SENG321: Requirements Engineering

CHANGE MANAGEMENT AND

Dr. Daniela Damian

thesegalgroup.org

danielad@uvic.ca

Stakeholders and an interesting relationship

Many classes of stakeholders

Different requirements

Diverse market/political issues

–-> RE = continuous negotiation process

Software evolution and change management

Change in software development: as inevitable as difficult to control

- Reasons for change:
  - New requirements become apparent
  - Business and strategic goals change
  - Changes in customer market
- Possible responses to change:
  - Add new requirements
  - Modify existing requirements
  - Remove existing requirements

Change management: largely a project management activity

Change management techniques

1. Formal change management process

2. Requirements traceability

3. Release planning and Requirements prioritization

Formal change management

(traditional) Configuration management

- Versioning of the Requirements Document (RD)
- Baselining the RD after the requirements analysis was complete

The latest RD should include the current understanding and agreement wrt requirements

Change control board in charge of reviewing and approving change requests

Changes between versions should be documented

2. Requirements traceability

Backward traceability

- To previous stages in development
- Most important: traceability to requirement rationale (pre-RS traceability)

Forward traceability

- To all documents/artifacts that follow RD (post-RD traceability)

Important for:

- Assessing change requests
- Project management: progress visibility, risk management

2. Requirements traceability

Easier in theory than in practice

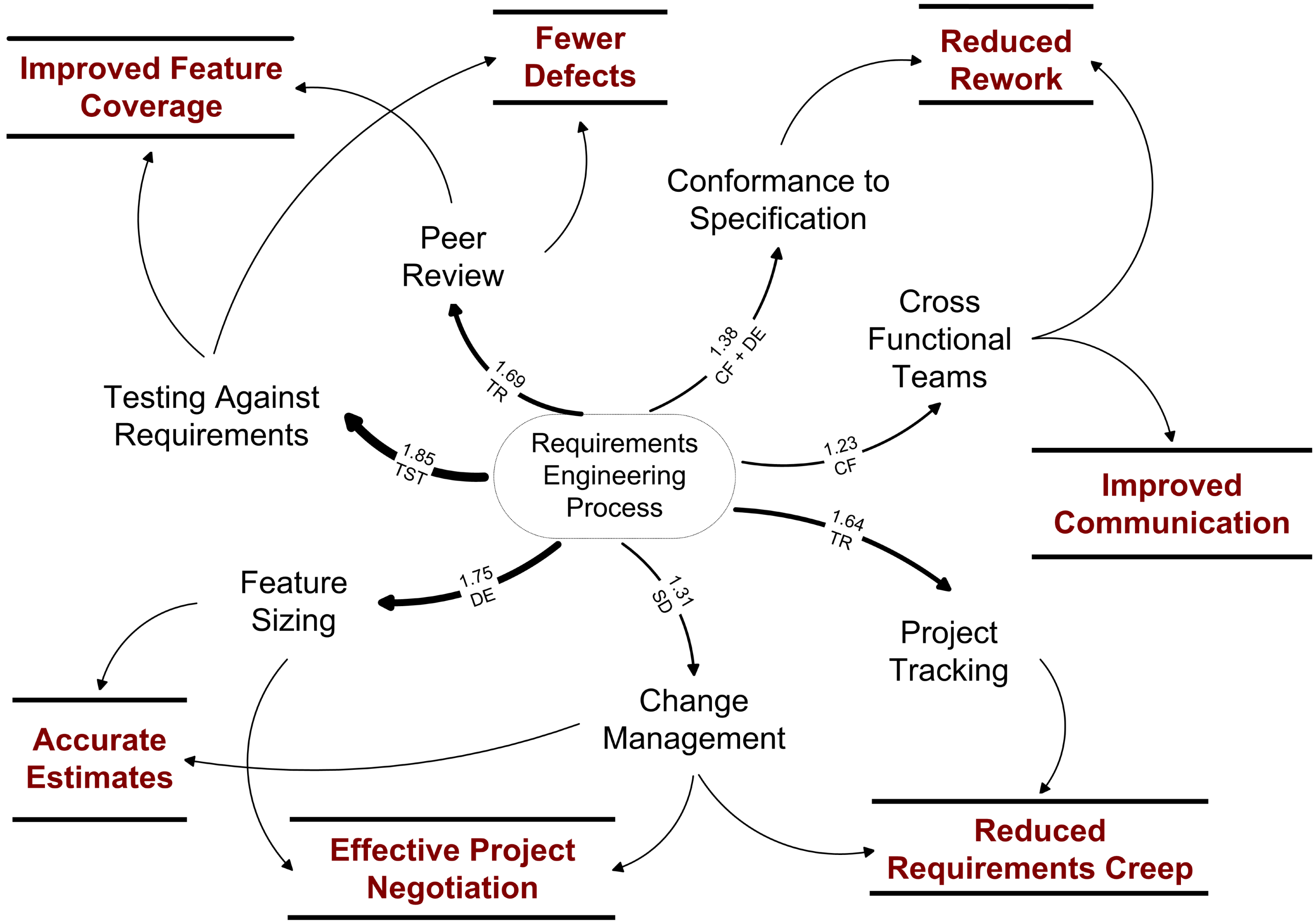
Costs may not even out for those who get the benefits

Weak current tool support in:

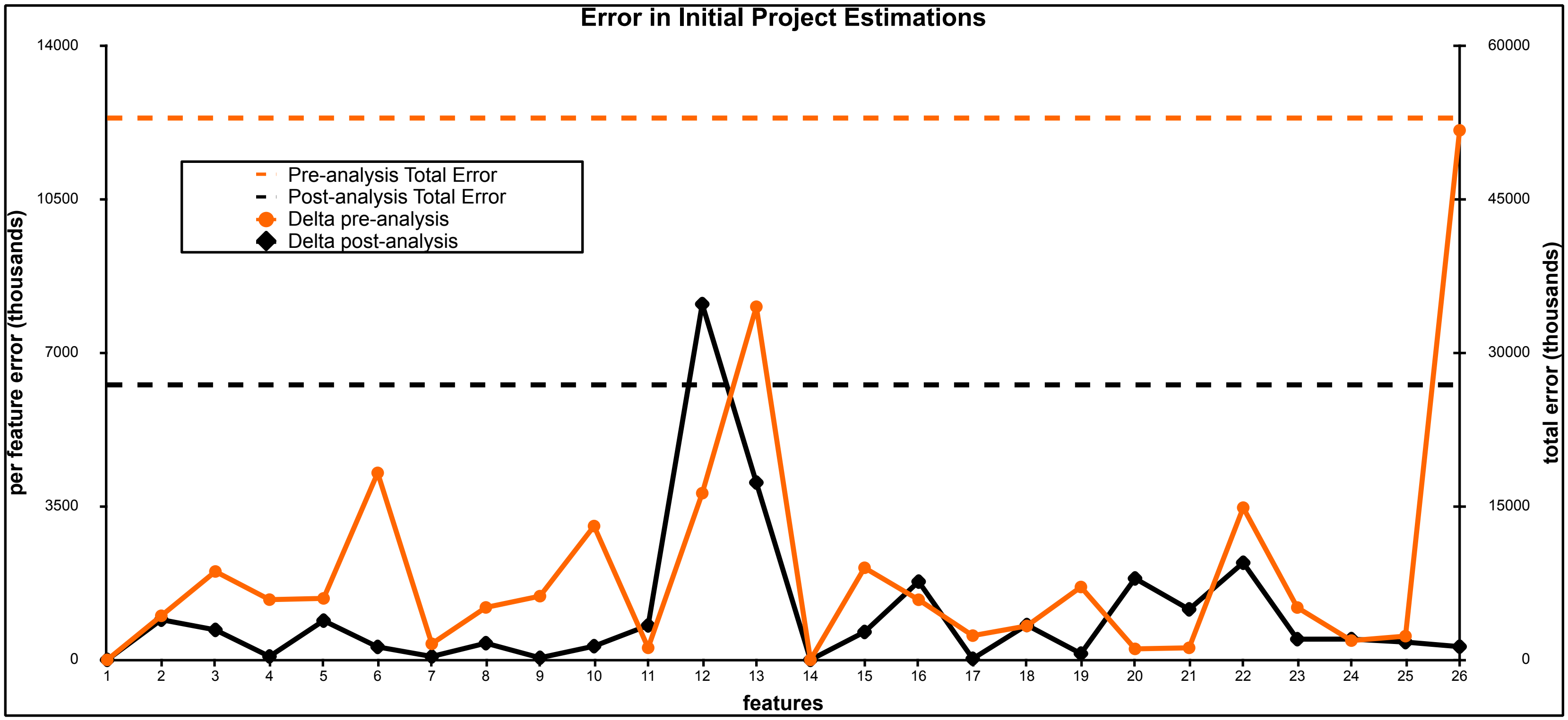
- Tracing between different tools
- Capturing informal communication

An example: Requirements traceability at Unisys

Rich interaction between the RE Process and other development processes contributed to gains in **productivity** (communication, rework), **quality** (defects) and **risk management** (estimations, feature coverage, negotiation, requirements creep).



Success stories: Significant improvements in estimation ability due to good RE processes



An example: Requirements traceability at Unisys

Sample Requirement:

Initial Feature: Scriptable Interface
Derived Technical Requirement: EA Developer shall provide a version of the D2L utility that allows the exchange of affected screen definitions from Graphical Interface Workbench to EA Developer.
Rationale: Customers may already have painted versions of the affected screens in GIW. They will likely wish to keep those existing painted screens, and be able to enhance them in the future.
Test Scenario: Enhance screens in a GIW environment. Load the provided model into EA Developer. Use D2L to transfer the enhanced screens from the GIW environment to EA Developer, into the installed model. Verify that the enhanced screens are available in EA Developer and can be further enhanced in EA Developer Painter.

Requirements Traceability Matrix, examples

REQUIREMENTS TRACEABILITY MATRIX					
Project Name: Online Flight Booking Application					
Business Requirements Document BRD		Functional Requirements Document FSD			Test Case Document
Business Requirement ID#	Business Requirement / Business Use case	Functional Requirement ID#	Functional Requirement / Use Case	Priority	Test Case ID#
BR_1	Reservation Module	FR_1	One Way Ticket booking	High	TC#001 TC#002
		FR_2	Round Way Ticket		TC#003 TC#004
		FR_3	Multicity Ticket booking	High	TC#005 TC#006
BR_2	Payment Module	FR_4	By Credit Card	High	TC#007 TC#008
		FR_5	By Debit Card	High	TC#009
		FR_6	By Reward Points	Medium	TC#010 TC#011

Requirements Traceability Matrix, examples

Test Case ID#	Test Case Description
TC#001	Verify if user is able to book one way ticket
TC#002	Verify if user is able to book multiple one way tickets
TC#003	Verify if user is able to book round way ticket
TC#004	Verify if user is able to book multiple round way tickets
TC#005	Verify if user is able to book one way ticket for multiple cities
TC#006	Verify if user is able to book round way ticket for multiple cities
TC#007	Verify if usre is able to pay by Master Card
TC#008	Verify if usre is able to pay by Visa Card
TC#009	Verify if usre is able to pay by Debit Cards
TC#010	Verify if usre is able to pay Fully by Reward Points
TC#011	Verify if usre is able to pay partially by Reward Points

(An example) How to build a Requirements Traceability Matrix

Business Requirements (BR)

BR1: The system should allow the user to book one or more tickets, one way or around way for future dates

BR2: The user should be able to make payment for booked tickets via Credit/Debit Card or through Reward Points

(An example) How to build a Requirements Traceability Matrix

BR1: The system should allow the user to book one or more tickets, one way or around way for future dates

FR1: One way Ticket booking
The system should allow the user to book one way ticket

FR2: Round way Ticket booking
The system should allow the user to book round way ticket

FR3: Multiplicity Ticket booking
The system should allow the user to book one way or round way ticket for multiple cities

(similar for BR2..)

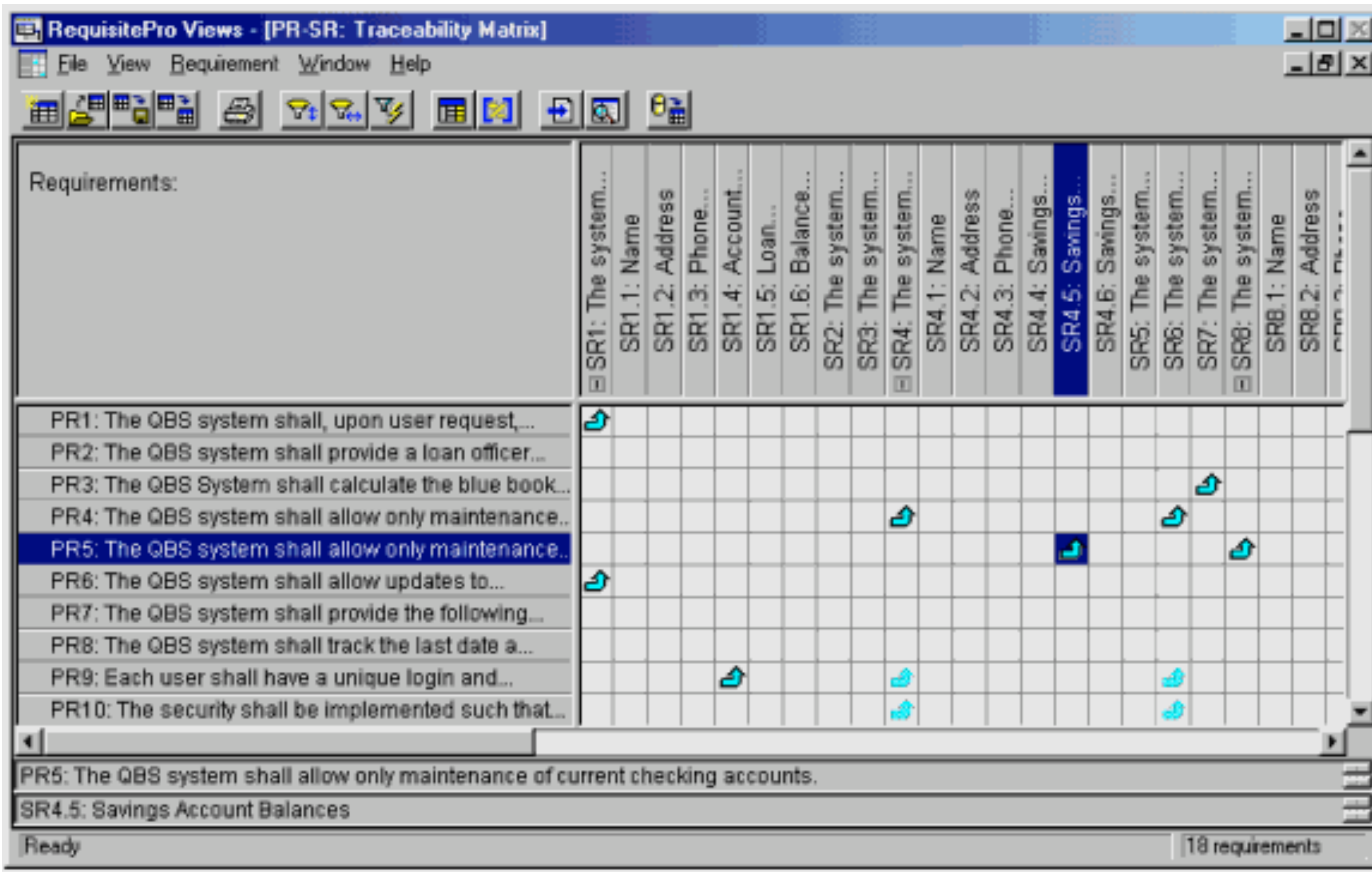
Requirements Traceability Matrix, examples

Test Case ID#	Test Case Description
TC#001	Verify if user is able to book one way ticket
TC#002	Verify if user is able to book multiple one way tickets
TC#003	Verify if user is able to book round way ticket
TC#004	Verify if user is able to book multiple round way tickets
TC#005	Verify if user is able to book one way ticket for multiple cities
TC#006	Verify if user is able to book round way ticket for multiple cities
TC#007	Verify if usre is able to pay by Master Card
TC#008	Verify if usre is able to pay by Visa Card
TC#009	Verify if usre is able to pay by Debit Cards
TC#010	Verify if usre is able to pay Fully by Reward Points
TC#011	Verify if usre is able to pay partially by Reward Points

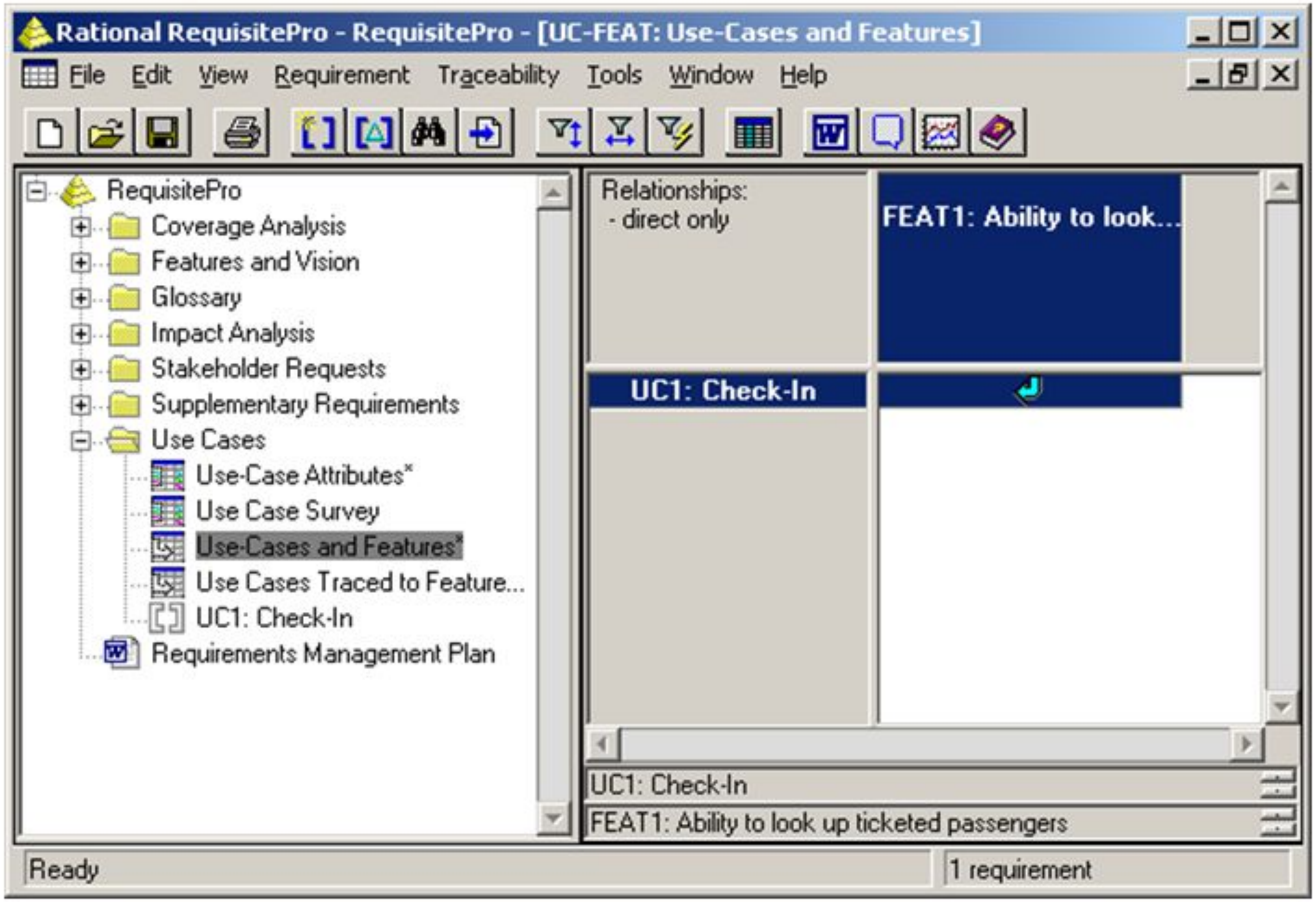
Requirements Traceability Matrix, examples

REQUIREMENTS TRACEABILITY MATRIX					
Project Name: Online Flight Booking Application					
Business Requirements Document BRD		Functional Requirements Document FSD			Test Case Document
Business Requirement ID#	**Business Requirement / Business Use case**	**Functional Requirement ID#**	**Functional Requirement / Use Case**	**Priority**	**Test Case ID#**
BR_1	Reservation Module	FR_1	One Way Ticket booking	High	TC#001 TC#002
		FR_2	Round Way Ticket		TC#003 TC#004
		FR_3	Multicity Ticket booking	High	TC#005 TC#006
BR_2	Payment Module	FR_4	By Credit Card	High	TC#007 TC#008
		FR_5	By Debit Card	High	TC#009
		FR_6	By Reward Points	Medium	TC#010 TC#011

Requirements Traceability Matrix, ReqPro tool



RequisitePro – Traceability Matrix



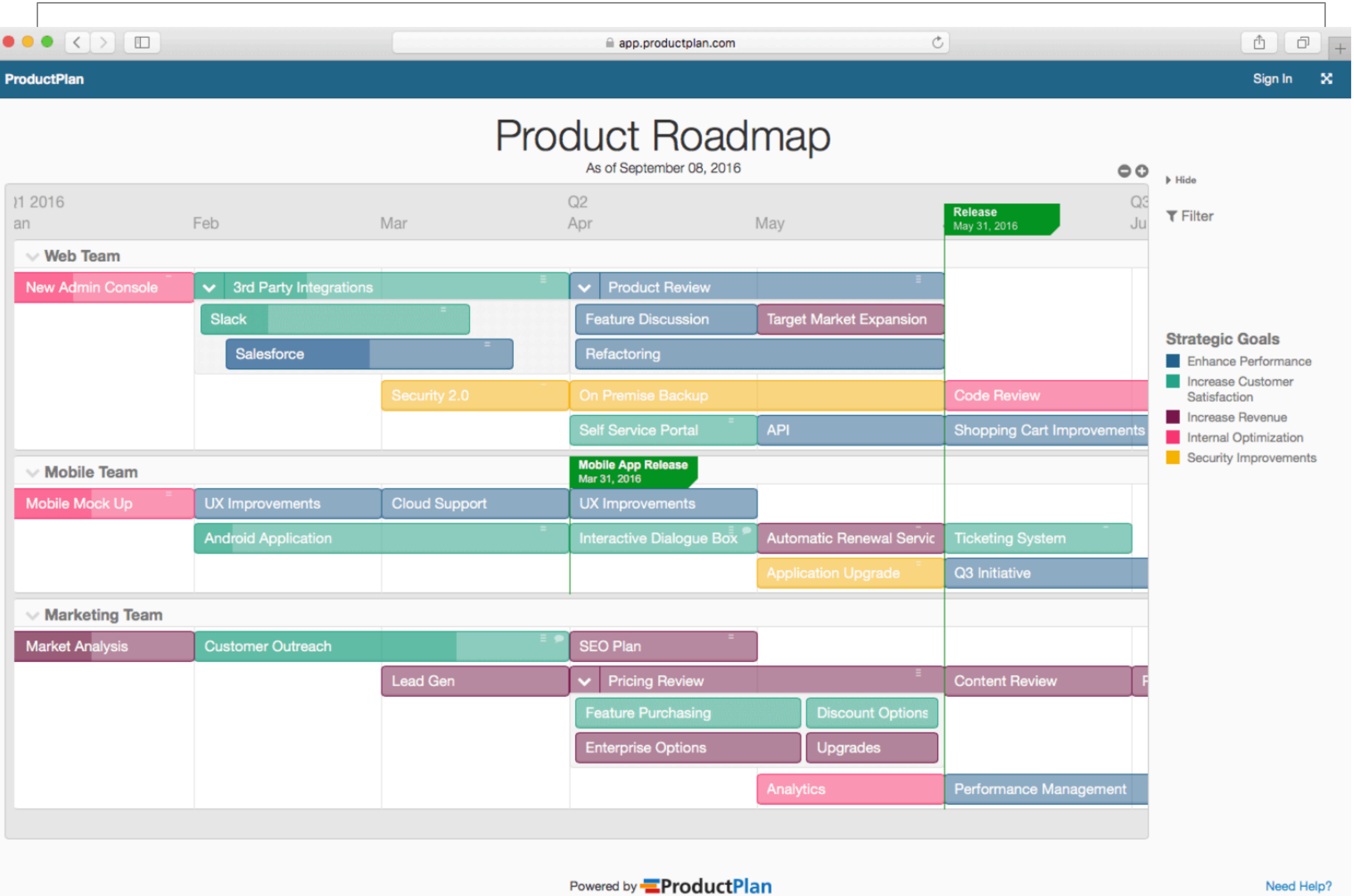
3. Release Planning

Create **iterations** and **iteration Plans** and **Product Roadmap**

To manage clients' needs

To align with Company Business Roadmap

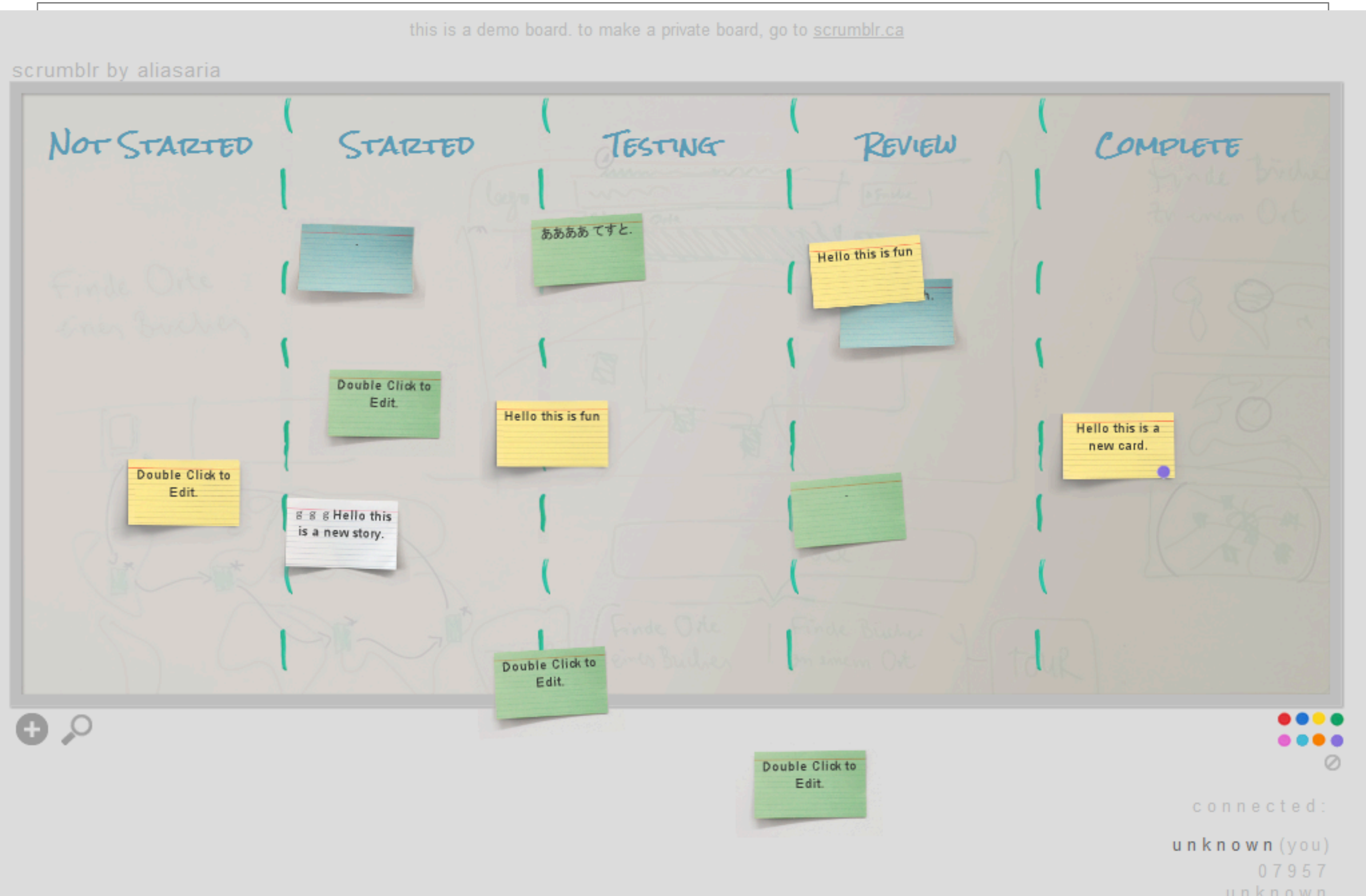
Typical to *Incremental Development*

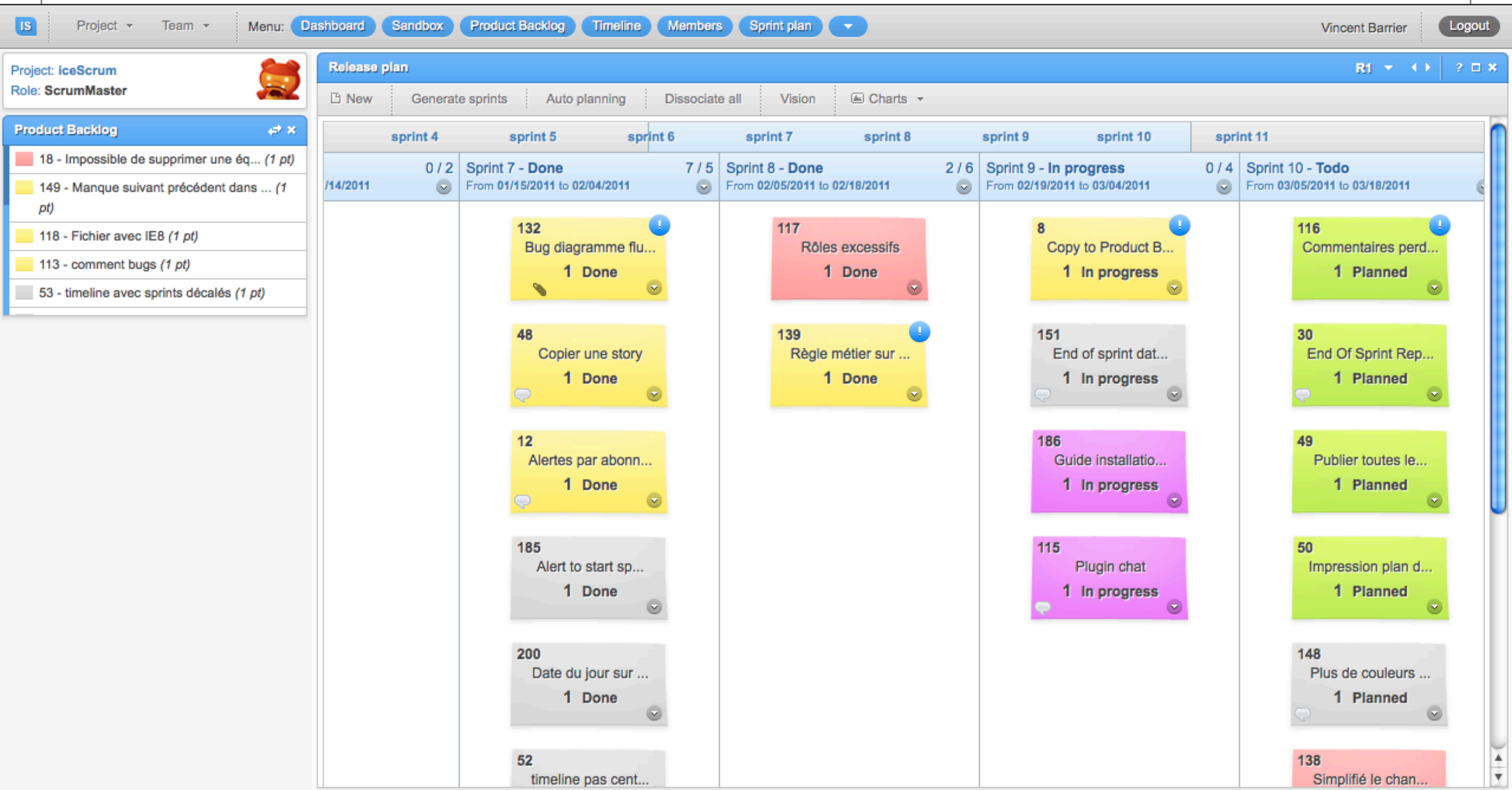

app.productplan.com
ProductPlan
Sign In
Product Roadmap
As of September 08, 2016
Hide
Filter
Q2
Q3
Feb
Mar
Apr
May
Release
May 31, 2016
Web Team
New Admin Console
3rd Party Integrations
Product Review
Slack
Feature Discussion
Target Market Expansion
Salesforce
Refactoring
Security 2.0
On Premise Backup
Code Review
Self Service Portal
API
Shopping Cart Improvements
Mobile Team
Mobile App Release
Mar 31, 2016
Mobile Mock Up
UX Improvements
Cloud Support
UX Improvements
Android Application
Interactive Dialogue Box
Automatic Renewal Servic
Ticketing System
Application Upgrade
Q3 Initiative
Marketing Team
Market Analysis
Customer Outreach
SEO Plan
Lead Gen
Pricing Review
Content Review
Feature Purchasing
Discount Options
Enterprise Options
Upgrades
Analytics
Performance Management
Strategic Goals
Enhance Performance
Increase Customer Satisfaction
Increase Revenue
Internal Optimization
Security Improvements
Powered by ProductPlan
Need Help?

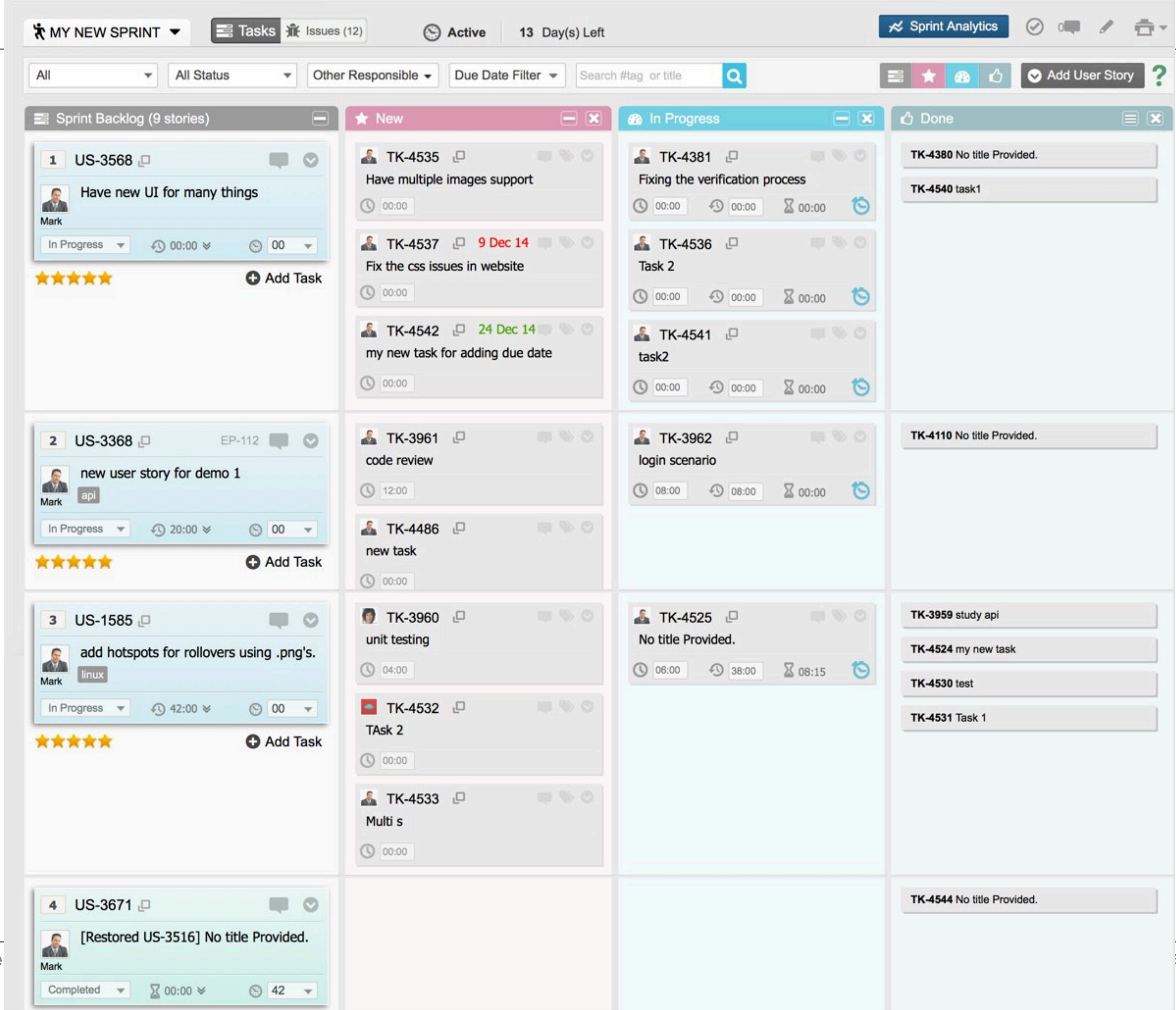
Agile Development has changed that

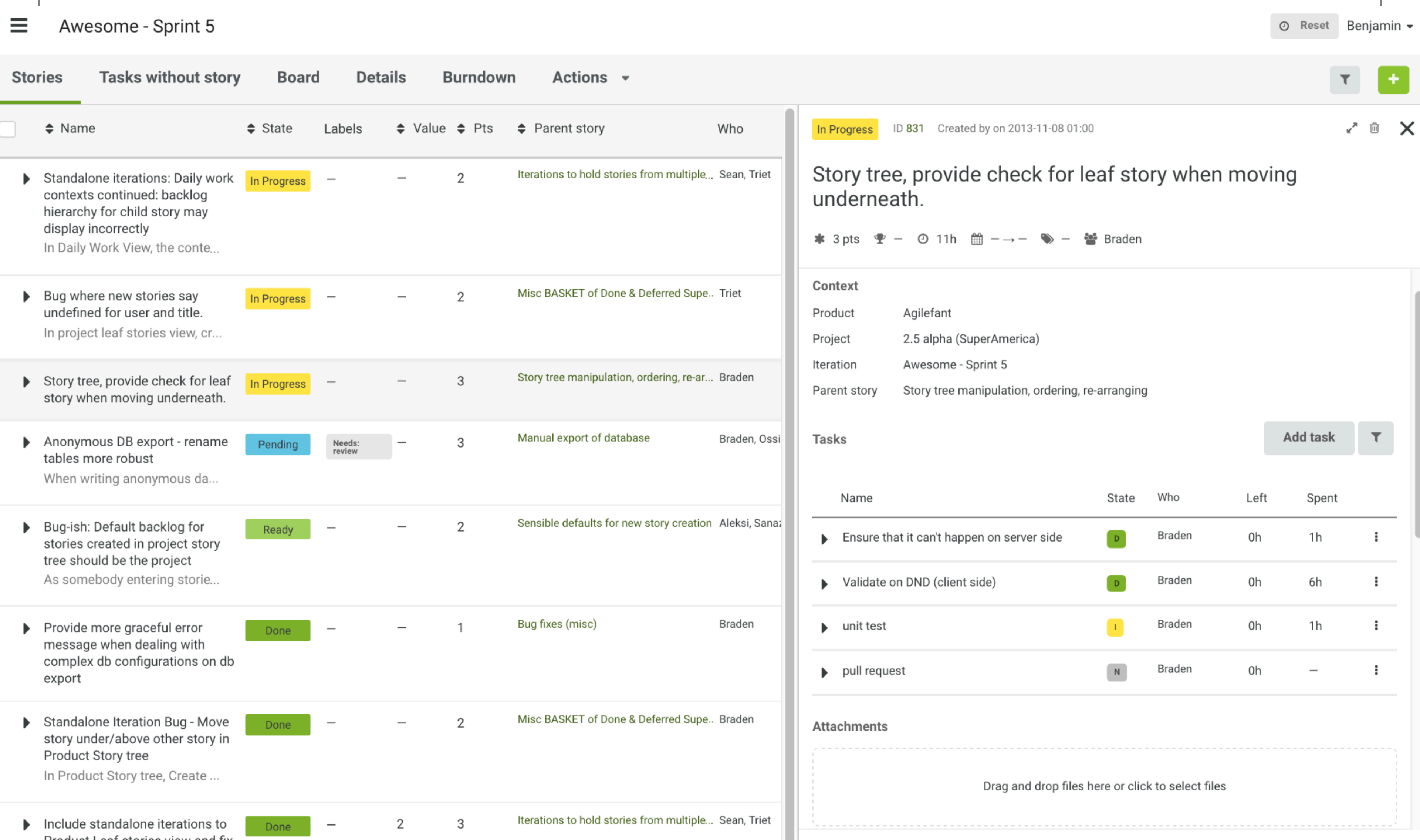
Iteration –> Sprints

Iteration Plan –> simply the Backlog!


this is a demo board. to make a private board, go to scrumblr.ca
scrumblr by aliasaria
Not Started
Started
Testing
Review
Complete
ああああ てすと.
Hello this is fun
Double Click to Edit.
Hello this is fun
Hello this is a new card.
Double Click to Edit.
g g g Hello this is a new story.
Double Click to Edit.
Double Click to Edit.
connected:
unknown (you)
07957
unknown

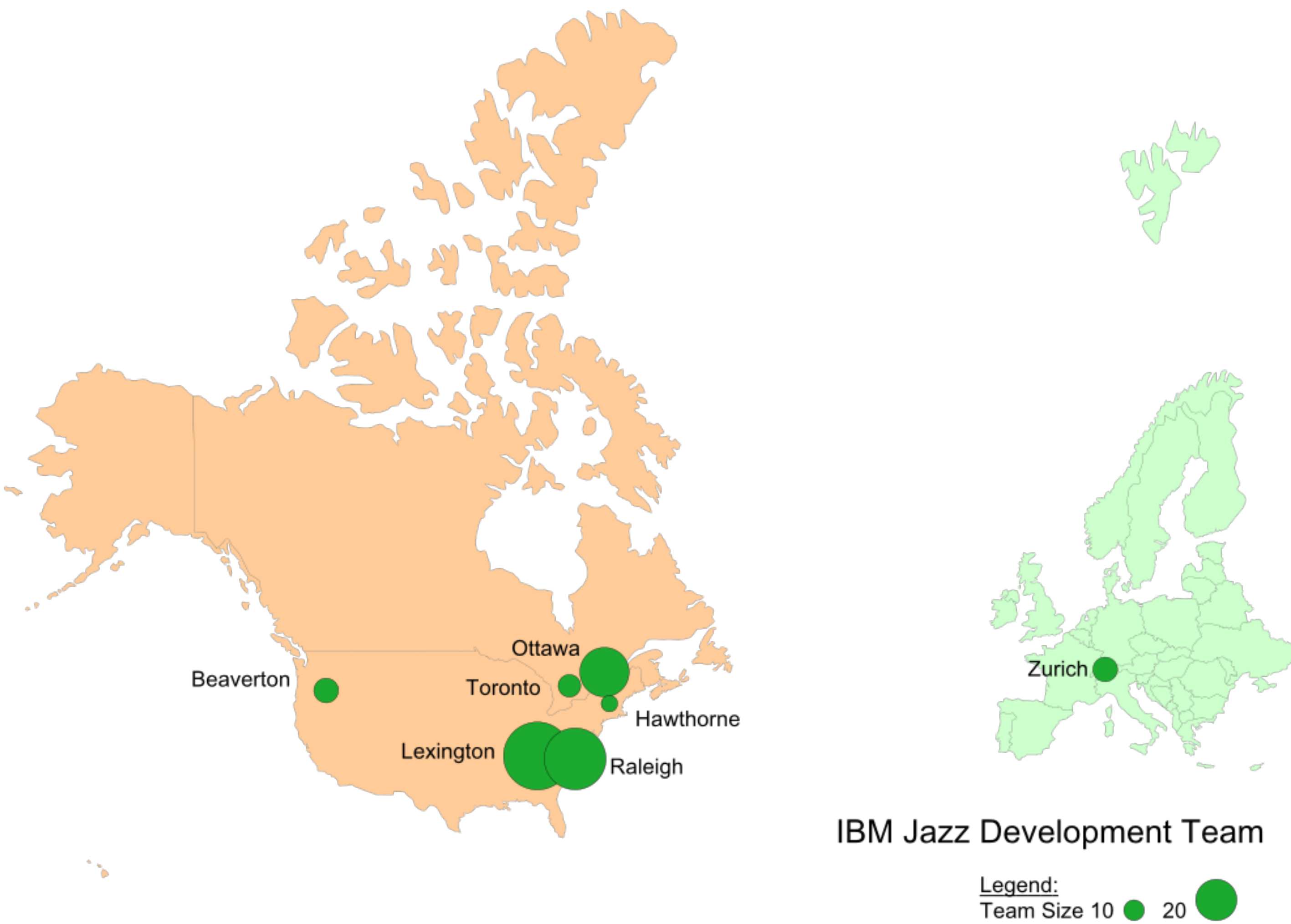

Project Team Menu: Dashboard Sandbox Product Backlog Timeline Members Sprint plan
Vincent Barrier
Logout
Project: iceScrum
Role: ScrumMaster
Product Backlog
18 - Impossible de supprimer une éq... (1 pt)
149 - Manque suivant précédent dans ... (1 pt)
118 - Fichier avec IE8 (1 pt)
113 - comment bugs (1 pt)
53 - timeline avec sprints décalés (1 pt)
Release plan
R1
New
Generate sprints
Auto planning
Dissociate all
Vision
Charts
sprint 4 sprint 5 sprint 6 sprint 7 sprint 8 sprint 9 sprint 10 sprint 11
Sprint 7 - Done 7 / 5
From 01/15/2011 to 02/04/2011
Sprint 8 - Done 2 / 6
From 02/05/2011 to 02/18/2011
Sprint 9 - In progress 0 / 4
From 02/19/2011 to 03/04/2011
Sprint 10 - Todo
From 03/05/2011 to 03/18/2011
132 Bug diagramme flu... 1 Done
48 Copier une story 1 Done
12 Alertes par abonn... 1 Done
185 Alert to start sp... 1 Done
200 Date du jour sur ... 1 Done
52 timeline pas cent...
117 Rôles excessifs 1 Done
139 Règle métier sur ... 1 Done
8 Copy to Product B... 1 In progress
151 End of sprint dat... 1 In progress
186 Guide installatio... 1 In progress
115 Plugin chat 1 In progress
116 Commentaires perd... 1 Planned
30 End Of Sprint Rep... 1 Planned
49 Publier toutes le... 1 Planned
50 Impression plan d... 1 Planned
148 Plus de couleurs ... 1 Planned
138 Simplifié le chan...

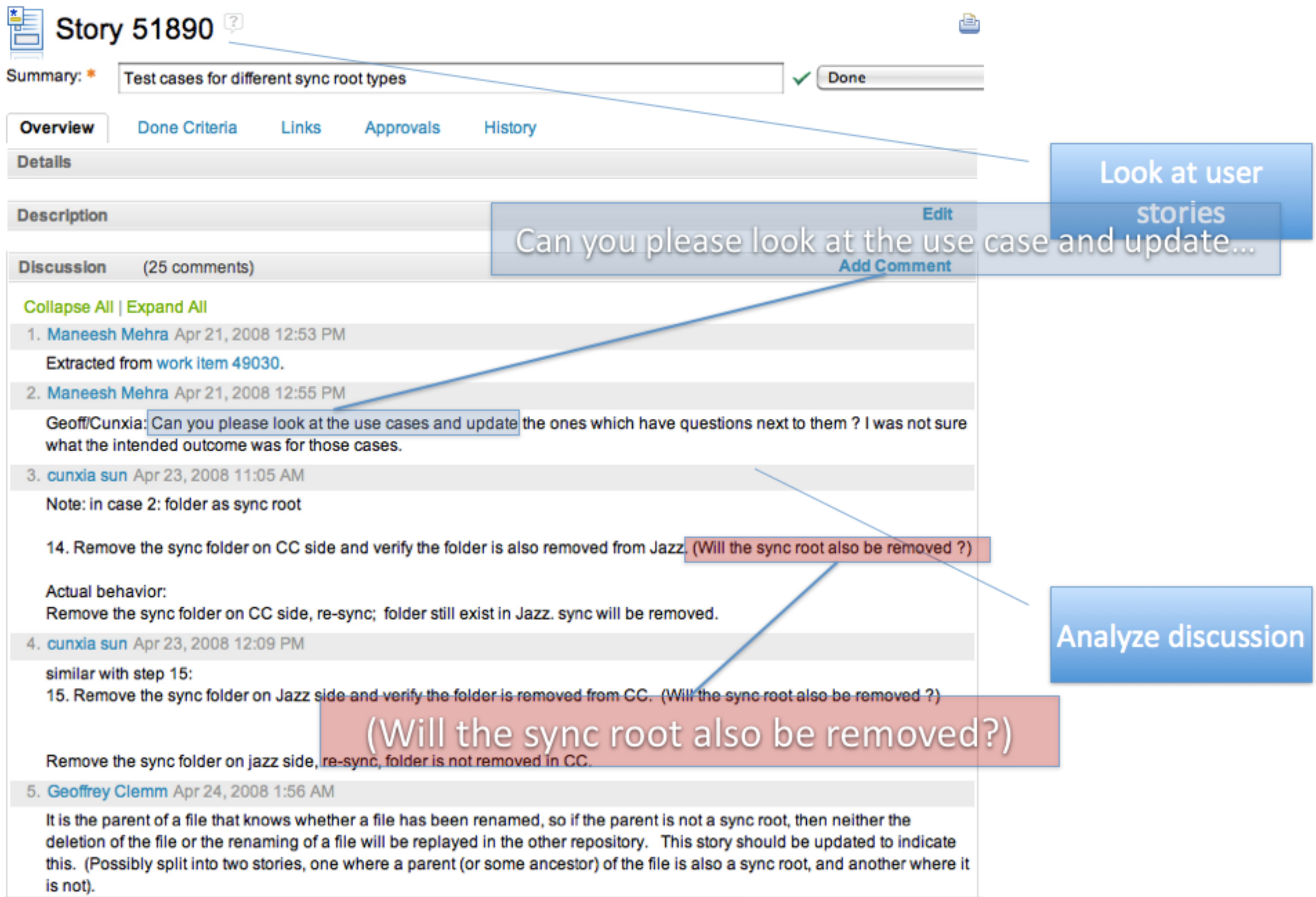

MY NEW SPRINT
Tasks
Issues (12)
Active
13 Day(s) Left
Sprint Analytics
All
All Status
Other Responsible
Due Date Filter
Search #tag or title
Add User Story
Sprint Backlog (9 stories)
New
In Progress
Done
1 US-3568
Have new UI for many things
Mark
In Progress 00:00 00
Add Task
TK-4535
Have multiple images support
00:00
TK-4537 9 Dec 14
Fix the css issues in website
00:00
TK-4542 24 Dec 14
my new task for adding due date
00:00
TK-4381
Fixing the verification process
00:00 00:00 00:00
TK-4536
Task 2
00:00 00:00 00:00
TK-4541
task2
00:00 00:00 00:00
TK-4380 No title Provided.
TK-4540 task1
2 US-3368 EP-112
new user story for demo 1
api
Mark
In Progress 20:00 00
Add Task
TK-3961
code review
12:00
TK-4486
new task
00:00
TK-3962
login scenario
08:00 08:00 00:00
TK-4110 No title Provided.
3 US-1585
add hotspots for rollovers using .png's.
linux
Mark
In Progress 42:00 00
Add Task
TK-3960
unit testing
04:00
TK-4532
TAsk 2
00:00
TK-4533
Multi s
00:00
TK-4525
No title Provided.
06:00 38:00 08:15
TK-3959 study api
TK-4524 my new task
TK-4530 test
TK-4531 Task 1
4 US-3671
[Restored US-3516] No title Provided.
Mark
Completed 00:00 42
TK-4544 No title Provided.



Awesome - Sprint 5
Reset
Benjamin
Stories
Tasks without story
Board
Details
Burndown
Actions
Name
State
Labels
Value
Pts
Parent story
Who
Standalone iterations: Daily work contexts continued: backlog hierarchy for child story may display incorrectly
In Daily Work View, the conte...
In Progress
2
Iterations to hold stories from multiple...
Sean, Triet
Bug where new stories say undefined for user and title.
In project leaf stories view, cr...
In Progress
2
Misc BASKET of Done & Deferred Supe..
Triet
Story tree, provide check for leaf story when moving underneath.
In Progress
3
Story tree manipulation, ordering, re-ar...
Braden
Anonymous DB export - rename tables more robust
When writing anonymous da...
Pending
Needs: review
3
Manual export of database
Braden, Ossi
Bug-ish: Default backlog for stories created in project story tree should be the project
As somebody entering storie...
Ready
2
Sensible defaults for new story creation
Aleksi, Sanaz
Provide more graceful error message when dealing with complex db configurations on db export
Done
1
Bug fixes (misc)
Braden
Standalone Iteration Bug - Move story under/above other story in Product Story tree
In Product Story tree, Create ...
Done
2
Misc BASKET of Done & Deferred Supe..
Braden
Include standalone iterations to Product Leaf stories view and fix
Done
2
3
Iterations to hold stories from multiple...
Sean, Triet
In Progress
ID 831
Created by on 2013-11-08 01:00
Story tree, provide check for leaf story when moving underneath.
3 pts
11h
Braden
Context
Product
Agilefant
Project
2.5 alpha (SuperAmerica)
Iteration
Awesome - Sprint 5
Parent story
Story tree manipulation, ordering, re-arranging
Tasks
Add task
Name
State
Who
Left
Spent
Ensure that it can't happen on server side
D
Braden
0h
1h
Validate on DND (client side)
D
Braden
0h
6h
unit test
I
Braden
0h
1h
pull request
N
Braden
0h
Attachments
Drag and drop files here or click to select files

Case studies of developer communications in large projects





Story 51890

Summary: * Test cases for different sync root types ✔ Done

Overview | Done Criteria | Links | Approvals | History

Details

Description Edit

Discussion (25 comments) Add Comment

Collapse All | Expand All

1. Maneesh Mehra Apr 21, 2008 12:53 PM

Extracted from work item 49030.

2. Maneesh Mehra Apr 21, 2008 12:55 PM

Geoff/Cunxia: Can you please look at the use cases and update the ones which have questions next to them ? I was not sure what the intended outcome was for those cases.

3. cunxia sun Apr 23, 2008 11:05 AM

Note: in case 2: folder as sync root

14. Remove the sync folder on CC side and verify the folder is also removed from Jazz. (Will the sync root also be removed ?)

Actual behavior:
Remove the sync folder on CC side, re-sync; folder still exist in Jazz. sync will be removed.

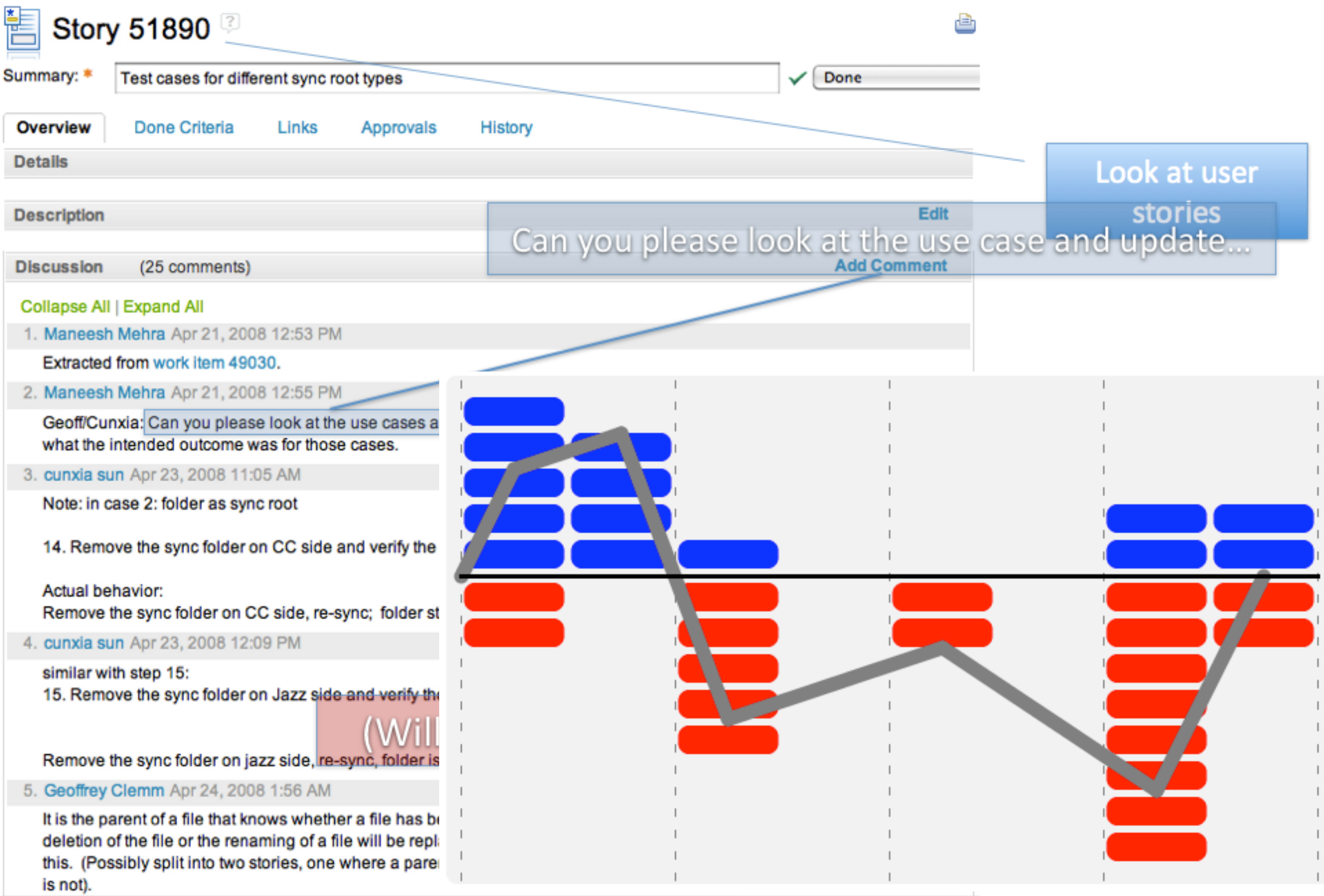
4. cunxia sun Apr 23, 2008 12:09 PM

similar with step 15:
15. Remove the sync folder on Jazz side and verify the folder is removed from CC. (Will the sync root also be removed ?)

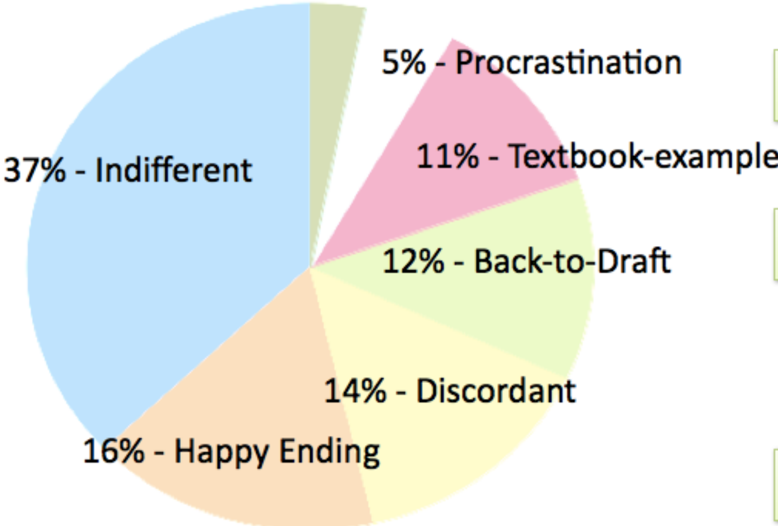
Remove the sync folder on jazz side, re-sync, folder is not removed in CC.

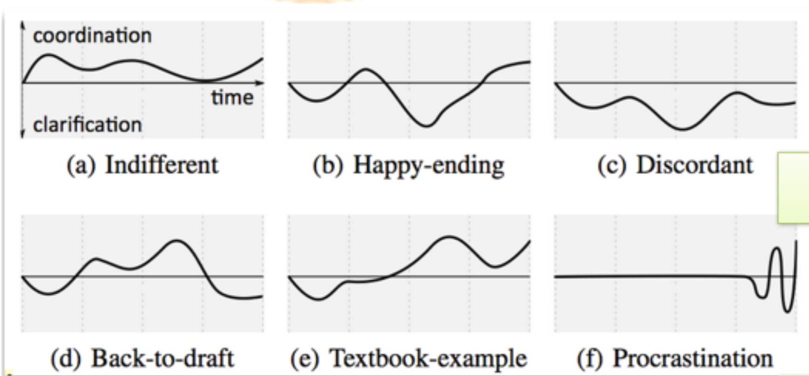
5. Geoffrey Clemm Apr 24, 2008 1:56 AM

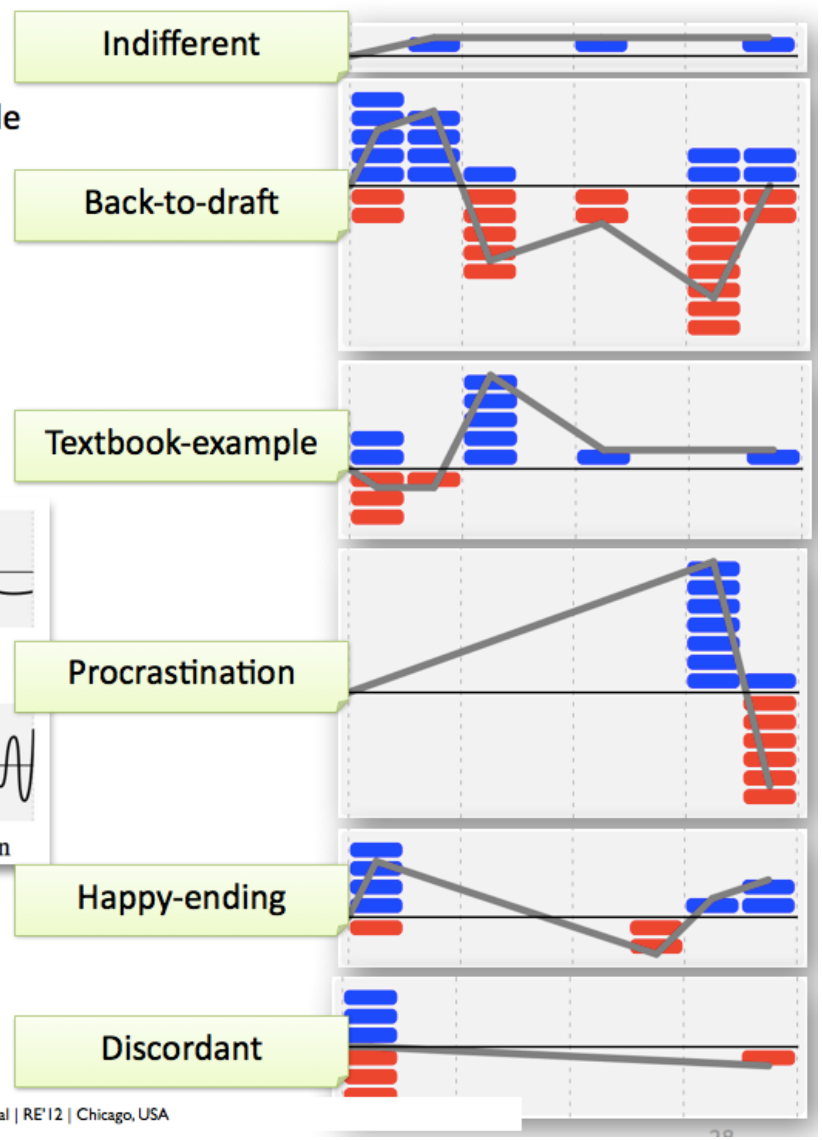
It is the parent of a file that knows whether a file has been renamed, so if the parent is not a sync root, then neither the deletion of the file or the renaming of a file will be replayed in the other repository. This story should be updated to indicate this. (Possibly split into two stories, one where a parent (or some ancestor) of the file is also a sync root, and another where it is not).



Detecting and Classifying Patterns of Requirements Clarifications | E. Kanuss, D. Damian, et. al | RE'12 | Chicago, USA







Detecting and Classifying Patterns of Requirements Clarifications | E. Kanuss, D. Damian, et. al | RE'12 | Chicago, USA

SOFTWARE ENGINEERING GLOBAL INTERACTION LAB

Coordination and Communication in SE

Software Ecosystems

Software Engineering Education

Applications of SE

Welcome to SEGAL

SEGAL is a research facility in the Computer Science Department of University of Victoria, BC. We carry out research to improve the collaboration of geographically distributed software development teams. Global software development is increasingly becoming common practice in the software industry. The ability to develop software at remote sites in projects allows organizations to ignore the geographic distance and benefit from access to a qualified resource pool and reduced development costs. However, large software projects in such large organizations or IT ecosystems involve complex interactions across organizational, functional as well as national, cultural and socio-economic boundaries, making their study important but difficult. In our research projects we employ a synergy of empirical methods, data mining and social network analysis techniques to understand these complex interactions as well as develop methods, processes and tools to improve the effectiveness of communication and coordination in large, distributed software projects. Check out a 2014 report on our Research Programme and the list of our recent publications for results of our work.

How to Reach Us

Recent Research News

16 MAY

MSc and PhD positions in SEGAL!

Our lab is seeking applications for MSc and Doctoral research positions in the area of software ecosystems. The research pertains to... Read More →

13 APR

Supporting the Adaptation of Contextual Requirements at Runtime

3. Requirements prioritization

Difficult task!

Different stakeholders may have different priorities

Organizations lack systematic data, metrics or techniques to help the prioritization process

Often carried out informally

Research shows that few companies know how to establish and communicate priorities

However important

Prioritizing requirements helps:

- Making acceptable tradeoffs among goals of quality, cost and time-to-market
- Project planning in allocating resources based on requirements importance to the project as a whole

A process of prioritizing requirements

Must be simple and fast, for industry adoption

Yield accurate and trustworthy results

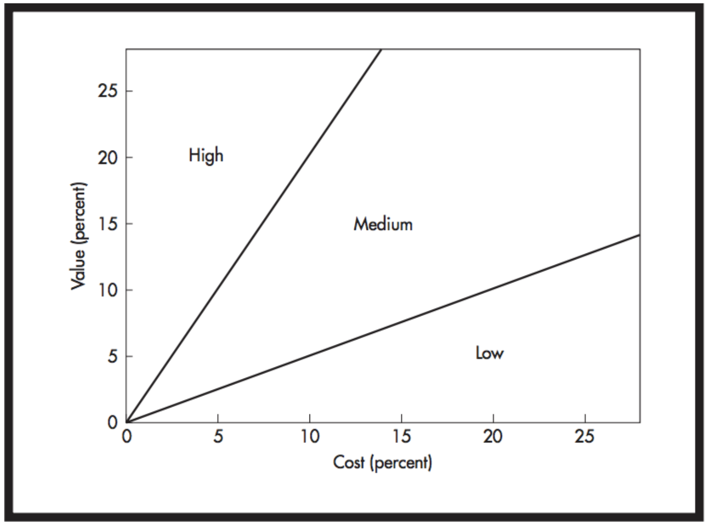
Should consider issues of:

- **Importance** of requirement to the **user/customer** (maximize)
- **Cost** of implementation (minimize)
- **Time-to-delivery** (minimize)

A cost-value approach

Calculate return on investment by:

- Assessing the **value** of each requirement
- Assessing the **cost** of each requirement
- Assess the **cost-value trade-off**



Difficulties:

- Hard to calculate absolute value/cost, relative values easier to obtain
- Interdependent requirements difficult to treat individually
- Inconsistencies or conflicts in priorities assigned by individual stakeholders

The Analytical Hierarchy Process (AHP)*

Steps (to prioritize n requirements):

1. Set up the n requirements in the rows and columns of a nxn matrix
2. Perform pairwise comparisons of all the requirements according to the criterion
3. Use averaging over normalized columns to estimate the eigenvalues of the matrix:
   - Calculate sum of the column = the sum of the n elements in each column
   - Divide each element in the matrix by the sum of the column the element is a member of, and calculate sums for each row
   - Normalize the sum of the rows (divide each row sum with the number of requirements)
   - The result == priority information for each requirement

* see example application and calculation in the PDF paper in the same connex Resource Folder

Pairwise comparison of requirements

Use a 1-9 scale with:

- $a_{ij} = 1$ if the two requirements are equal in importance
- $a_{ij} = 3$ if R_i is weakly more important than R_j
- $a_{ij} = 5$ if R_i is strongly more important than R_j
- $a_{ij} = 7$ if R_i is very strongly more important than R_j
- $a_{ij} = 9$ if R_i is absolutely more important than R_j

Example requirements prioritization techniques that leverage AHP

As developed by Karlsson and Ryan:

- Use the Analytic Hierarchy Process to assess relative values and costs of requirements
- Use cost-value diagrams to analyze and discuss candidate requirements

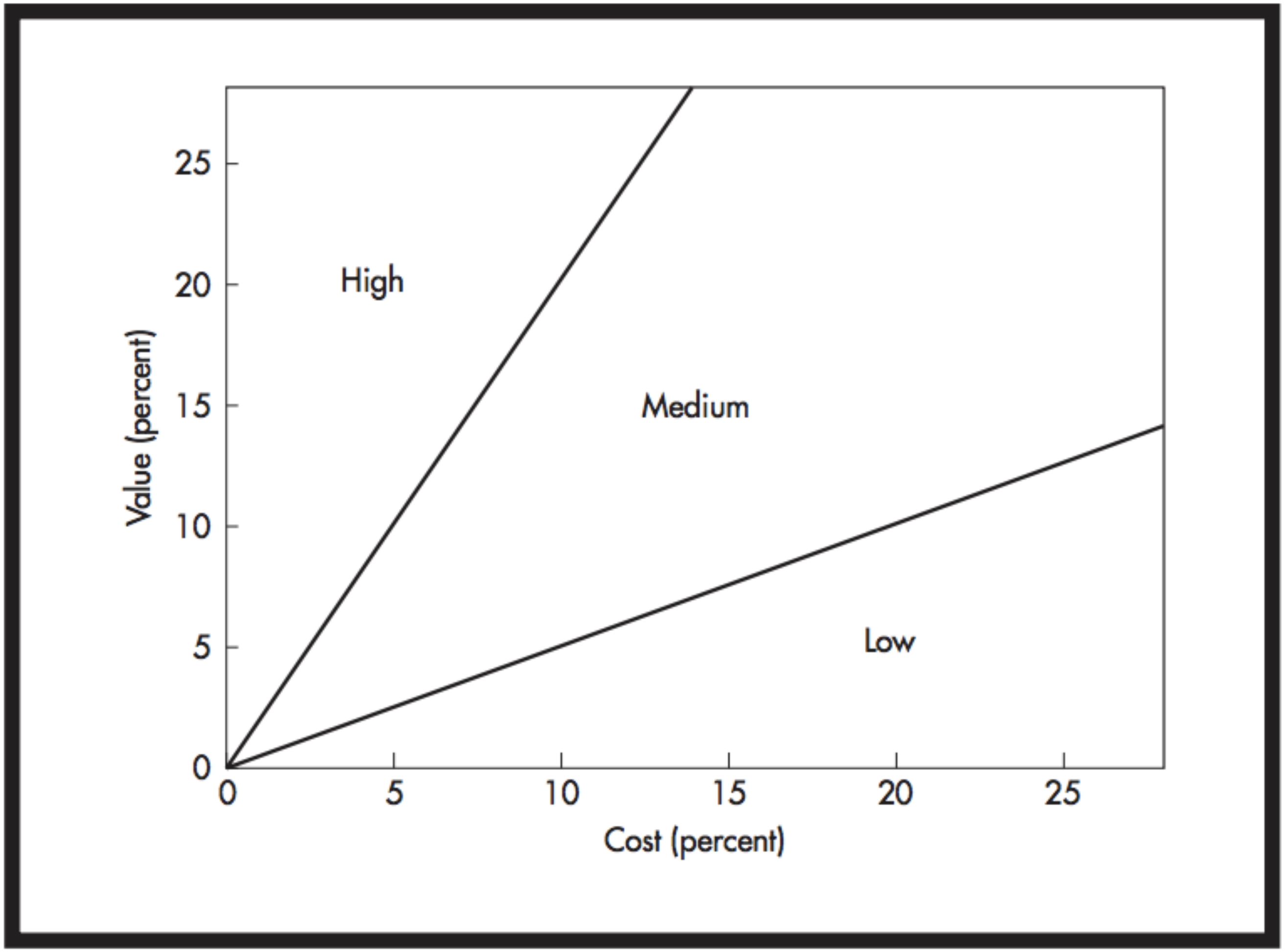
Useful to managers for requirements triage and release planning

Detailed practical steps

1. ***Requirements engineers* check requirements** for ambiguities, completeness, etc.
2. Apply AHP's pairwise comparison to **estimate the relative *value* of candidate requirements**
3. ***Software engineers*** use AHP's pairwise comparison to **estimate the *cost* of candidate requirements**
4. Plot these values on a **cost-value diagram**
5. Stakeholders use this map to **analyze and make trade-offs**

Detailed practical steps

1. ***Re*** ...uities,
2. Ap... ***value* of**
3. ***Sc*** ... **mate the**
4. Pl...
5. St... ***s***



Applicability of method

This cost-value technique has been applied successfully to industrial projects

Has some limitations in projects with:

- Large number of requirements, pairwise comparison can be tedious
- Many interdependencies between requirements
- Distributed stakeholders

Moscow Method

M - Must have: These are non-negotiable requirements that the project must deliver for it to be considered successful. Without these, the project's objectives cannot be met.
S - Should have: These are important requirements but not vital for launch. They are high-priority items that should be included in the delivery if it is possible to do so without impacting the must-have items.
C - Could have: These are desirable but not necessary items that could improve the user experience or customer satisfaction. They will be included if there is enough time and resources after delivering the must-have and should-have elements.
W - Won't have (this time): These are the lowest priority requirements, often referred to as "wish list" items. They are not planned for the current project scope but might be considered for the future. They are explicitly stated to manage stakeholders' expectations and to ensure that everyone is clear on what will not be delivered in the current project.

Moscow Method

Can each feature be broken down into smaller features?

Prioritize each feature in one of the four categories

Must Have	Should Have	Could Have	Won't Have

A method for ***distributed*** prioritization of requirements

When stakeholder groups geographically distributed

Each stakeholder can contribute with their particular priorities

The Product Strategy Team (PST) works with the Market operations (MO) at each customer site to iteratively prioritize features

A distributed prioritization process

Step 1. PST sends out a list of feature groups

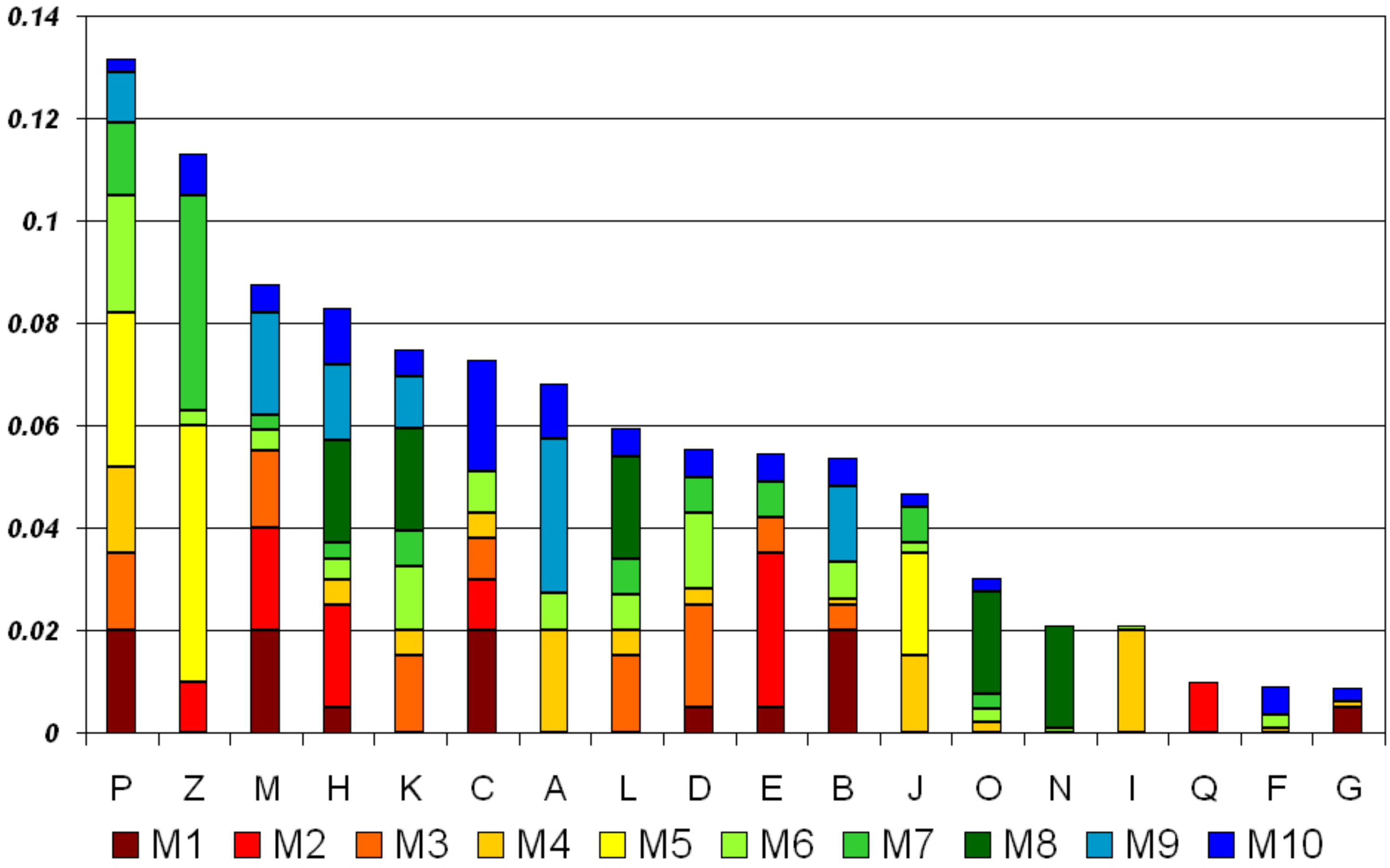
Step 2. Each MO makes a prioritization of each feature group

Step 3. based on info from step 2, PST makes a recommendation

Step 4. Stakeholder groups give feedback

Step 5. Iterate if necessary.

Visualization of prioritization data
Distribution chart with equal market influence

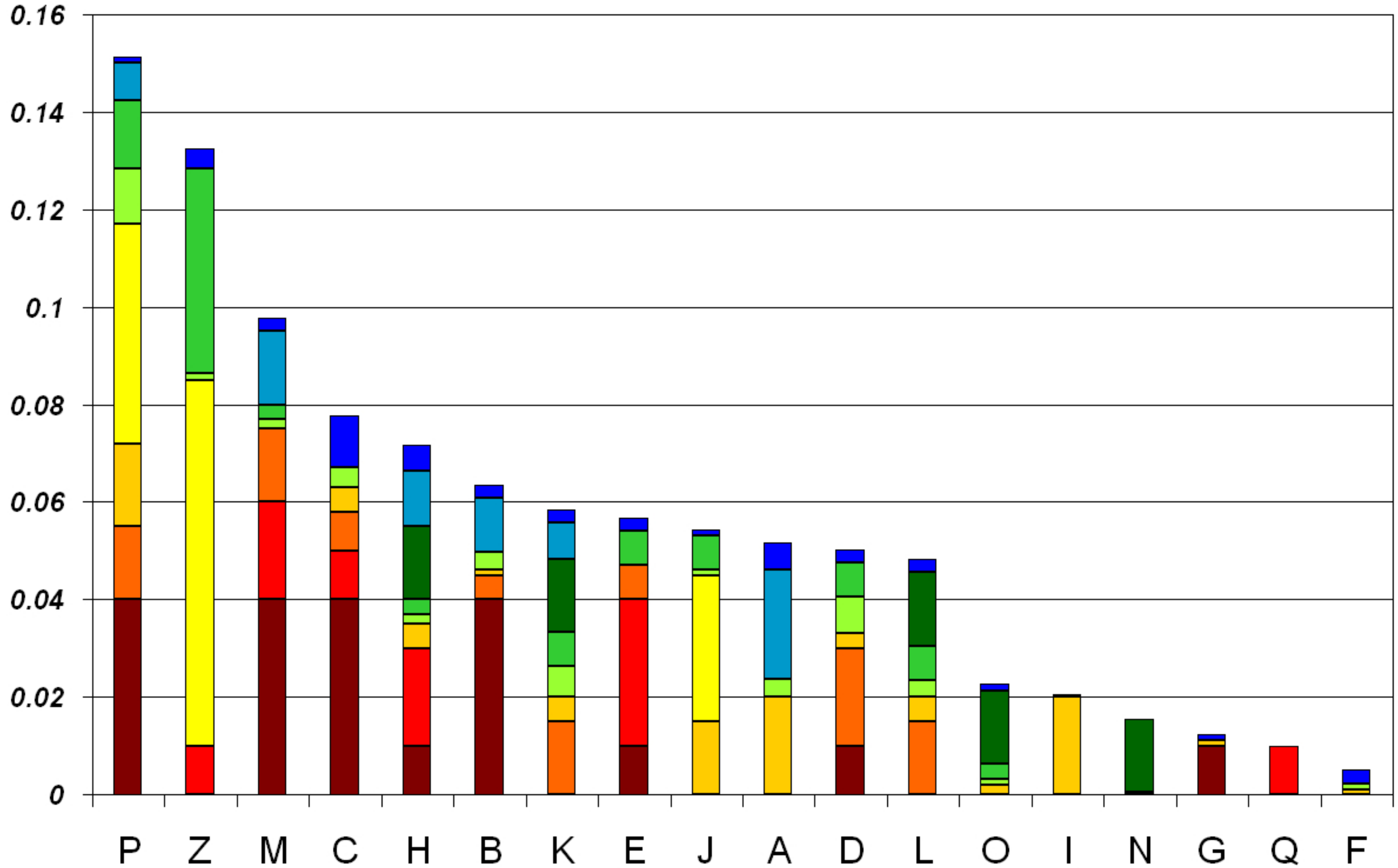


Each customer is unique

The process uses a weighting criteria to consider each individual stakeholder group by:

- Revenue last release
- Profit last release
- Number of sold licenses last release
- Predictions of the above criteria for the coming release
- Number of contracts lost to competitors
- Number of potential customer with nil licenses to date
- Size of total market segment

Visualization of prioritization data
Priority distribution chart with weighted market influence



In RD Final version

For each requirement, include Rationale for each feature, functional requirement, non functional requirement

Ensure that each has an Acceptance Test