

Introduction to
Requirements Gathering

Prepared for:
St. Edwards University
Analysis, Modeling and Design MCIS6310 – Dr. David Franke

6 June 2006 Copyright ©2005-2006 Tyner Blain LLC 1

This presentation provides an overview of the initial parts of the requirements engineering process. It touches on requirements gathering, a framework for managing requirements, and some of the key requirements management areas. Several slides in the presentation will reference specific posts at Tyner Blain – <http://tynerblain.com/blog/> . Those posts include additional background information, comments from other professionals in the field, and links to other topics of interest.

Outline

1. Overview of Requirements Management
 - Who is involved?
 - How do we approach the space?
 - What outputs do we create?

2. Structuring Requirements Information
 - How do we categorize requirements?
 - What is the hierarchy of requirements?

3. Managing the Requirements
 - How do we gather, maintain and use requirements?

We'll cover a high level view of the space in this presentation. We start with identifying the key players and the artifacts we create as part of working together. We will look in a little more detail into the structure of that information, and then touch on some of the elements of maintaining the requirements information with the remaining time. Since this is a 60-90 minute presentation, we won't go into a lot of depth on any one topic.

Outline

1. Overview of Requirements Management
 - Who is involved?
 - How do we approach the space?
 - What outputs do we create?

2. Structuring Requirements Information
 - How do we categorize requirements?
 - What is the hierarchy of requirements?

3. Managing the Requirements
 - How do we gather, maintain and use requirements?

Who Is Involved?

- Primarily two groups are involved in designing, developing and deploying a system or solution
 - Stakeholders of the system
 - Sponsor / Customer
 - User
 - Creators of the system
 - Strategy (What do we build?)
 - Execution (How do we build it?)
 - Subject matter experts (SME) / BA
 - Product and program mgr / BA
 - Developers
 - Quality engineers

Stakeholders

- Sponsor / Customer
 - Funds the project
 - Benefits from the project
 - Great contact for that “vision thing”
- User
 - Uses the solution after deployment
 - Great source of validation
 - Great source of post-deployment feedback

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

5

Pricing Optimization – sponsor is VP of Sales – who benefits from higher gross revenue and profits

Process Automation – sponsor is a manager with P&L responsibility for an area of the business

Inventory Mgmt Solution – sponsor is Plant Mgr, responsible for ROA for his plant – reducing WIP/Materials, or lead time for delivery

Simply put, users are people who interact with the system

Related links

Defining and designing for users - <http://tynerblain.com/blog/2006/04/17/persona-grata/>

Creating user personas - <http://tynerblain.com/blog/2006/03/22/how-to-create-personas-for-goal-driven-development/>

Creators - Strategy

- SME (subject matter expert) / BA (domain)
 - Domain expertise, problem definition
- Product Manager
 - Prioritization and definition of market requirements
 - Defines vision to achieve benefits
 - Outbound (marketing, sales support)
 - Inbound (product definition, release planning)
- Requirements / Program Manager / BA (software)
 - Defines what will be delivered and when
 - Validates that requirements will achieve vision/goals
 - Coordinates activities of all parties
 - Manages documentation and communication

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

6

SME

A participant in 'the market' – someone who understands the problems, or ideally *is* a stakeholder. Lack of support from SME/stakeholders seriously jeopardizes a project. A great example is Maine's recent failures at meeting HIPAA compliance <http://tynerblain.com/blog/2006/04/21/maine-mangles-medicaid-charges-cio/>

Product Manager

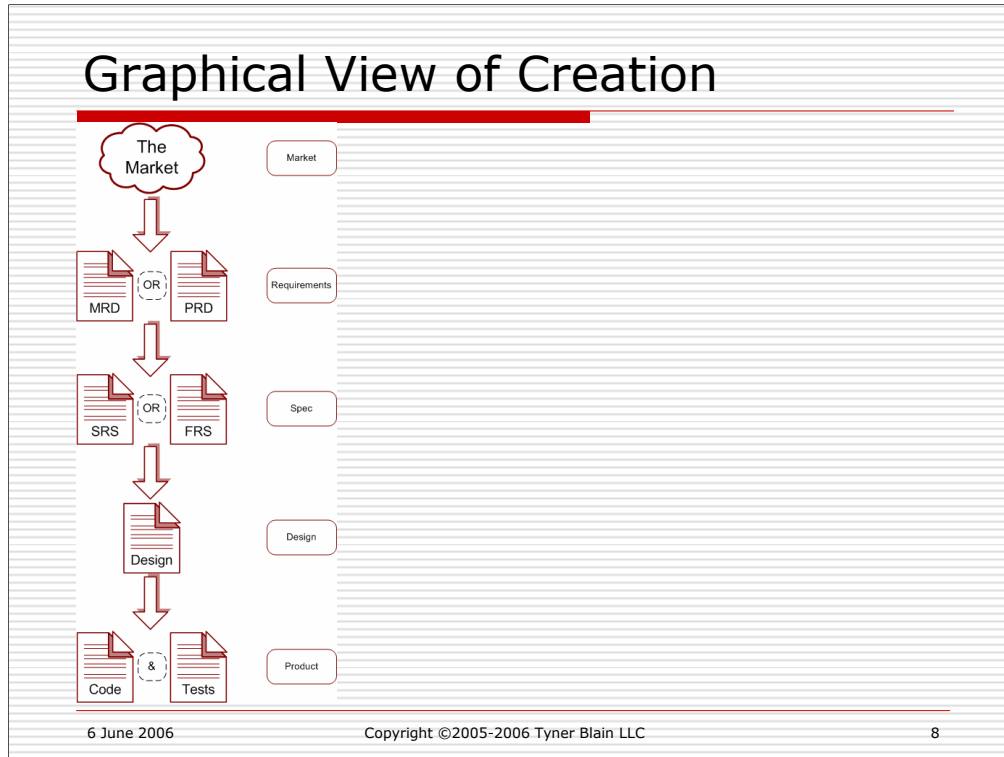
Understands what software can do, understands the domain of the SME. Brings those two worlds together. In terms of software development, the person who can marry problems and solutions. Innovative thinking and prioritization – “This should be solved with software” “This is not worth solving”. For more details, see <http://tynerblain.com/blog/2006/01/24/from-mrd-to-prd-the-key-to-defining-a-spec/>

Requirements Manager / Program Manager / Software Business Analyst / “Program Designer” in Pragmatic Marketing's vocabulary

This person creates an *actionable* specification (SRS or FRS) based on the targeted market requirements (in an MRD or PRD). See <http://tynerblain.com/blog/2006/05/11/requirements-documents-one-mans-trash/> for more details. Depending on the organization, this person will cross the boundary between strategy and execution. The primary responsibilities are to make sure that we have a spec that will satisfy the market requirements (validation) and coordinate the execution of delivery. Not a project management task, but almost. Varies with organizations.

Creators - Execution

- Project Manager / Development Manager
 - Manages the execution of deliverables
- Designer & Architect
 - Interface Design and Program Design
- Developer
 - Responsible for delivering components of solution
- Quality Engineer
 - Responsible for assuring quality of solution



The flow starts with an understanding of the problems in the market.

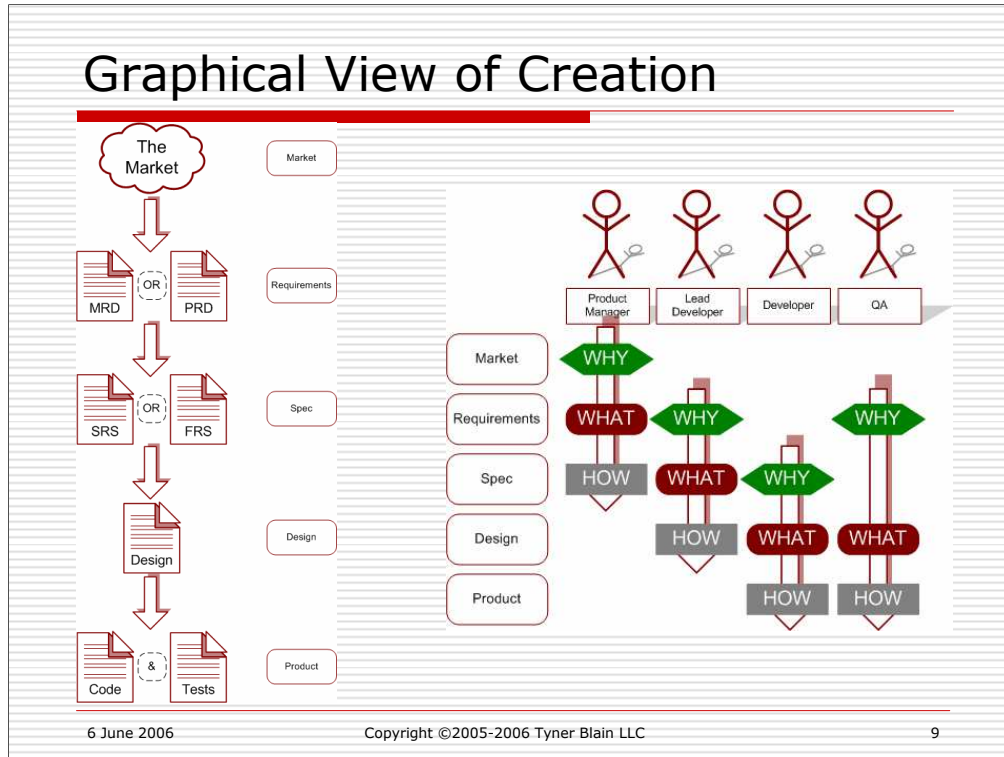
Some of those problems are valuable and practical candidates for software solutions.

Detailed software approaches can make this possible.

A design is developed to present a roadmap for how this will be implemented.

Implementation involves both coding and testing. They are (or should be) inseparable.

Diagram from <http://tynerblain.com/blog/2006/05/11/requirements-documents-one-mans-trash/>



A simplifying view of how people in each role think about the problem and the solution.

Why – Why am I (are we) doing this?

What – What are we trying to do (to satisfy the *why*)

How – How will we do it?

Diagrams from <http://tynerblain.com/blog/2006/05/11/requirements-documents-one-mans-trash/>

Example Requirement

- Market
 - Music sales are dropping with 18-35 yr olds.
- MRD
 - Online music sales are growing at double the rate of in-store sales decline. Create a music catalog with sample clips for ordering music online.
- PRD
 - We need to sell songs and albums online for downloading and media purchase.
 - We need to enable customers to preview songs before purchasing.
 - We need customers to be able to find available music.
- SRS
 - The system shall present all available media in a searchable catalog.
 - Sample downloads will be available from product pages when samples are available.
 - Downloadable products will be available from product pages when available.
 - ...

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

10

At the market level, we articulate a valuable problem or opportunity.

In the MRD we triage market problems for software solutions, and present a vision for the product.

In the PRD we provide a synthesized solution approach.

In the SRS we provide actionable, measurable software criteria.

We validate the MRD to assure that it addresses the market need.

We validate the PRD to assure that it satisfies the MRD requirements.

We validate the SRS to assure that it elucidates the PRD requirements.

We verify the SRS as being practical and assure that it can be implemented by our team.

Crossing the Line

- Difference between requirements and design – simple to define, hard to live by
- If the language talks about implementation, then it's probably design
 - "Users select their language from a dropdown list"
 - "Users can specify their default language"
 - "Search all records created in the database between the specified dates"
 - "Search all records with a creation date between the specified dates"
- Reality gets in the way sometimes
 - "Populate the list of customers from <external system>"
 - "Populate the list of customers from TBL1.CUST_ID not TBL2.CUST_ID in <external system>"

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

11

Reality can be hard because sometimes, the person managing the requirements is the right person to gather non-requirements information

Identifying which fields in an external system to use is too low level to be called a requirement – it's an implementation detail. However, on one project I worked on, the implementation team was

- 1) relatively green
- 2) geographically and temporally displaced
- 3) did not have relationships with the SMEs

Further, the requirements repository was the only/best place to store this information. The solution we used was to keep the table-details in the requirement, in a field called "design ideas".

Requirements Frameworks (1)

- Karl Wiegers methodology
 - Framework for describing requirements hierarchically (Goal>Use Case>Functional requirement>Test)
- User centered design (UCD)
 - Software features and functions are driven (formally) from use cases, scenarios, or scripts.
 - Relation to value (profits) is informal
- Interaction Design
 - Identifies primary users of the system
 - Designs solution specifically for them
 - Applies psychology, human factors and others

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

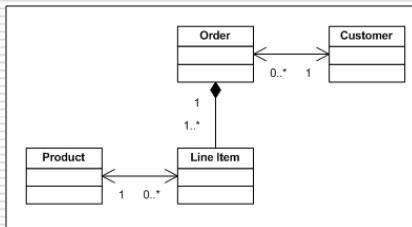
12

UCD differs from Wiegers mainly in that the goals behind the use cases are not formally articulated

Interaction Design takes UCD and applies psychology, human factors, and other design sciences. ID is a subset of UCD, which is a general philosophy

Requirements Frameworks (2)

- OOA/OOD (Object oriented analysis/design)
 - Technique using UML to present analysis findings
 - Build two object models –
 - one that describes the system (e.g. Customers have Orders)
 - one that describes the solution (class Customer has a collection of Order objects)



6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

13

Note OOA/OOD *requires* supporting documents – while information density is much higher in diagrams, supporting prose is still needed

<http://tynerblain.com/blog/2005/12/09/a-picture-is-worth-a-thousand-requirements/>

Artifacts

- MRD / PRD for strategy
- Use cases bridge PRD to FRS
- SRS / FRS for tactics
- OOA/OOD diagrams
- Stakeholder documents
 - Release schedule / Product roadmap
 - ROI analysis / SOW
 - Status updates

- Screenshots? Technically no.

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

14

Screenshots are fundamentally a design artifact – they represent a manifestation or an implementation designed to achieve the required functionality.

The water gets muddied sometimes, when the customer micromanages the requirements process, things like “I require a search field/button in the top left corner of every page” masquerade as requirements. They are really constraints. Same thing for UI standards (match Look and Feel of corporate intranet).

Another challenge – screenshots/mockups/prototypes are fantastic for conveying the ideas behind a solution to a user or stakeholder for feedback – as well as elicitation of new ideas for future releases. “Oh, yeah – I see it now”. hear that all the time.

So – screenshots are design artifacts, but we use them in the requirements gathering process. Yes.

Outline

1. Overview of Requirements Management

- Who is involved?
- How do we approach the space?
- What outputs do we create?

2. Structuring Requirements Information

- How do we categorize requirements?
- What is the hierarchy of requirements?

3. Managing the Requirements

- How do we gather, maintain and use requirements?

Structured Requirements

- Goals
 - What is the ROI? The objective?
- Use Cases
 - How will users interact with the system?
- Functional requirements
 - What, precisely must the system do?
- Non-functional requirements
 - Characterize *how* software performs
 - Constraints (must use J2EE)
 - Limits design flexibility in exchange for conformance

6 June 2006

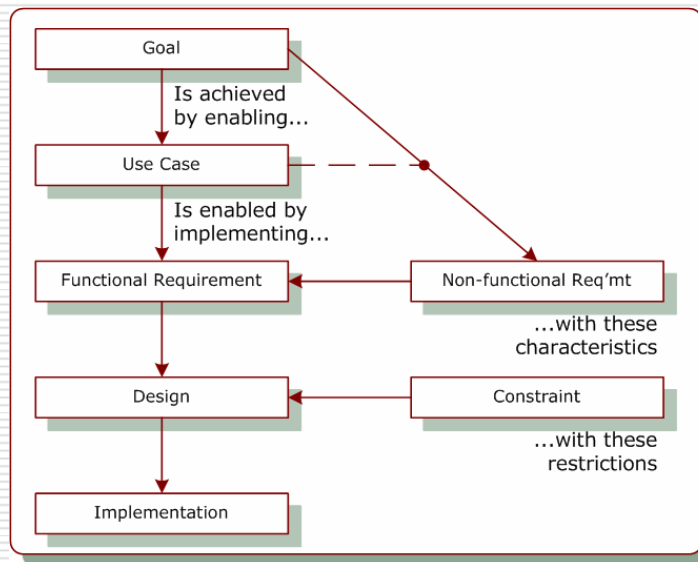
Copyright ©2005-2006 Tyner Blain LLC

16

See <http://tynerblain.com/blog/2006/05/23/non-functional-requirements-era/>

For an alternate view, see <http://tynerblain.com/blog/2006/03/23/interaction-design-and-structured-requirements/> which presents a mashup of Weigers' and Cooper's approaches.

Types of Requirements



6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

17

From <http://tynerblain.com/blog/2006/05/23/non-functional-requirements-era/>

Goals are achieved by enabling use cases.

Goals drive non-functional requirements.

Use cases are enabled by [implementing functional requirements](#).

[Use cases](#) influence non-functional requirements.

Non-Functional Requirements define the characteristics of functional requirements.

Functional requirements drive design decisions

Design choices are restricted by constraints

Design choices guide implementation

Implementation is product

Goals

- Stakeholders fund projects to achieve a specific objective
- Increased Profits
 - Increased Profits from cost reductions
 - Increased Profits from increased revenue
 - Increased Profits from improved branding
 - Increased Profits from increased user adoption
- There's a theme

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

18

Reduce costs

Make current process more efficient (usually automation and workflow enhancements to existing business processes)

Increase revenue

Define new business processes to enable new modes of operation

-Base pricing on data-mining results to attempt to profit-maximize

-Enable cross-selling or up-selling of products in a retail web-site

Improve branding

-Create a community of users or a sense of identity with a site or products

-Enable email campaigns, targeted discounts based on behavior, demographics

Increase user adoption

-Make a website sticky – provide information, “rewards” for users, create feedback mechanisms for users

-Address user needs, provide value, make roadmap visible (show future enhancements, schedule)

-Ease of use, task-focus

Use cases – Informal

- Narrative description of what the user can do with the system (from the user's perspective) and some detail about how it is accomplished.
 - After logging in, the user can edit his personal profile information. The user will be able to view all of his changes prior to committing them or canceling the changes. The system will provide feedback that the changes were saved or cancelled, as appropriate.
 - The user can create a hot-sheet showing the products with the most sales for a specified period. The user can specify the period of interest, the product areas she is interested in, and the minimum dollar amount to use to limit the length of the report. The system will present the results of all products meeting the specified criteria.

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

19

Pros:

Easy to create – quick development, iteration, and collaboration. This enables a rapid approach to documenting use cases, and minimizes the cost of developing the use cases.

When done correctly, yields the most bang for the buck of any use case approach.

Cons:

Challenging to be rigorous – the short format makes it difficult to capture all the relevant information (and difficult to avoid capturing irrelevant information).

Lack of consistent structure – can be transition from use case to use case, since the format is free-form

Capturing the right level of content for *your* team can be tricky.

<http://tynerblain.com/blog/2005/12/21/use-case-series-informal-use-case/>

Use cases - Formal

- Detailed step-by-step descriptions of how objectives are achieved.
 - Preconditions – Required state to initiate
 - Normal course – The most common sequence
 - Alternate courses – More than one way to do it
 - Exception courses – What happens when something goes wrong or is unexpected (error handling, for example)
 - Post-conditions – “Contract” for results of completion

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

20

Preconditions – don't include trivial and redundant info (user must be logged in, computer must be turned on, etc)

Normal course – describe what “generally” happens

Alternate courses – user manually creates profile instead of using wizard

Exception courses – handling errors – “what happens when desired data is insufficient to complete calculation”

Pros:

Detailed information about use cases, making it easy to estimate the cost of implementation.

Thorough coverage of the use cases is influenced by the use of a template.

Easy for readers to absorb. The structure of the document makes scanning easy, and also helps targeted *lookups* when a reader needs a specific piece of information.

Consistency with other use cases is much easier to assure when using a template.

Cons:

Expensive to maintain. Mapping a “use case” to the template requires some effort. Since formal use cases contain more (explicit) information than other use cases, there is more content to document, validate and modify. More content equals more cost (of maintenance).

<http://tynerblain.com/blog/2005/12/20/use-case-series-formal-use-case/>

Functional requirements

- Defining precisely *what* is to be built
- Primary consumers are developers and testers.
- Precise about behavior, ambiguous about implementation choices

<http://tynerblain.com/blog/2006/02/10/writing-functional-requirements-to-support-use-cases/>

Non-functional requirements

- Defining precisely *how* the application must perform
 - Must use server's clock when time-stamping transactions
 - Must support internationalization
 - Batch processing must complete in <8 hrs
 - Search must return some results within 3 seconds, all results within 20 seconds

<http://tynerblain.com/blog/2006/05/05/non-functional-requirements-list/>

Constraints

- Specify the environment or implementation details that restrict *design* choices
 - Must use an existing Oracle database
 - Must run on Linux machines
 - Must be XHTML only
 - Must be Managed code (.NET)
 - Must meet section 508 Accessibility standards

Constraints impact scoping and estimation.

Constraints also bound the possible solution space

<http://www.section508.gov/> -> required for software sold to government agencies

Glossary of Terms

- Provides a *Rosetta Stone* of domain-specific and esoteric knowledge.
 - Definitions of financial terms
 - (e.g. Net30, EBITDA)
 - Definitions of business structures
 - (e.g. divisions, product lines, fiscal year)
 - Definitions of engineering terms
 - (e.g. Erlang calculations, QoS, load balancing)
- Enables efficient, unambiguous communication from stakeholders and SMEs to other team members

Decomposition

- Solely an organizational technique for combining requirements
- X aggregates X1, X2, X3
 - User Account Management (X)
 - Create new account (X1)
 - Modify existing account (X2)
 - Delete existing account (X3)
- Priority of sub-requirements may vary.
- Delivery schedule may vary.

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

25

Allows for clarity of communication – scoping, scheduling, testing, delivery of constituent components

1. Proposal support

1.1 Proposal states. Proposals will be shown to users as having one of the following states, as defined in the data dictionary: draft, submitted, approved, rejected, completed.

1.2 Proposal creation. Salespeople can create and save proposals which are automatically created in the draft state.

1.3 Proposal workflow. Managers can create, submit, approve or reject a proposal. Salespeople can create and submit proposals.

1.4 Proposal viewing. The system shall present an interface that allows users to view all proposals in the system, filtering and sorting by proposal state and creation date.

1.5 Proposal modification. Completed proposals can not be modified. All proposals can be edited, and their state will be changed to draft after editing. Only draft proposals can be submitted. Only submitted proposals can be approved or rejected.

<http://tynerblain.com/blog/2005/12/07/composition-in-requirements/>

Outline

1. Overview of Requirements Management
 - Who is involved?
 - How do we approach the space?
 - What outputs do we create?

2. Structuring Requirements Information
 - How do we categorize requirements?
 - What is the hierarchy of requirements?

3. Managing the Requirements
 - How do we gather, maintain and use requirements?

Elicitation

- Verification of documented requirements
 - Active Listening is the key
 - Initial documentation is iterative
 - Synthesis of inputs from stakeholders
 - Rigorous exploration of the domain
 - Document what *won't* be implemented
- Iteration
 - Requirements *will* change once the stakeholder can visualize the solution
 - Show screenshots, mockups, prototypes
 - Validate interpretation of requirements

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

27

<http://tynerblain.com/blog/2006/01/14/top-five-requirements-gathering-tips/>

1. Interviewing
2. Brainstorming
3. Documenting Use Cases
4. Prototyping
5. Analyzing Documents

<http://tynerblain.com/blog/2006/01/27/top-five-ways-to-be-a-better-listener/>

1. Use Active Listening
2. Attentive Body Language
3. Ask Questions
4. 100% Focus
5. Non-Verbal Attends

Validation

- Validation of interpretation of requirement
 - Use traceability to assure that supporting requirements will achieve parent-objective
- Happens at all levels within requirements
 - Will we achieve 10% growth if we allow users to do A,B and C but not D?
 - If we implement L,M,N, can user do B?
- Implementation
 - Will this design meet the specifics of N?
- Testing
 - Will testing X,Y,Z assure that L is properly implemented?

Baselining

- On date X, stakeholders / sponsors / customers agree – these *are* the requirements
- Any changes after date X have ripple effects
 - Must validate proposed changes in context of non-functional requirements
 - Must revalidate changed requirements
 - Must re-evaluate implementation design
 - Must revalidate scoping and schedule of implementation and testing
 - Must revalidate schedule of delivery
- Use traceability to manage the propagation, identify the ripples and dependencies

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

29

Getting signoff on a version of the requirements at a given date is key to negotiating changes in the requirements (and interpretation of the requirements).

Signoff may stagger as well – signoff of content for release 2 may happen half-way through release 1

Ripple-effects can't be overstated. Changes to priority of requirements, or details of requirements cause changes in implementation times/schedule, which cause changes in delivery schedule (of features), which cause changes to ROI of the project. This is a fact of life, but sponsors often aren't software professionals, and don't appreciate the impact unless told about it.

Getting signoff on a version of the requirements gives you a good starting point for entering into those discussions. It also gives your customer a tool to make sure you deliver.

In some more formal environments, signoff of each stage along the way may be required (e.g. signoff on use cases prior to developing the functional requirements that support them).

Traceability

- Goals are supported by use cases
- Use cases are supported by functional requirements
- Functional requirements may be supported by other functional requirements
- Non-functional requirements *affect* functional requirements

Traceability is the key tool for managing the ripple effects of changes. It's also important for bridging the gap between delivery/implementation schedule and feature-release product schedule.

Scheduling

- Different things for different people
 - Stakeholders need to know when they can achieve X (ROI, product launch, etc)
 - Users want to know when they can perform action Y
 - Developers need to know when functional requirement Z is needed
- Create different documents
 - Product roadmap showing areas of functionality
 - Release schedule showing when use cases are enabled
 - Delivery schedule showing when each functional requirement will be delivered
- Requires coordination with development / project managers responsible for implementation and testing

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

31

<http://tynerblain.com/blog/2005/12/22/communicating-a-delivery-schedule-with-use-cases/>

Management systems

- Enterprise Solutions
 - Planning / Strategic
 - FeaturePlan
 - Execution / Management
 - Borland CaliberRM
 - Telelogic DOORS
 - IBM (Rational) RequisitePro
- Home-grown
 - MS Office, +emails, IMs + post-it notes
 - Wiki
 - Wiki + Bugtracking
 - Focus on Prototypes instead of docs (iRise)

6 June 2006

Copyright ©2005-2006 Tyner Blain LLC

32

Caliber – good schema for archiving. poor support for incremental/rapid development. good for generating functional specs, weak for other artifacts. very limited support for searching repository

RequisitePro – enables integration with Rational Rose, very difficult for generating customized output documents

MSOffice – many people manage software/solution creation via PPTs (release schedule) Word (requirements, use cases) Visio (OOA/OOD) and Excel (anything – schedules, UI mockups, functional specs, project plans)

Wiki – some groups at Google use wikis to maintain requirements. Mozilla uses a wiki for their PRD (http://wiki.mozilla.org/Firefox:2.0_PRD)

Good references

- Karl Wiegers
 - <http://www.processimpact.com/>
 - <http://www.processimpact.com/goodies.shtml#reqs>
 - Software Requirements, 2nd Edition
- Blogs I like about requirements
 - <http://tynerblain.com/blog>
 - <http://requirements.seilevel.com/blog/atom.xml>
 - <http://feeds.feedburner.com/MichaelHighTechPM>
 - <http://feeds.feedburner.com/FromStartToEnd>

Thank you

- Thanks very much for the time you spent with me today. If you have any questions about the content, or about applying this stuff in the real world, please don't hesitate to contact me

• scott@tynerblain.com