Agile Methods

Andy Podgurski
EECS Department
Case Western Reserve University

Agile Methods

- Alternative to document-driven, heavyweight development processes
- Intended to better accommodate *frequent changes* in requirements
- A collection of different methods with *shared principles*
- Based on ideas of iterative, incremental, and evolutionary development
- However, agile methods entail *less* planning, analysis, and documentation

The “Agile Manifesto” [Beck et al]

- “We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
  - Individuals and interaction over process and tools,
  - Working software over comprehensive documentation,
  - Customer collaboration over contract negotiation,
  - Responding to change over following a plan.”
- That is, while there is a value in the items on the right, we value the items on the left more.”

Extreme Programming (XP) [Beck et al]

- Like Spiral Model, XP is intended to *resolve risks*, e.g.:
  - Misunderstanding of business needs
  - Schedule slippage
  - Unneeded features
  - Poor reliability
  - Poor maintainability
  - Staff turnover

XP Principles

- Rapid feedback
- Assume simplicity
- Incremental change
- Embracing change
- Quality work

XP Practices

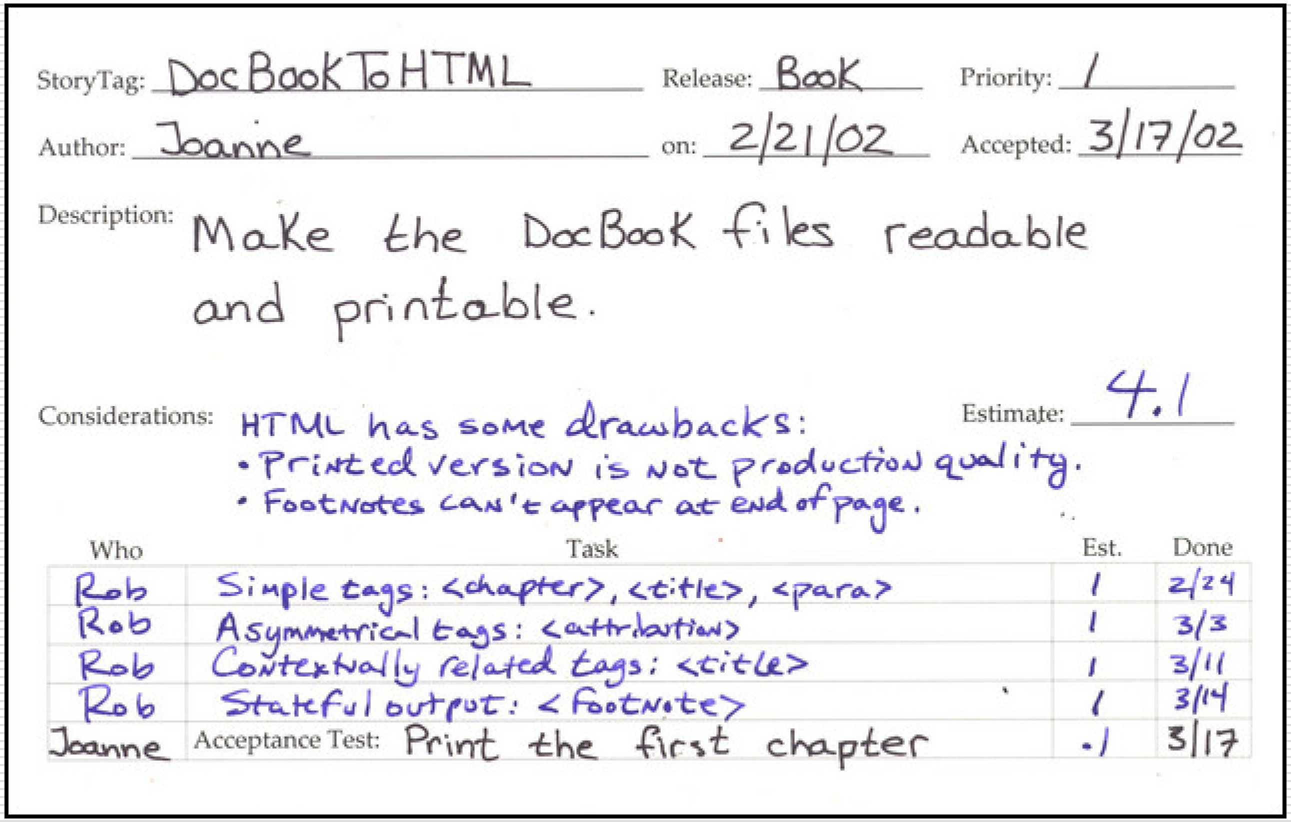
- *The Planning Game*:
  - Customers, managers, and developers meet to flesh out, estimate, and prioritize requirements for the next release.
  - The requirements are called "user stories" and are captured on "story cards" in a language understandable by all parties.
- *Metaphor*: Customers, managers, and developers construct a metaphor, or set of metaphors, after which to model the system, e.g.,
  - Desktop Metaphor
  - Spreadsheet Metaphor
  - Shopping Cart Metaphor
  - Auction Metaphor
  - Blackboard Metaphor

Example User Stories

- Students can purchase monthly parking passes online.
- Parking passes can be paid via credit cards.
- Parking passes can be paid via PayPal.
- Professors can input student marks.
- Students can obtain their current seminar schedule.
- Students can order official transcripts.
- Students can only enroll in seminars for which they have prerequisites.
- Transcripts will be available online via a standard browser.

From www.agilemodeling.com/artifacts/userStory.htm

Example User Story Card



StoryTag: DocBookToHTML Release: Book Priority: 1

Author: Joanne on: 2/21/02 Accepted: 3/17/02

Description: Make the DocBook files readable and printable.

Considerations: HTML has some drawbacks:
- Printed version is not production quality.
- Footnotes can't appear at end of page.

Estimate: 4.1

Who	Task	Est.	Done
Rob	Simple tags: <chapter>, <title>, <para>	1	2/24
Rob	Asymmetrical tags: <attribution>	1	3/3
Rob	Contextually related tags: <title>	1	3/11
Rob	Stateful output: <footnote>	1	3/14
Joanne	Acceptance Test: Print the first chapter	.1	3/17

From www.extremeperl.org/i/docbook-to-html.jpg

Detailing a User Story

- A user story must be fleshed out, e.g., during
  - Analysis and modeling with stakeholders
  - Iteration planning
  - Implementation

See www.agilemodeling.com/artifacts/userStory.htm

XP Practices (2)

- *Simple design*:
  - The system should be designed as simply as possible at any given moment.
  - Extra complexity is removed as soon as it is discovered.
- *Test-Driven Development*:
  - *Developers* write unit tests for their code before they write the code itself.
  - *Customers* write functional tests for each iteration and at the end of each iteration, all tests should run.
  - All code must pass all tests before it can be released.
- *Small releases*:
  - An initial version of the system is put into production after the first few iterations.
  - Subsequently, working versions are put into production anywhere from every few days to every few weeks

XP Practices (3)

- *Continuous integration*:
  - Developers integrate new code frequently.
    - Ideally the system is built after every commit.
  - All automated tests must *pass* after integration or the new code is *discarded* from the build.
- *Refactoring*: Programmers restructure the system to
  - Simplify
  - Remove duplication
  - Improve understandability
  - Add flexibility
  - See www.cs.unc.edu/~stotts/COMP723-s13/refactor/chap1

XP Practices (4)

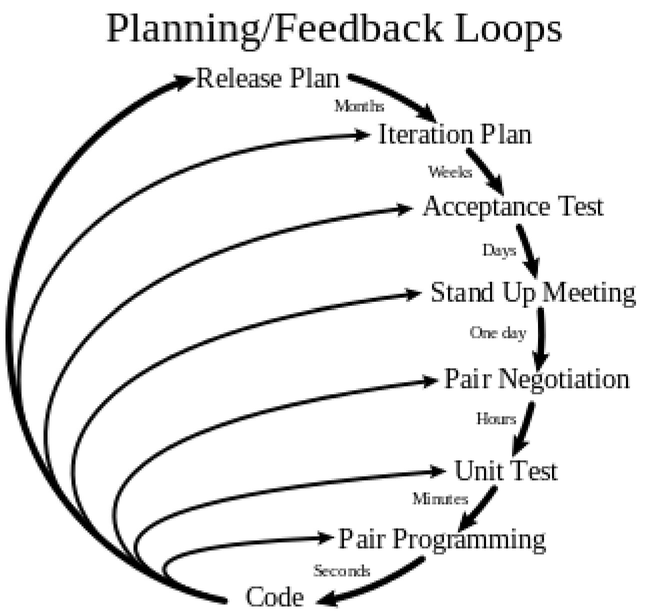
- *Pair programming*: All production code is written with two programmers at one machine.
- *Coding standards:* Programmers write all code in accordance with rules emphasizing communication through code.
- *Collective ownership*: The code is owned by all developers, and they may make changes anywhere in the code at anytime they feel necessary.

XP Practices (5)

- *40-hour week*: Requirements should be selected for each iteration so that developers do not need to put in overtime.

- *On-site customer*: A customer works with the development team to answer questions, perform acceptance tests, and ensure that development is progressing as expected.

Planning & Feedback in XP



[en.wikipedia.org/wiki/File:XP-feedback.gif]

Question

- Do you see any problems with XP?

Objections to XP

- You need to do *all* of XP or *none* at all.
- XP is aimed at customers that don’t know what they want.
- *Up-front analysis* using mockups, storyboards, prototypes, and use cases is *less risky* than XP.
- Constant *refactoring* entails *high overhead* and tends to introduce *bugs*.
- XP is over-reliant on testing.
- Many programmers *dislike* pair programming.
- The on-site customer representative is likely to be *inexperienced* and may be hard to deal with.
- XP essentially requires a customer to sign an *optional-scope contract* specifying a fixed amount of development time for a fixed price, without any commitment as to what is actually delivered.
- XP makes it hard to develop good early estimates of the work effort and project cost

Scrum

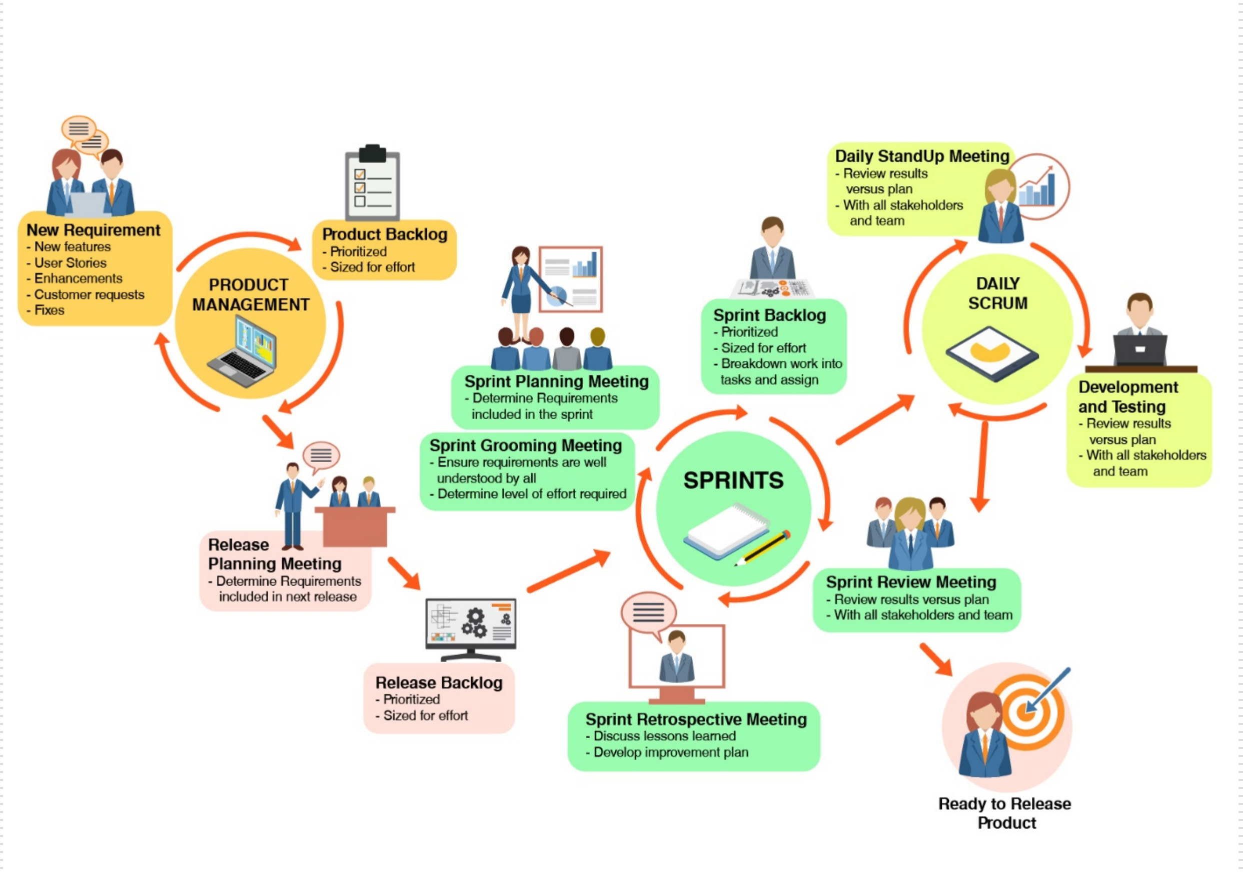
- Due to
  - Takeuchi and Nonaka Ikujiro
  - Schwaber and Sutherland
- Based on iterative and incremental development
- Somewhat more conventional than XP
- Projects split into iterations called *sprints*
- A sprint typically takes *1-4 weeks*
  - The interval is based on product complexity, risk assessment, and degree of oversight desired

Scrum Concepts & Events

- *A Product Backlog* of prioritized work to be done:
  - Bug fixes, enhancement requests, competitive product functionality, technology upgrades
- Completion of a fixed set of backlog items in a series of short iterations or *sprints*
- A brief daily meeting or *scrum*, at which progress is explained, upcoming work is described and impediments are raised
- A brief *Sprint Planning Session* in which the *Sprint Backlog* will be defined.
- A brief *Sprint Retrospective*, at which all team members reflect about the past sprint

From en.wikipedia.org/wiki/Scrum_(development)

Scrum Lifecycle


New Requirement
- New features
- User Stories
- Enhancements
- Customer requests
- Fixes
PRODUCT MANAGEMENT
Product Backlog
- Prioritized
- Sized for effort
Release Planning Meeting
- Determine Requirements included in next release
Release Backlog
- Prioritized
- Sized for effort
Sprint Planning Meeting
- Determine Requirements included in the sprint
Sprint Grooming Meeting
- Ensure requirements are well understood by all
- Determine level of effort required
Sprint Backlog
- Prioritized
- Sized for effort
- Breakdown work into tasks and assign
SPRINTS
Sprint Retrospective Meeting
- Discuss lessons learned
- Develop improvement plan
Daily StandUp Meeting
- Review results versus plan
- With all stakeholders and team
DAILY SCRUM
Development and Testing
- Review results versus plan
- With all stakeholders and team
Sprint Review Meeting
- Review results versus plan
- With all stakeholders and team
Ready to Release Product

From xbosoft.com/agile-test-automation-where-in-the-agile-lifecycle/

Scrum Roles

- *Product owner*
  - Maximizes product value
  - Manages product backlog
- *Development team*
  - Create and deliver product increment
  - Self organizing and cross-functional
- *Scrum Master*
  - Ensures Scrum process is followed
  - Removes impediments
- Scrum teams are *“self-organizing and cross-functional”*

Scrum Planning Phase

- Development of a *comprehensive backlog list*
- Definition of the delivery date and functionality of one or more releases
- Mapping of backlog items to *product packets* (components) that must be changed in the selected release
- Definition of project team(s) for the building of the new release
- Assessment of *risk* and appropriate risk controls
- Review and possible adjustment of backlog items and packets.
- Selection and validation of *development tools and infrastructure*
- Estimation of *release cost*, including development, collateral material, marketing, training, and rollout
- Verification of *management approval* and funding

Scrum Architecture Phase

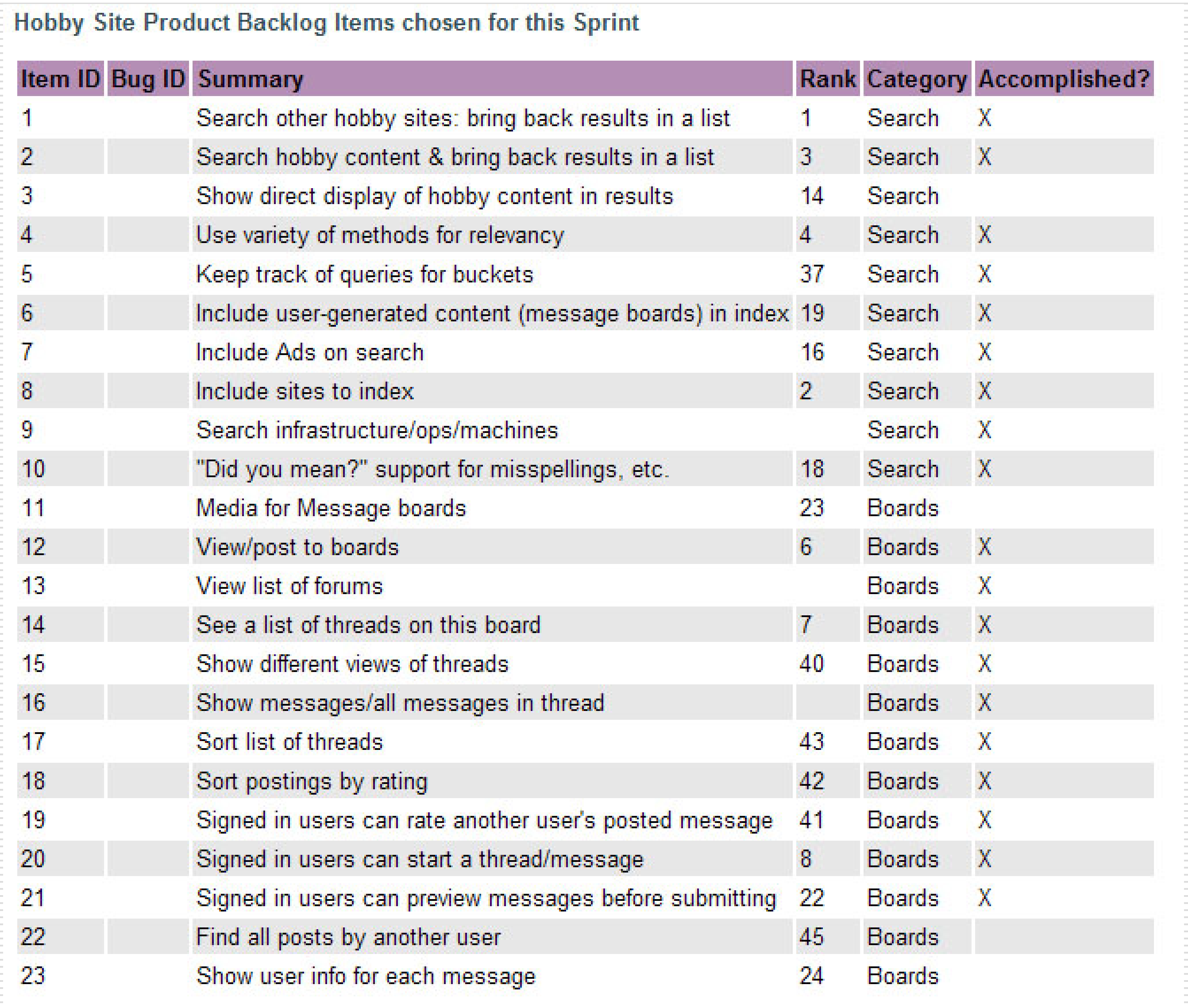
- *Review* assigned backlog items.
- *Identify changes* necessary to implement backlog items.
- *Refine* the system architecture to support the new context and requirements.
- Identify any *problems* or *issues* in developing or implementing the changes.
- *Design review meeting*, with each team presenting approach and changes to implement each backlog item.

The Sprint

- Limited to one month
- Consists of *Sprint Planning*, *Daily Scrums*, *Development*, *Sprint Review*, and *Sprint Retrospective*
- Has definition of what is to be built, a design, and a flexible implementation plan

Sprint Planning

- Done by entire Scrum team
- At most *8 hours* for one sprint
- Input: *Product Backlog*
- Answers two questions:
  - What can be delivered in this increment?
  - How will the work be achieved?
    - Involves *design*, *work scheduling*
- Produces *Sprint Goal* and *Sprint Backlog*

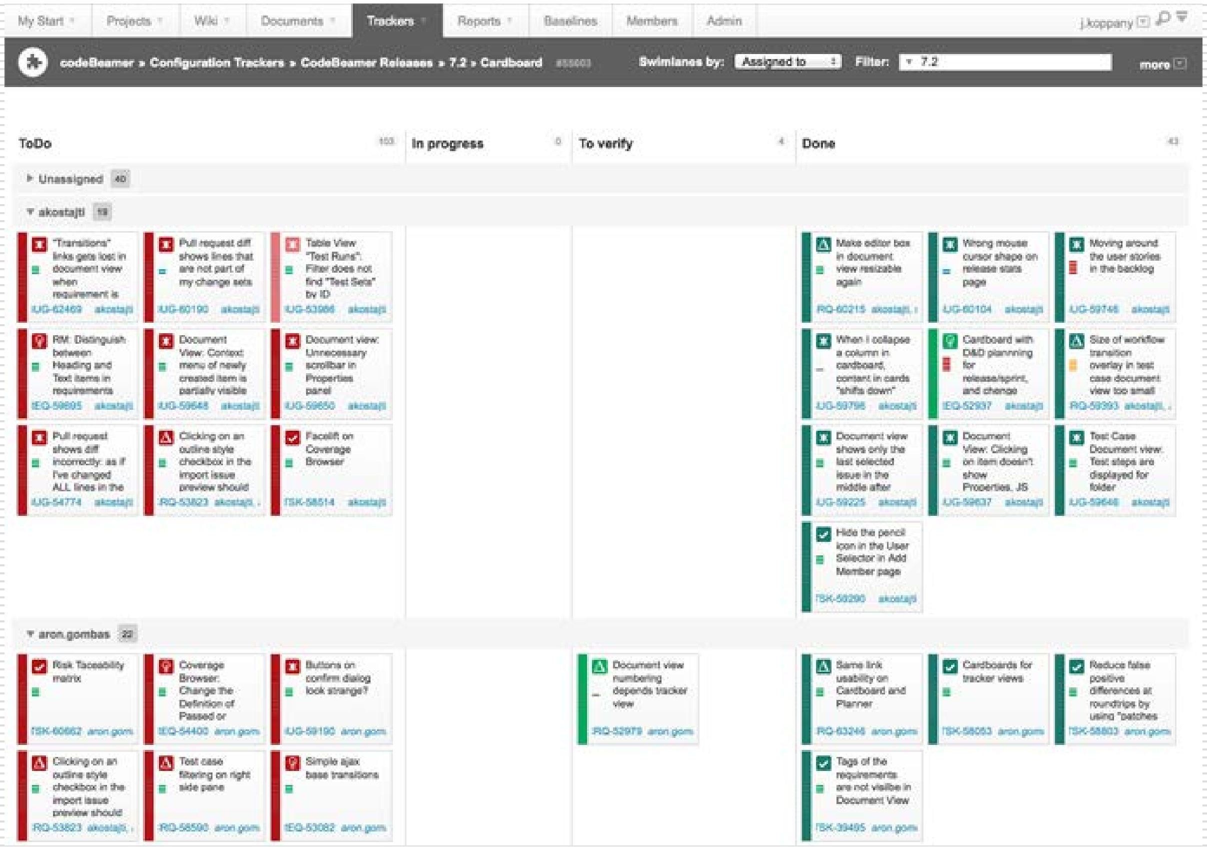


Hobby Site Product Backlog Items chosen for this Sprint

Item ID	Bug ID	Summary	Rank	Category	Accomplished?
1		Search other hobby sites: bring back results in a list	1	Search	X
2		Search hobby content & bring back results in a list	3	Search	X
3		Show direct display of hobby content in results	14	Search	
4		Use variety of methods for relevancy	4	Search	X
5		Keep track of queries for buckets	37	Search	X
6		Include user-generated content (message boards) in index	19	Search	X
7		Include Ads on search	16	Search	X
8		Include sites to index	2	Search	X
9		Search infrastructure/ops/machines		Search	X
10		"Did you mean?" support for misspellings, etc.	18	Search	X
11		Media for Message boards	23	Boards	
12		View/post to boards	6	Boards	X
13		View list of forums		Boards	X
14		See a list of threads on this board	7	Boards	X
15		Show different views of threads	40	Boards	X
16		Show messages/all messages in thread		Boards	X
17		Sort list of threads	43	Boards	X
18		Sort postings by rating	42	Boards	X
19		Signed in users can rate another user's posted message	41	Boards	X
20		Signed in users can start a thread/message	8	Boards	X
21		Signed in users can preview messages before submitting	22	Boards	X
22		Find all posts by another user	45	Boards	
23		Show user info for each message	24	Boards	

From www.scrumalliance.org/system/resource_files/0000/0100/hobby_backlog.jpg

“Kanban”-Style Backlog


My Start
Projects
Wiki
Documents
Trackers
Reports
Baselines
Members
Admin
j.koppany
codeBeamer » Configuration Trackers » CodeBeamer Releases » 7.2 » Cardboard
Swimlanes by: Assigned to
Filter: 7.2
more

ToDo 103 | In progress 0 | To verify 4 | Done 43

▸ Unassigned 40

▾ akostajti 19

ToDo:
- "Transitions" links gets lost in document view when requirement is — IUG-62469 akostajti
- Pull request diff shows lines that are not part of my change sets — IUG-60190 akostajti
- Table View "Test Runs": Filter does not find "Test Sets" by ID — IUG-53986 akostajti
- RM: Distinguish between Heading and Text items in requirements — IEQ-59695 akostajti
- Document View: Context menu of newly created item is partially visible — IUG-59648 akostajti
- Document view: Unnecessary scrollbar in Properties panel — IUG-59650 akostajti
- Pull request shows diff incorrectly: as if I've changed ALL lines in the — IUG-54774 akostajti
- Clicking on an outline style checkbox in the import issue preview should — RQ-53823 akostajti, .
- Facelift on Coverage Browser — TSK-58514 akostajti

Done:
- Make editor box in document view resizable again — RQ-60215 akostajti, .
- Wrong mouse cursor shape on release stats page — IUG-60104 akostajti
- Moving around the user stories in the backlog — IUG-59746 akostajti
- When I collapse a column in cardboard, content in cards "shifts down" — IUG-59798 akostajti
- Cardboard with D&D planning for release/sprint, and change — IEQ-52937 akostajti
- Size of workflow transition overlay in test case document view too small — RQ-59393 akostajti, .
- Document view shows only the last selected issue in the middle after — IUG-59225 akostajti
- Document View: Clicking on item doesn't show Properties, JS — IUG-59637 akostajti
- Test Case Document view: Test steps are displayed for folder — IUG-59646 akostajti
- Hide the pencil icon in the User Selector in Add Member page — TSK-59290 akostajti

▾ aron.gombas 22

ToDo:
- Risk Taceability matrix — TSK-60662 aron.gom.
- Coverage Browser: Change the Definition of Passed or — IEQ-54400 aron.gom.
- Buttons on confirm dialog look strange? — IUG-59190 aron.gom.
- Clicking on an outline style checkbox in the import issue preview should — RQ-53823 akostajti, .
- Test case filtering on right side pane — RQ-58590 aron.gom.
- Simple ajax base transitions — IEQ-53082 aron.gom.

To verify:
- Document view numbering depends tracker view — RQ-52979 aron.gom.

Done:
- Same link usability on Cardboard and Planner — RQ-63246 aron.gom.
- Cardboards for tracker views — TSK-58053 aron.gom.
- Reduce false positive differences at roundtrips by using "batches — TSK-58803 aron.gom.
- Tags of the requirements are not visible in Document View — TSK-39495 aron.gom.

www.intland.com/wp-content/uploads/2014/03/why-to-use-kanban-in-software-development.png

Daily Scrum

- *15-minute* event for Dev Team to
  - *Synchronize* activities
  - Create *plan* for next 24 hours
- Dev Team members explain:
  - What I did yesterday toward Sprint Goal
  - What I will do today
  - Impediments encountered/foreseen
- Dev Team members may have more *detailed* discussions afterward

Sprint Review

- *4-hour meeting*
- *Reviews* latest increment and *adapts* Product Backlog
- Attended by *Scrum Team* and *key stakeholders*
- Dev Team demonstrates work done and answers questions
- Review of timeline, budget, potential capabilities, marketplace for next release
- Group collaborates on what to do next to optimize value
- Produces *revised Product Backlog*

Sprint Retrospective

- Opportunity for Scrum Team to *inspect itself* and *plan improvements*
- 3-hour meeting
- Occurs after Sprint Review and prior to next Sprint Planning
- Considers people, relationships, process, and tools

Scrum Closure Phase

- Occurs when management feels time, competition, requirements, cost, and quality call for a *new release*
- Prepares the developed product for general release
- Tasks include: integration, system test, user documentation, preparation of training and marketing material

Sources (1)

- P. Abrahamsson, J. Warsta, M. Siponen, and J. Ronkainen. New directions on agile methods: a comparative analysis. 25th international Conference on Software Engineering (Portland, Oregon, May 2003).
- Kent Beck. Extreme Programming Explained: Embrace Change, Addison Wesley, 1999.
- D. Cohen, M. Lindvall, and P. Costa. An introduction to agile methods. Advances in Computers, No. 62, 2004.
- Scrum Guide, www.scrumalliance.org/why-scrum/scrum-guide.
- M. Stevens. The case against extreme programming. www.softwarereality.com/lifecycle/xp/case_against_xp.jsp

Sources (2)

- B. Weinstein. Eight reasons why extreme programming won't work in your shop. articles.techrepublic.com.com/5100-22-1046490.html.