

IBM Software

Innovate2012

The Premier Event for Software and Systems Innovation

Next  NOW!

Agile Development using Visual Requirement Definition and Management

In Rational Requirements Composer 4.0

Jared Pulham

Senior Product Manager, Requirements Management
jared.pulham@uk.ibm.com



Introduction

The advantages of applying lean and agile techniques to software design and development activities of are now well established and understood in IT focused organisations.

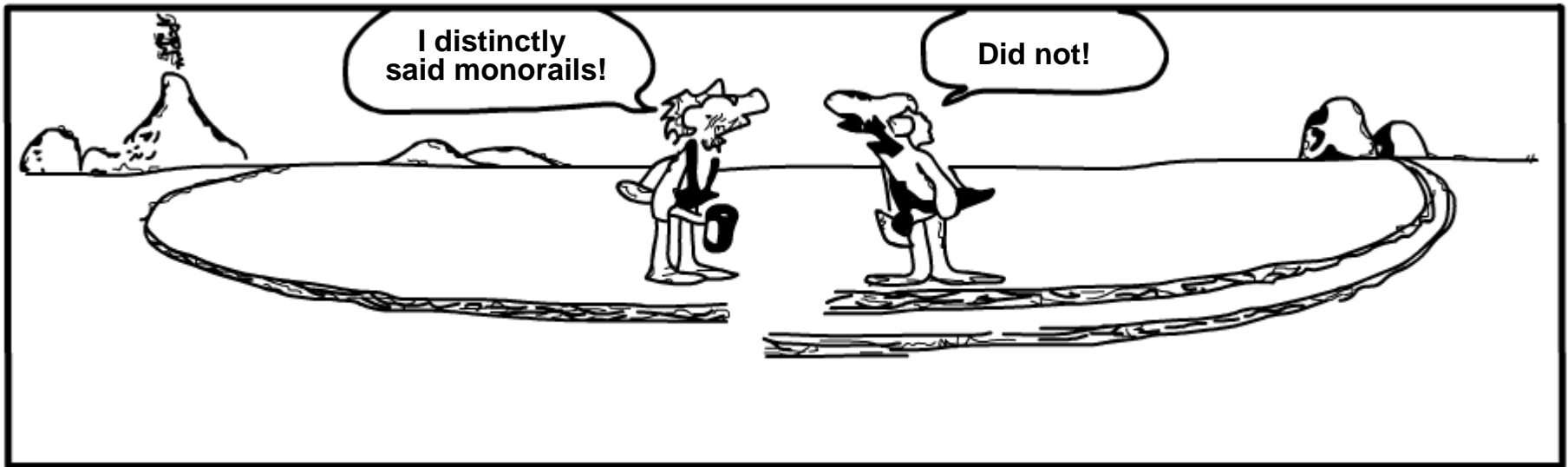
Often many IT organisations who have implemented agile techniques continue to struggle because they either do business analysis activities through a waterfall process or skip upfront requirements analysis completely as proposed by agile purists. How do we use requirements definition and management in an agile process?

This session provides a framework for evaluating best use of various type of requirements definition and management in agile development. It also explains some of the iterative analysis approach being used by a some customers today, their lessons learnt and plans for further deployment including using the latest version of IBM Rational Requirements Composer 4.0.



The need for requirements management

B.C. by Johnny Hart



Why people ignore requirements

No perceived value

Requirements shouldn't be just a box to check in at the front of the development process

Nothing in place to USE the requirements

In the past requirements took a lot of time and just sat on the shelf

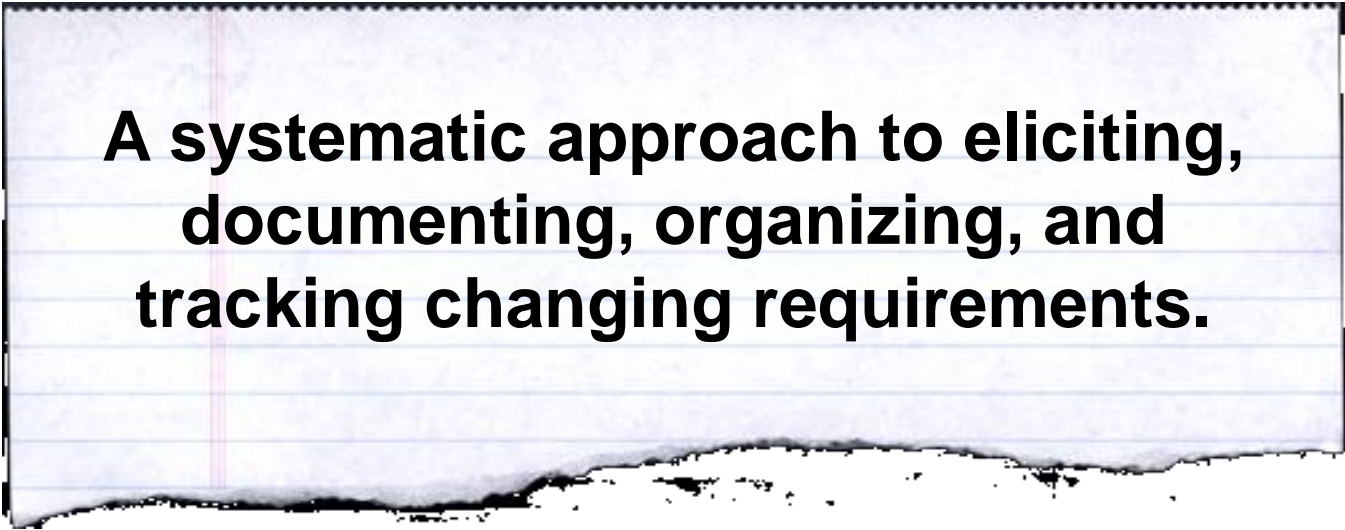
They always change anyway, so why gather them in the first place?

Unmanaged change is very frustrating



What is requirements management?

Ensuring that your team identifies, builds, tests and documents the right system for your customer



A systematic approach to eliciting, documenting, organizing, and tracking changing requirements.



Who needs requirements?



All project team members need access to requirements



Analyzing how much Analysis to do

How much Requirements Analysis?

Agile purists who argue ‘do none or at the most don’t do much because the requirements will change’

“Rather than coming up with a bunch of features and planning a multi-month release, come up with new ideas continually and try them out individually on users.” 1

Traditionalists who want to do as much as possible, because we need to know we are doing the right thing before investing

“For the second consecutive year, IAG found poor requirements definition and management consume over one-third of IT’s application development budget.” 2

Context Determines the Approach

Both the agile approach and the verifiable approaches to requirements engineering are appropriate in their own context. Projects with a lot of change that need to get out to the market quickly might be best done with high-level, low-ceremony requirements practices.

Stable projects with safety-critical implications could best be done with a plan-driven, well-documented specification.

1 <http://www.informit.com/articles/article.aspx?p=1829417>

2 <http://esj.com/articles/2009/09/29/wasted-it-development-spending.aspx>

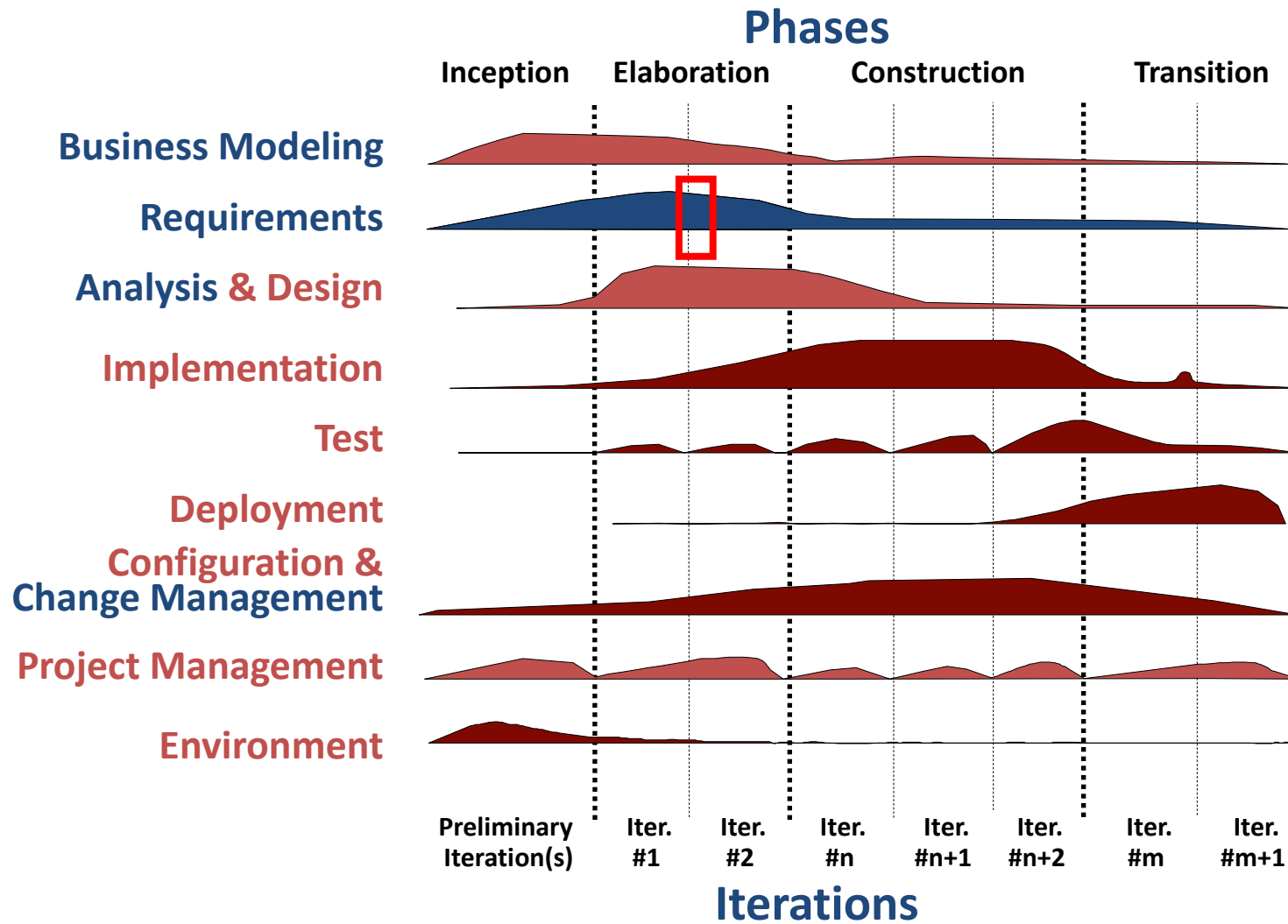


Requirements Management Key Activities

- ✓ Analyze the Problem
- ✓ Understand Stakeholder Needs
- ✓ Define the System
- ✓ Manage the Scope of the System
- ✓ Refine the System Definition
- ✓ Manage Changing Requirements



Iterative development process



Analyze the problem

Users and customers don't want systems – they want their problems solved

Solving the wrong problem well or fast doesn't help

The problem as first stated is rarely the true problem

Understand the problem

- Determine the purpose

- Look for root causes

Gain agreement and document the problem as appropriate



Understand the problem your solution will solve



Eliciting requirements

Getting requirements requires people skills

Often times user don't know what they want

Some time user's know what they want but can't express it

You need skills and techniques for getting good requirements

“The closest distance between two points in human affairs is usually not a straight line!”



Understand stakeholders needs

Elicitation options

Requirement workshop

Interviews

Role playing

Prototypes/Working Software

User Stories/Storyboards

Use-case workshop

Ensure requirements meet user's needs



Define the system

Identify product features

Create a 'big picture' of the solution

Create a supplementary specification

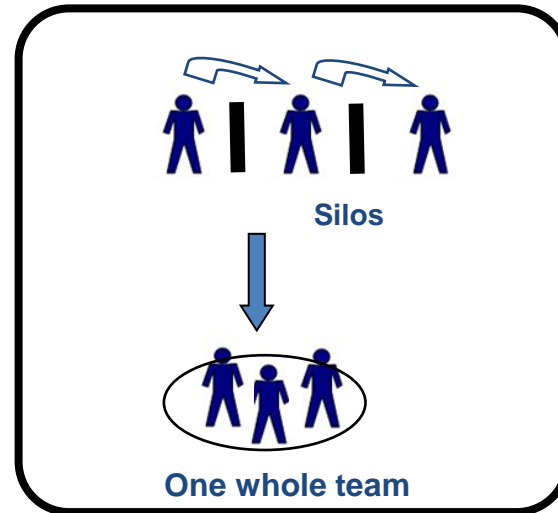
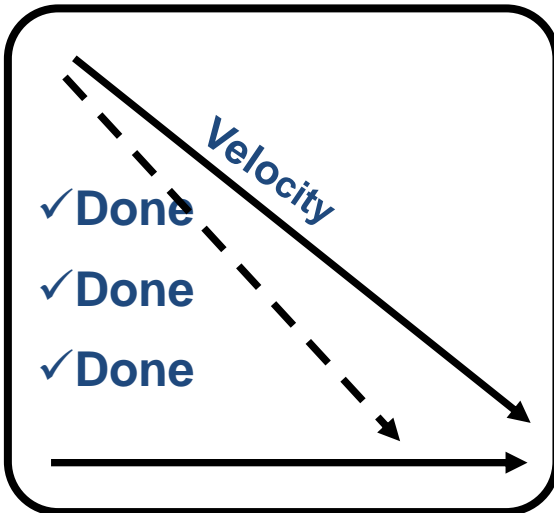
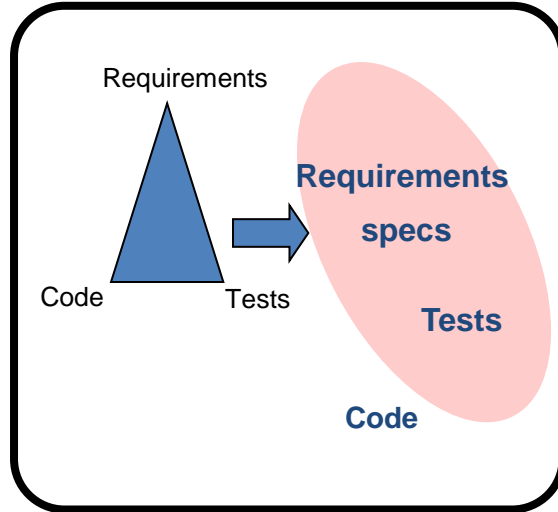
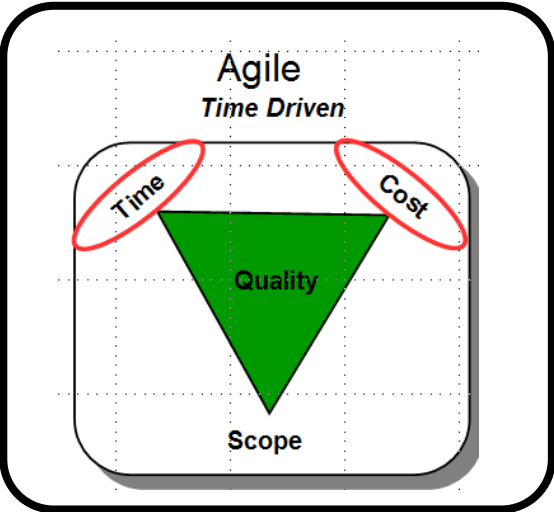
- Non-functional requirements

Review the vision document

- With the team and with the customer

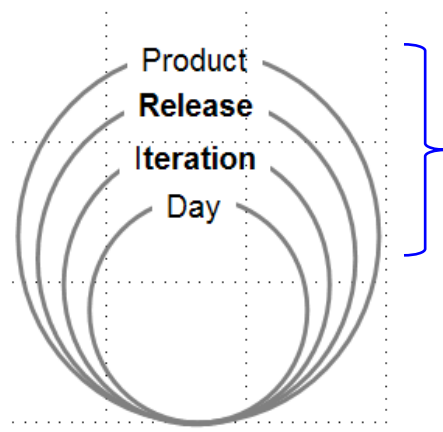
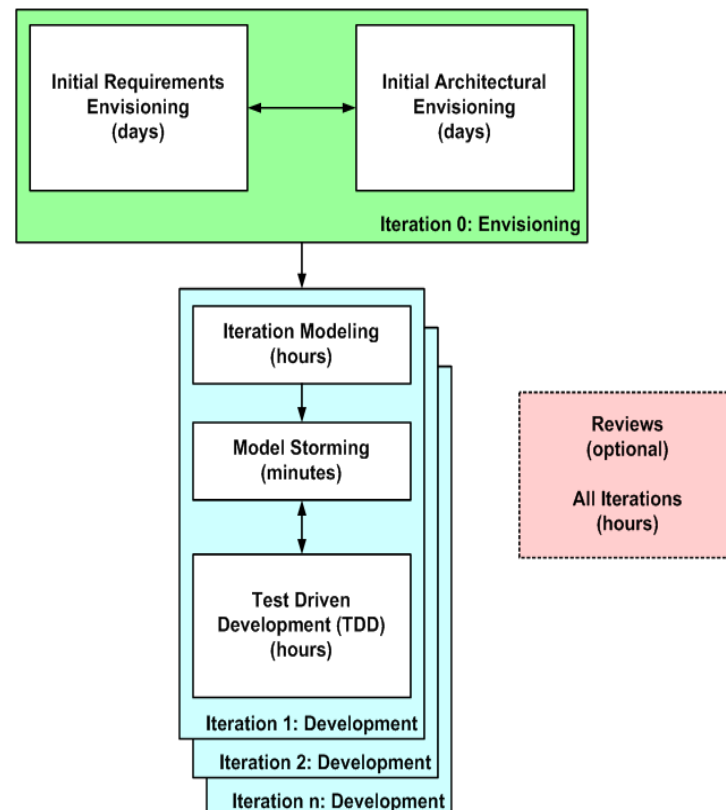


Consider an Agile Approach



The Agile way of defining requirements

- Initial requirements are initially envisioned at a very high level .
- The goal of the requirements envisioning is to identify the high-level requirements as well as the scope of the release (what you think the system should do).



Most agile teams are concerned only with the three innermost levels of the planning onion

Mike Cohn (2008)



Agile requirements gathering techniques

Story telling

Story cards

Story boards and sketches

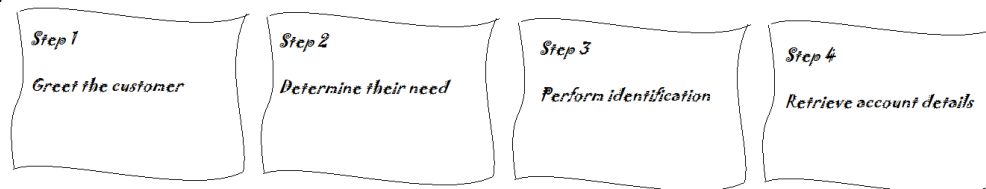
User stories and Story Points

Requirements stacks

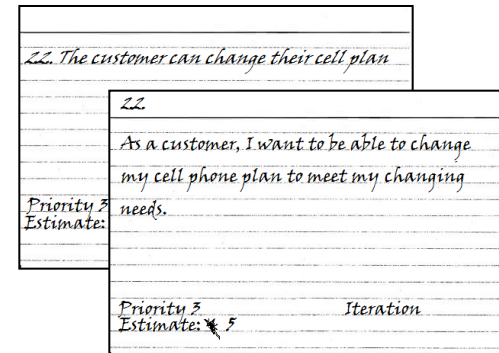
Writing just enough requirements

Talking rather than writing

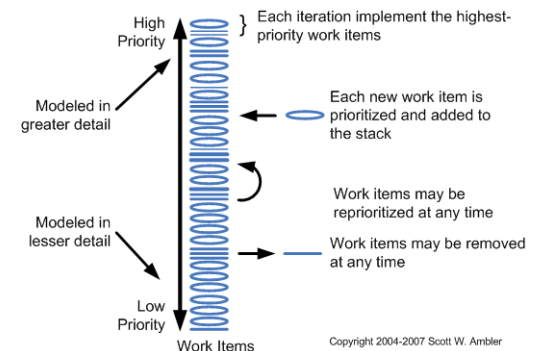
Not designing screens too early



Storyboards



Story cards

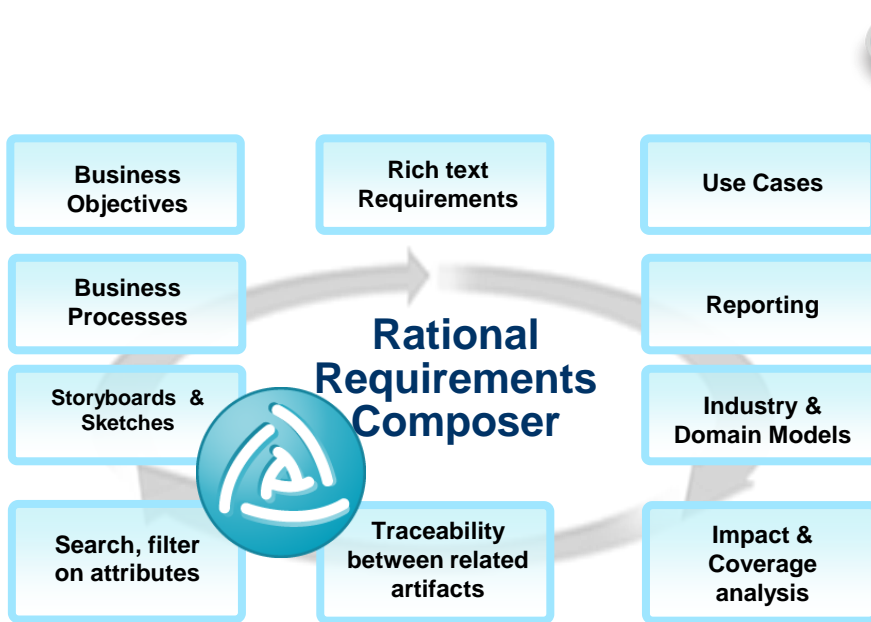


Backlog stack



IBM Rational Requirements Composer

Project driven requirements management for your global team



**Better requirements... Less rework...
Better results!**

Combined Definition and Management

- Empower teams of all sizes/complexity to capture, define, analyze, manage, report
- Clear, centralized requirements eliminate redundancy and aid real-time development
- Develop using agile-at-scale and iterative processes in volatile markets

Lifecycle Solutions and Collaboration

- Align business, development and test effort using light-weight process
- Move beyond file based management with easy Word/Excel migration
- Engage project stakeholders early and regularly collaborate to improve quality

Improved Planning and Visibility

- Realise visibility using traceability across requirements, test, and development
- Provides up-to-date reporting based on your requirements
- Manage scope of development project



IBM Rational Requirements Composer 4.0

Requirements Management for the Development Lifecycle

Rational Requirements Composer

Definition

- Rich-text documents
- Diagrams: Process, Use Case
- Storyboards, UI sketching & flow
- Project glossaries
- **Templates (formal/agile)**

Visibility

- Customizable dashboards
- **Project dashboards**
- Analysis views
- Collections
- Milestone tracking & status

Collaboration

- Review & Approval
- Discussions
- Email Notification

Improved!

Improved!



Supports RequisitePro Data Migration

Management

- Structure, Attributes/Types
- **Traceability, Suspect Link**
- Filtering, Change History
- Tags, Reuse, Baselines,
- Reporting Metrics & Doc.

Lifecycle

- Central requirements, test, & development repository
- **WAS Clustered Server**
- Common admin and role-based user licensing
- Warehouse reporting

Planning

- Integrated planning
- Effort estimation
- Task management

Improved!

Improved!



User Stories and Story Cards

A conversation with the end user

Used to capture the customer's requirements as simple statements or 'features'

Written on cards

Used by development team to flesh out the user tasks

Can be estimated with Story Points

Can be fast tracked or delayed by varying the priority

Good for small work items

Not good for communicating between projects or over time

less formal



more formal



Sketching

Actively involve the customer in design decisions.

Less threatening, more interactive.

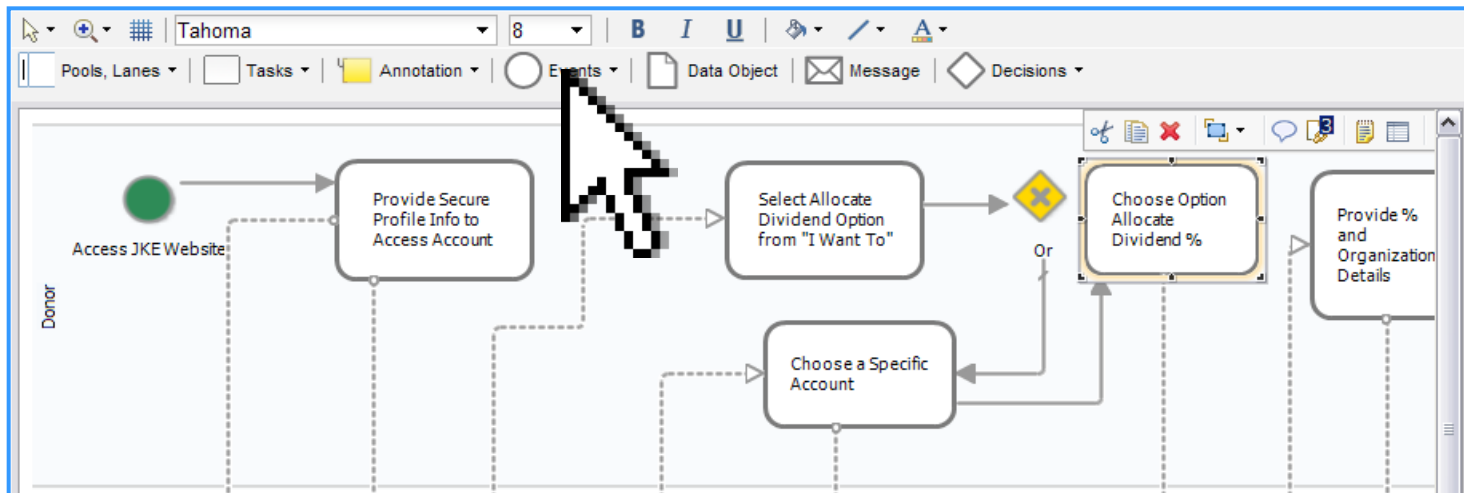
Can be applied to many situations, not just programming.

Easy to change.

Suits common office equipment like whiteboards and whiteboard markers.

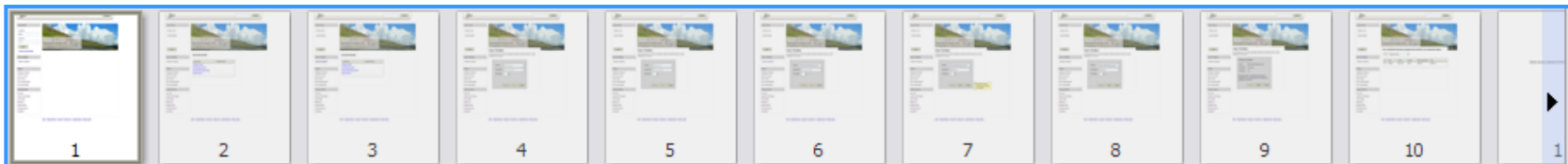
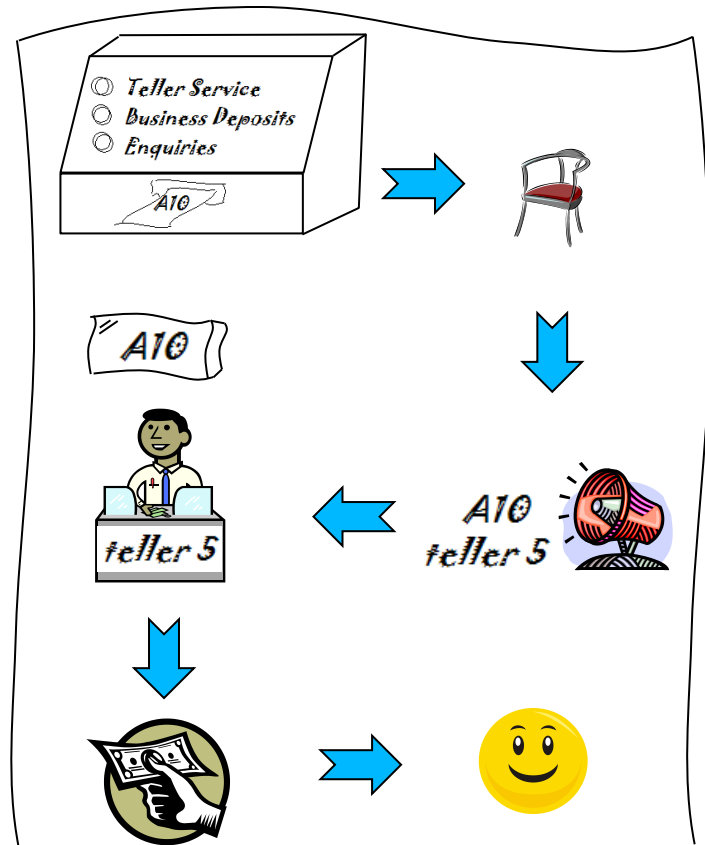
Take snapshots at regular intervals.

Cross-index to user stories.



Storyboards

- A series of sketches that tell the user task.
- Can be simple, sketched on a whiteboard and photographed.
- Good to gain agreement of steps.
- If informal, capture the customer's thoughts at a point in time.
- Can be a mixture of Clip Art and text.
- Visually rich.
- If more formal, takes time to create and maintain.
- Can't be changed once captured as a photograph.



Write just enough - and then stop

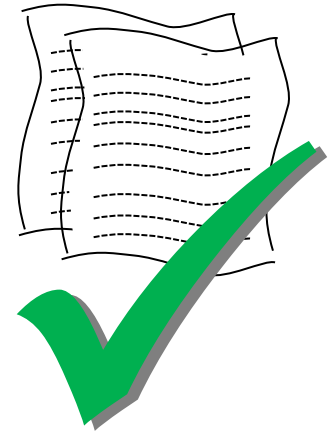
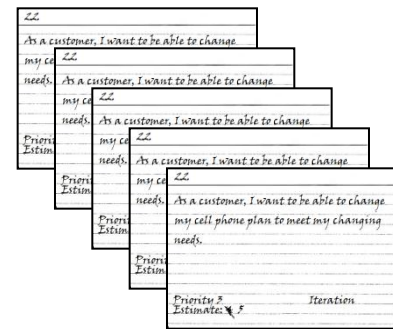
Completeness and exhaustiveness are not the same thing

Gather the requirements at the right time

Know your target audience

Do not follow templates slavishly

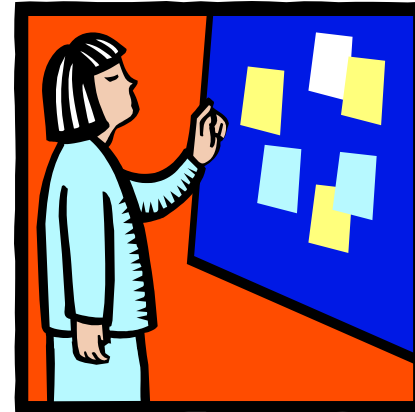
Use the right device



Managing requirements in an agile environment

The progress board

Shows the status of all cards at a point in time – but don't forget to capture it as it changes!



Moving to more rigorous techniques – traceability, flexibility and maintainability

Maybe even a software tool or two

	Feature 2	Feature 4	Feature 6	Feature 9	Feature 14	Feature 26	Feature 28
Task 2		X			X		
Task 17					X		X
Task 22			X		X		
Task 35	X						

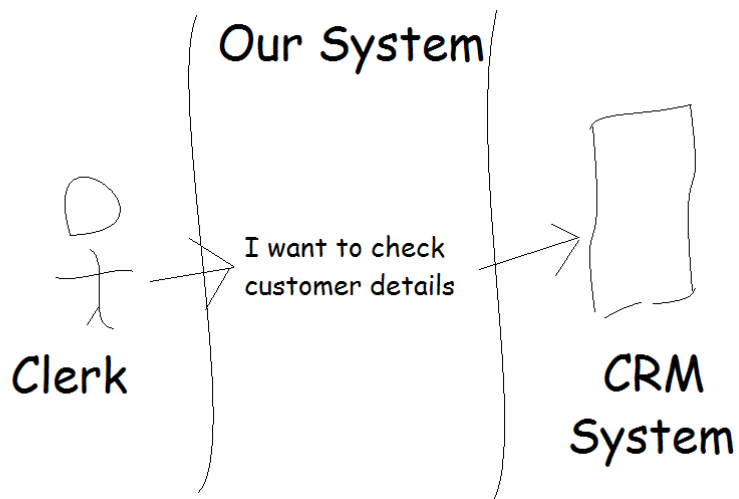
Avoiding introducing too many new requirements late in the lifecycle



What about the Use Case?

The Use Case can be agile too...

“Call me Function”

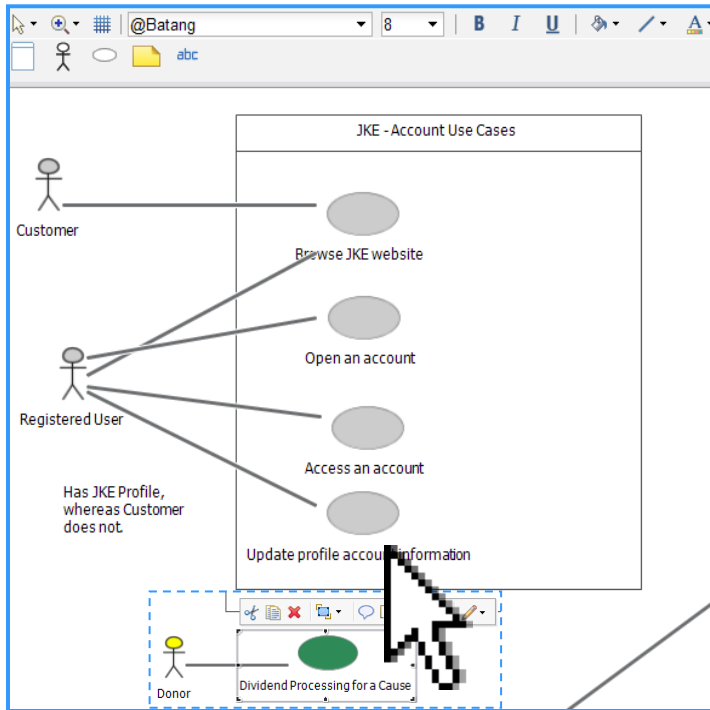


<i>35 "Check customer details"</i>
<i>Step 1 Select "Check customer details"</i>
<i>Step 2 Enter the customer ID</i>
<i>Step 3 Confirm right customer</i>
<i>Step 4 View details</i>
<i>Step 5 Take a print out</i>
<i>What if.. No customer ID?</i>
<i>Search on name instead</i>

It's a combination of sketch and story card



Use cases are graphical...but mostly textual



Identified graphically

Described textually

The screenshot shows the "RM Product Definitions (RM)" interface. The main content area displays "330: Common - Link Suspicion". Under the "Brief Description" section, there is a note: "Note: This type of feature is often called 'Suspect Link'." The main text explains that users need link relationships between project information content for different requirements on to other development life-cycle artifacts. It further states: "Links are suspect or not suspect depending on the user's perspective and which direction the link is being examined. For example suppose that the requirement for a particular link was modified since the link was created. The person making a modification typically is aware of links in requirements owned by them and they make requirement changes in the context of the link presence. From the perspective of this requirement, the change modification and the link would not be suspect. However from the perspective of the opposing requirement (on the opposite side of the link) the modification may be significant and that change was made without their knowledge. So therefore from the perspective of the opposing requirement (or link) the change is suspect. The user on the other end of the link needs an automated way of recognizing this change and provided with the features to examine the change and remove the suspicion or resolve the suspicion by changing the value of the requirement on the other end of the link to validate the link relationship."

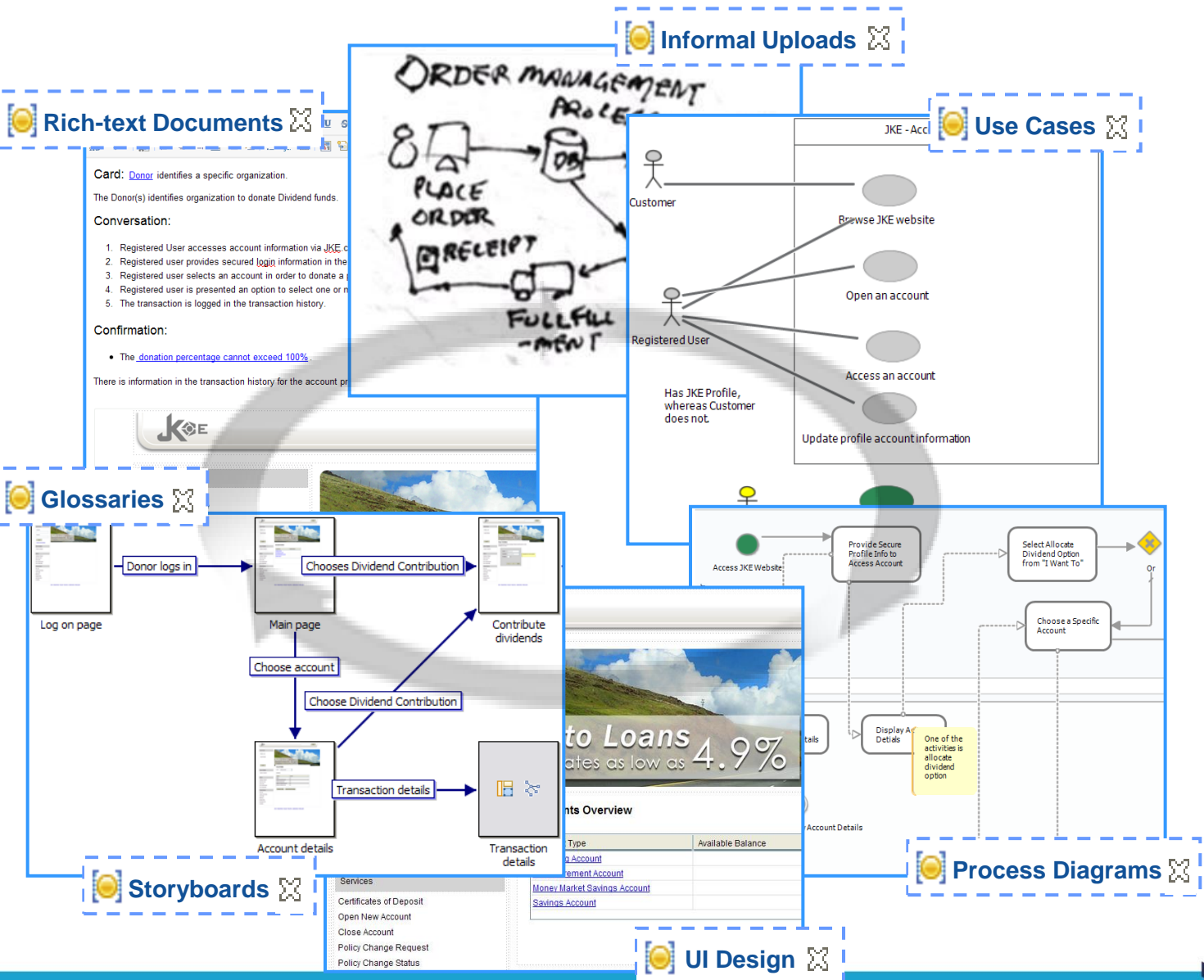
Business Justification

Link Suspicion in a requirements management tool is vital to many businesses and critical to those who rely on traceability consistency in their development process. Link Suspicion is now a well understood capability (not market differentiating) and would be quickly identified as a significant market gap if not provided as a feature or properly implemented. There could be ideas or methods to extend the basic concept of link suspicion to create something market differentiating.

It is a feature that was originally developed in a requirements management discipline for tools like RequisitePro and DOORS. It is a capability that is mandatory to many organization's processes and extends beyond requirements to any tool in the development lifecycle where traceability has been applied.

Scope

Use Visual Scenarios to Uncover Customer Needs



- Defining requirement flows using scenarios to uncover missing critical details
- Text requirements link to diagrams to complete the development picture
- Visualise your development results through a variety of requirements forms
- Traceable elements helps ensure complete coverage thinking



Benefits of use cases

Facilitate efficient communication between end users and customers, and the development team

- Provide context around requirements by expressing sequences of events

- Use case diagrams act as a 'big picture' of the system

Defines what the system does to satisfy its stakeholders

Help reduce design constraints

- Focus on the "what" not the "how"

Are reusable by the rest of the team

- For design, usability design and testing



Use a Product Backlog with Context

Epics and Stories

High Level Requirements

- Shows how stories fit together
- Shows which are completed
- Shows how we have ranked them

As an End User, I want to search the Catalogue, so I can find the item k

As a customer support person, I would like to search the catalogue for histo

As a customer I would like to search the catalogue, in order to find the item

As a Customer, I would like to search the catalogue, in order to find similar

As a customer I would like to search recommended items in the catalogue,

As a Customer, I would like to search the catalogue, in order to find similar products to the ones

As a customer I would like to search recommended items in the catalogue, in order to see recor

Catalogue Search results are taking too long

Index Setup for Catalogue Search may need

Catalogue Search results are taking too long

Index Setup for Catalogue Search may need to be more robust

TECH Story: Query System needs indexing to to return all queries

Story Points	Priority	Rank
--	High	1
5 pts	High	3
20 pts	High	--
5 pts	High	--
8 pts	...	5
--	High	4

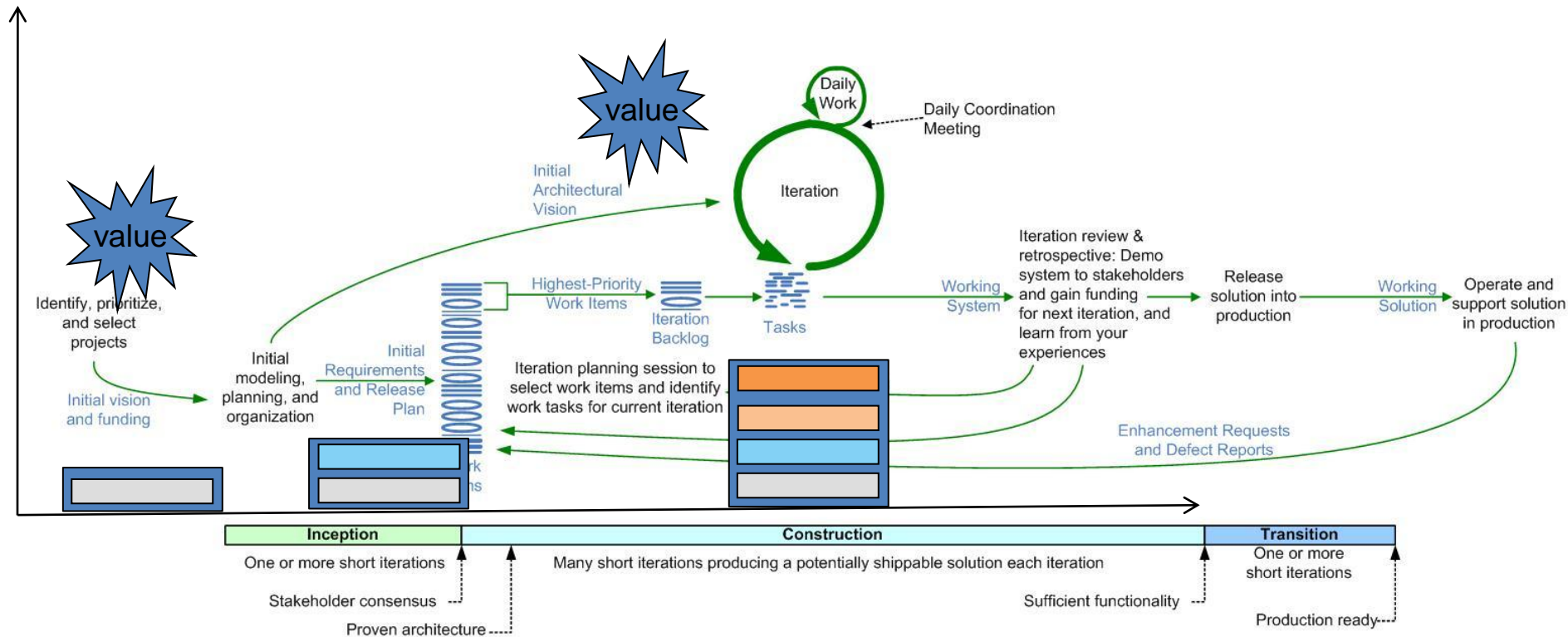
Other Rankable 'Requirements'

Showing Context in the Backlog

- Shows what isn't done
- Shows Architecture concerns
- Shows were other things rank



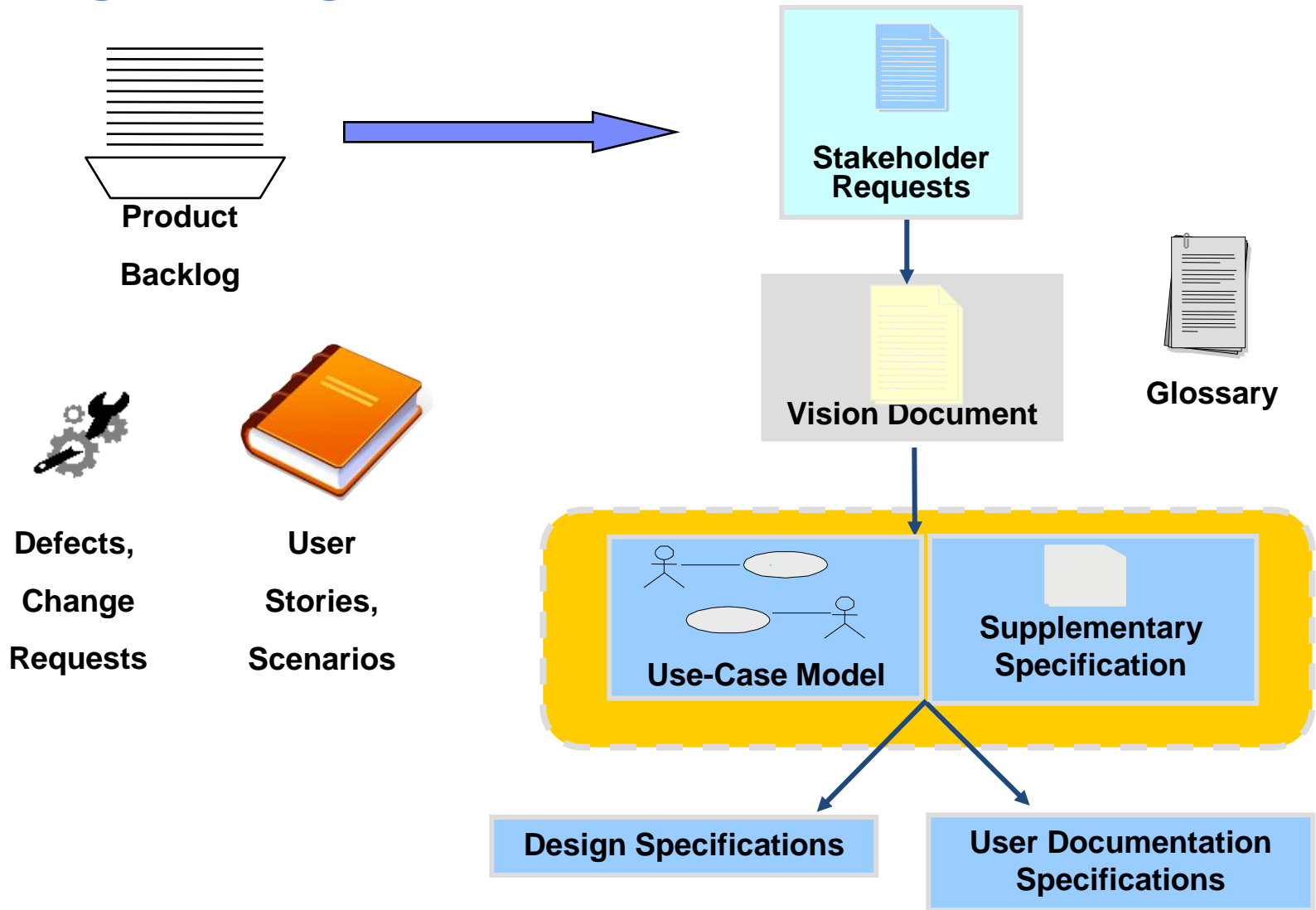
Elaborate Requirements Progressively



Growing details over time



Putting it All Together



User Requirements (RRC)

Summary	Id
DNextBeta1 PMC_Req UX [RM] Core Module support	44243
CLM PMC_Req UX [RM] Suspect links 169220/164497	44275

9929: Common - Module

Brief Description
Module is where people author, structure, reuse, link requirements, and...

Business Justification
Module is a key feature that brings together different RM capabilities to...

Scope

Note:
This document defines some User Requirements that may be broken...

Definitions
In this document we have borrowed definitions from [929: Common - S...](#)

- **Originating requirement / artifact** - a requirement / artifact that
- **Requirement instance / artifact** - a new instance of an originat

Key Functions

- **Modules as Artifacts**
 - A Module shall behave like any existing RRC Artifact. By this we mean that the set of base behaviors we see in today's RRC Artifacts (Text Artifacts etc) are also present in Module Artifacts. For example, from the mile high perspective of the RRC Project Dashboard, Module Artifacts would be virtually indistinguishable from Text Artifacts.
 - The Module as an Artifact is a completely separate concern from the Module's role as a container of Artifacts. This is how Collection behaves today.
 - So with respect to Modules as Artifacts in their own right, the following operations can be applied:
 - Tagging
 - Commenting
 - Reviews

330: Common - Link Suspicion

Brief Description
Note: This type of feature is often called "Suspect Link".

Users need link relationships between project information content for different requirements on to other development life-cycle artifacts. After a relationship link is created between two requirements (or two ALM elements) users need an automated method to identify the (data) change status to the artifacts on both ends of the links. If the information content of one of the artifacts changes, the link (relationship) may no longer be appropriate or valid. In this case, we would say that the link has become "Suspect" or there is "Change Suspicion".

Links are suspect or not suspect depending on the user's perspective and which direction the link is being examined. For example suppose that the requirement for a particular link was modified since the link was created. The person making a modification typically is aware of links in requirements owned by them and they make requirement changes in the context of the link presence. From the perspective of this requirement, the change modification and the link would not be suspect.

However from the perspective of the opposing requirement (on the opposite side of the link) the modification may be significant and that change was made without their knowledge. So therefore from the perspective of the opposing requirement (or link) the change is suspect. The user on the other end of the link needs an automated way of recognizing this change and provided with the features to examine the change and remove the suspicion or resolve the suspicion by changing the value of the requirement on the other end of the link to validate the link relationship.

Business Justification
Link Suspicion in a requirements management tool is vital to many businesses and critical to those who rely on traceability consistency in their development process. Link Suspicion is now a well understood capability (not market differentiating) and would be quickly identified as a significant market gap if not provided as a feature or properly implemented. There could be ideas or methods to extend the basic concept of link suspicion to create something market differentiating.

It is a feature that was originally developed in a requirements management discipline for tools like RequisitePro and DOORS. It is a capability that is mandatory to many organization's processes and extends beyond requirements to any tool in the development lifecycle where traceability has been applied.

Scope

Key Functions

Overview
Comments (5)

3. **Kirk Grotjohn to Daniel Moul, Jared Pulham**
Aug 16, 2011 (1 reply)
We don't really have "owners"
I don't believe "Owner" is a system attribute, so I'm not sure we can base things on artifact, document or collection owners.

5. **Jared Pulham to Daniel Moul, Kirk Grotjohn**
Aug 17, 2011
RE: We don't really have "owners" [RE: #3]
The term owner was really just referencing the user of the collection, document, etc. I've changed the word...as these are user level requirements. Thanks for the feedback.

1. **George DeCandio to Jared Pulham, tina zhuo**
Jul 20, 2011 (1 reply)
What about some attributes not triggering suspicion?
Some customers have asked to allow some attributes changes on requirements not to trigger suspect state. Examples include development sizings, dev priority, etc...

4. **Jared Pulham to George DeCandio, tina zhuo**
Aug 17, 2011
RE: What about some attributes not triggering suspicion? [RE: #1]
Yes, absolutely. I was surprised that I missed...

Links
Where Used

Manage the scope of the system

Most projects try to do too much

Scope

- Functionality to be delivered

- Resources to do the work

- Time available for completion

When you can't do it all – how do you
decide what to leave out?

Prioritize requirements based
on Customer priority first

Access the effort

Manage scope throughout the project



Conclusion/Summary

Apply Agile principles and take them to heart

- No more kicking requirements over the wall
- No more big requirements documents
- Become embedded in the team and the process

Become part of the full project lifecycle

- Realise requirements is an ongoing process throughout project
- Prepare to be a part of the team for longer time frame, through many iterations/sprints
- Become embedded in the Quality aspect of the lifecycle

Embrace change!

- Embrace the organisational change that comes with agile
- Embrace constant change to the project scope/requirements/needs/priorities

