle steps. To reduce time to market, Borland helps organizations mature their existing requirements analysis process by implementing an effective approach to evaluate and prioritize requirements for specification, design, construction and testing. The process is enhanced through the deployment of tools and techniques that increase efficiency and accuracy. With the Borland Solution, organizations gain better estimation and thus, improved predictability for software delivery.

Specification: improve quality through more effective communications.

Requirements specification includes adding detail to requirements incrementally to the optimal level for validation, design, coding, testing and documentation. To improve quality, Borland helps organizations mature and automate their existing requirements specification process by defining a standard hierarchy of requirement types and developing standard templates to ensure completeness. Borland also helps teams identify various specification techniques (e.g. use case and business process models, prototypes and traditional requirements specifications), and apply technology such as Borland Caliber Analyst so that requirements are captured in a meaningful, easy-to-understand way. Borland tools provide automated traceability across the lifecycle so organizations are better able to predict the impact of change and proactively communicate those changes to all team members affected. With the Borland Solution, organizations reduce software defects because development teams have a better understanding of requirements.

Validation: improve accuracy and completeness to close development lifecycle gaps.

Requirements validation involves verifying the specification is complete and clear enough for the development team to understand exactly what it needs to build. It also includes validating with key stakeholders that specified requirements are consistent with the original need and intent of the business. To improve accuracy, Borland helps organizations mature their existing process by automating validation and verification to drive adoption and enforcement and improving consistency and quality through storyboard execution delivered in Borland Caliber Analyst technology. Borland also helps organizations develop team member skills using tools and education to improve quality and define and implement processes for validating requirements with stakeholders that ensures business and IT alignment. With the Borland Solution, organizations reduce software defects, increase satisfaction and alignment with business stakeholders and enhance business value.

Management: reduce costs through improved change management.

The process of gathering and managing change requests during the application lifecycle, requirements management also includes selecting changes to be incorporated within a particular release and ensuring effective implementation of changes with no adverse impact on schedule, scope or quality. To reduce costs, Borland helps organizations mature their existing requirements management process by establishing processes for defining and maintaining requirements baselines and defining a standard process for requesting changes. Borland also helps teams establish a systematic approach to evaluate and approve change requests so that scope changes and affected commitments are managed. With the Borland Solution, organizations improve ongoing change management, maximizing business impact, while minimizing schedule and scope impact. They also increase business stakeholder satisfaction.

The Borland approach to transforming requirements definition & management.

The Borland approach to requirements definition and management is unique in that it takes into account an organization's process maturity by leveraging industry best practices to evaluate current performance and identify specific areas for improvement. Borland does this by using the proven Borland Accelerate Framework, a disciplined, four-phase improvement approach that incorporates the five principles of a successful transformation and effectively aligns people, process and technology to maximize results.

Using the Framework and best practices process assets as a guide, Borland supports organizations at various stages in their requirements definition and management transformation.

- Phase I: Borland experts help define goals using executive workshops and other activities that result in clearly defined objectives.
- Phase II: Borland experts help organizations architect their approach and prioritize the key solution components, including critical process modules, required technology configurations and skill development plans.
- Phase III: Borland helps organizations develop, pilot and deploy the enterprise-wide requirements definition and management solution defined in Phase II, including implementing the process, installing and configuring the technology and training users.
- Phase IV: Borland helps organizations validate results against the goals defined in Phase I and identify gaps that need to be addressed in the next improvement cycle.

With the Borland Solution, typical organizations will move from an ad hoc requirements process— consisting of manual processes, word processing documents and ineffective requirements management systems—to a requirements lifecycle focus with a fully integrated system for managing each core process area described. As a result, organizations can more effectively define requirements and better manage the constant barrage of requirements changes that occur within the application lifecycle.

Summary

The Borland Solution delivers everything organizations need to successfully implement the five critical requirements definition and management processes into an organization for maximum impact, quickly and with a minimum of risk. The solution can be tailored to reflect an organization's maturity and goals, which is crucial to the ultimate success of any initiative.

The Borland Requirements Definition and Management Solution helps to ensure low-risk, rapid success at the lowest total cost by including “right sized” processes, customizable technology and tailored training packages to enable users and empower customers to proceed with confidence.