Polarion Software®

E-BOOK

Polarion goes SCRUM

Europe, Middle-East, Africa: Polarion Software GmbH
Lautlinger Weg 3 — 70567 Stuttgart, GERMANY
Tel +49 711 489 9969 - 0
Fax +49 711 489 9969 - 20
www.polarion.com - info@polarion.com

Americas & Asia-Pacific: Polarion Software, Inc.
406 Tideway Dr. Alameda, CA 94501, USA
Tel +1 877 572 4005 (Toll free)
Fax +1 510 814 9983
www.polarion.com - info@polarion.com

Why Scrum?

Today many software companies are switching to Agile processes and, particularly to Scrum. Polarion Software also adopted Scrum for several reasons, some specific to the company (size, area of business, customers, etc.), and others quite general. This paper will focus on the major factors that influenced us.

Perhaps the most important, reason was transparency. Before Scrum, customers would want something, Product Management then defined requirements, the development team committed to fulfilling them, but the end results were not always as expected. Delivery sometimes slipped, and if a release was rescheduled, the risk was unclear and nobody could tell if new date was really realistic. Also, we needed to be able to respond quickly to changes in market conditions and business strategy.

Scrum helps us understand what we need to do to build quality software faster. We switched from defined and predictive development to an empirical iterative incremental model - exactly what Scrum is about.

Scrum's Core Values

Let's quickly review some of the important values of Scrum:

- **Empiricism** – facilitates management, development and deployment of complex products
- **Inspections and Adaptations**, allowing people to check and reach goals
- **Full Transparency**: people know the exact state of the product. No guessing or relying on statements like "we're right on time".
- **Iterative development** generates visible Increments of functionality. Progress is measured not by time or money spent, but by concrete results.

- **Self-organization** – people want to do their best, and will consistently achieve this when there is room for them to work in whatever way is most efficient for them, rather than according to some dictum. Team Integrity is raised, enhancing productivity and motivation.
- **Delivery** – many great projects with very capable teams have failed to deliver anything. Scrum helps teams minimize this risks along the way. If a project is in fact going to fail, it's better to know this as soon as possible, kill it earlier, and cut the losses.

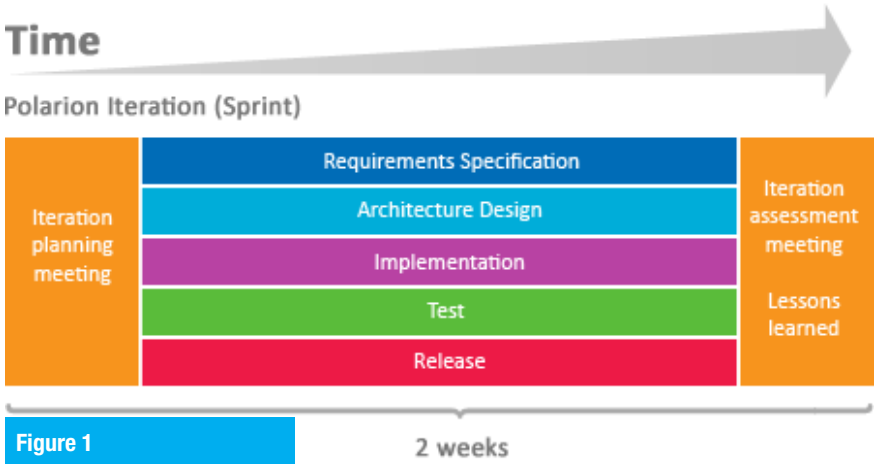

Figure 1

Iterative incremental Development

Unlike the old "waterfall" approach, Scrum recommends a highly-adaptive way of development with short iterations producing fully tangible results. Major benefits Polarion has realized from Scrum development include:

- Shorter time to release to the market
- Transparency to management and customers
- Faster reaction to market needs and customer confidence in Polarion's development
- Simpler synchronization of distributed teams
- Easier releases – smaller stabilization sprints, less things to test
- Faster feedback from the field
- Flexibility in prioritization, risk reduction

Also, some activities may be done in parallel. Specification of one feature for the next iteration may happen in parallel with implementation of another feature already specified in a previous iteration.

We have very short iterations of 2 weeks, with an iteration **Planning Meeting** at the beginning, and an iteration **Assessment Meeting** the end of each iteration. This has proven optimal for our several teams of 3 to 10 team members.

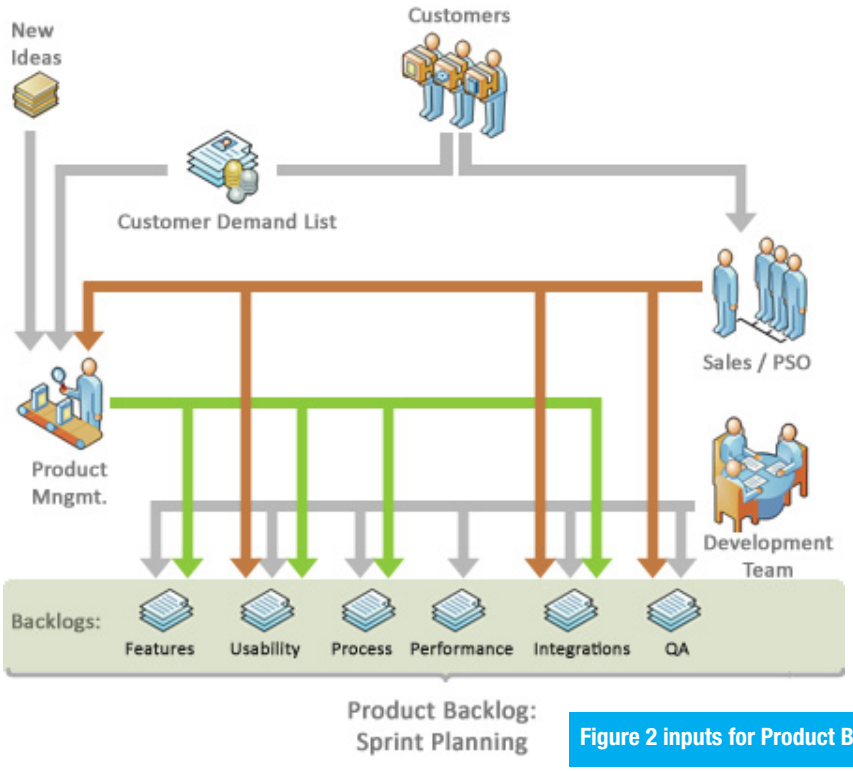

Figure 2 inputs for Product Backlog

Polarion ALM in Scrum Process

The remainder of this paper assumes you knows the basic functionality of Polarion, its terminology, and that you have at least some experience with the administration interface.

Work Item Types

We have configured Polarion ALM with 4 work item types to support our Scrum process:

- **User Story**: defines what functionality should be added, updated or removed. It is formulated on business language, has business value, and groups all relevant activities together.

- **Improvement**: specified some change(s) that will appear in a future release... code improvements, Documentation tasks, etc.
- **Defect**: I guess it's clear what defect is :)
- **Task**: any activity consuming time and human resources, but the results don't appear directly in products – write a test case, install a demo server, brainstorm discussion about a feature, etc.

You'll notice we don't use Change Requests or Requirements – those are covered by User Stories.

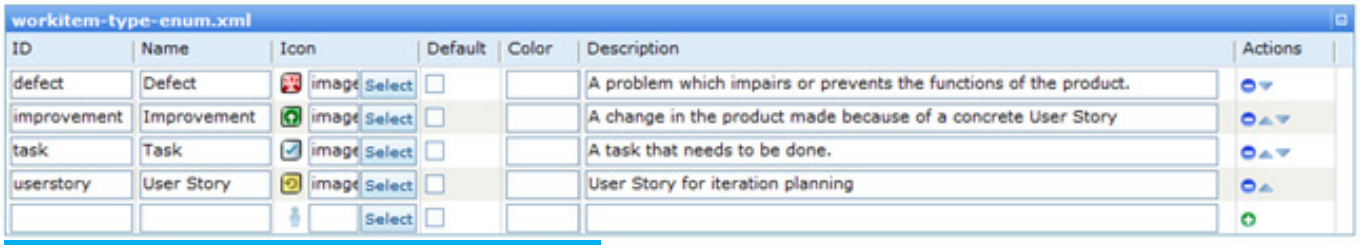

Figure 3 – Work Item type configuration in Polarion ALM

User Story Attributes

Attributes are reflected in Custom Fields we define for each work item type. For User Stories we track:

- **Source of request** (whom to ask for clarifications)
- **Backlog** it belongs to (for sorting by priority exactly as Backlog Owner wanted)
- **Relationships** between User Stories (People may require similar or related things. We need to see those relationships to simplify prioritization and grouping in the Product Backlog)
- **Product edition(s)** that will have a feature
- **Doc not required** – flag if a User Story doesn't require documentation,
- **Who requested** the functionality (customer, prospect, etc.)
- **Responsible developer** – this may seem a contradiction to a team-oriented approach but there's a reason for it. We found it useful to have a single person responsible for each user story, who checks all the activities around it, and who also leads the demos of the feature when the product is ready.
- **State** – most important states are: "Open" (new, to do), "In Progress" (there is active work on it), "Implemented" (implementation activities are finished) and "Done".
- **Initial Estimate** – this is typically empirical data, which the team agrees on. The Time Spent and Remaining Estimates are calculated automatically by Polarion ALM (via inherited fields) from linked child work items (Improvements, Defects, Tasks).
- There are more attributes, specific to our development cycles, but these are the main ones.

Improvement Attributes

As any implementation-related work item, an Improvement has references to the build in which it was implemented (so testers know which build o review), in which build it was reviewed by QA, the branch it was committed to, the assignee, time estimates, etc.

Prioritization of Improvement is typically done in the corresponding User Story. All Improvements planned to a Sprint should be linked to a User Story.

Defect Attributes

Attributes are similar to those of Improvement. Defects can be taken in a sprint without linking to a User Story, and they may be prioritized separately. Most important attributes include:

- **Build** (or Product version) where the problem was discovered[1]
- **Severity** – the impact on customers or internal users
- **Customer** – if reported by a customer, it needs higher priority. Also, the customer might need a patch, so we have to track who should be provided with one.[2]
- **Build** in which the problem was resolved, branches, assignee, estimates and time spent, etc.
- **Known Issue** – defect is not resolved and should be mentioned in "Known Issues" list for the release.

Task Attributes

This type of work item doesn't have direct connection to Customer or builds, therefore it doesn't have any specific attributes. This item also must be linked to a User Story to be selected for a Sprint.

Linking of Work Items

Perhaps less important than work item types, still Polarion's linking capabilities help us in creating work breakdown structure, and we benefit from the Planning features, which take in consideration various types of links.

The most important link types are:

- **Implements**: the relationship of Improvements, Defects and Tasks to the User Story. Until linked child items are resolved, the User Story is not considered Done.
- **Depends on**: meaning should be clear from the name – the linked item must be processed first.
- **Relates to**: flags some relationship between work items; just a hint for developers to review if there is anything relevant or important in the linked item.
- **Parent**: links work items of the same type. Used for decomposition of complex User Stories.
- **Follows**: Some work Items may be resolved in terms of the request, but it turns out they need further work: usability improvement, or a defect resulting from a fix of another.

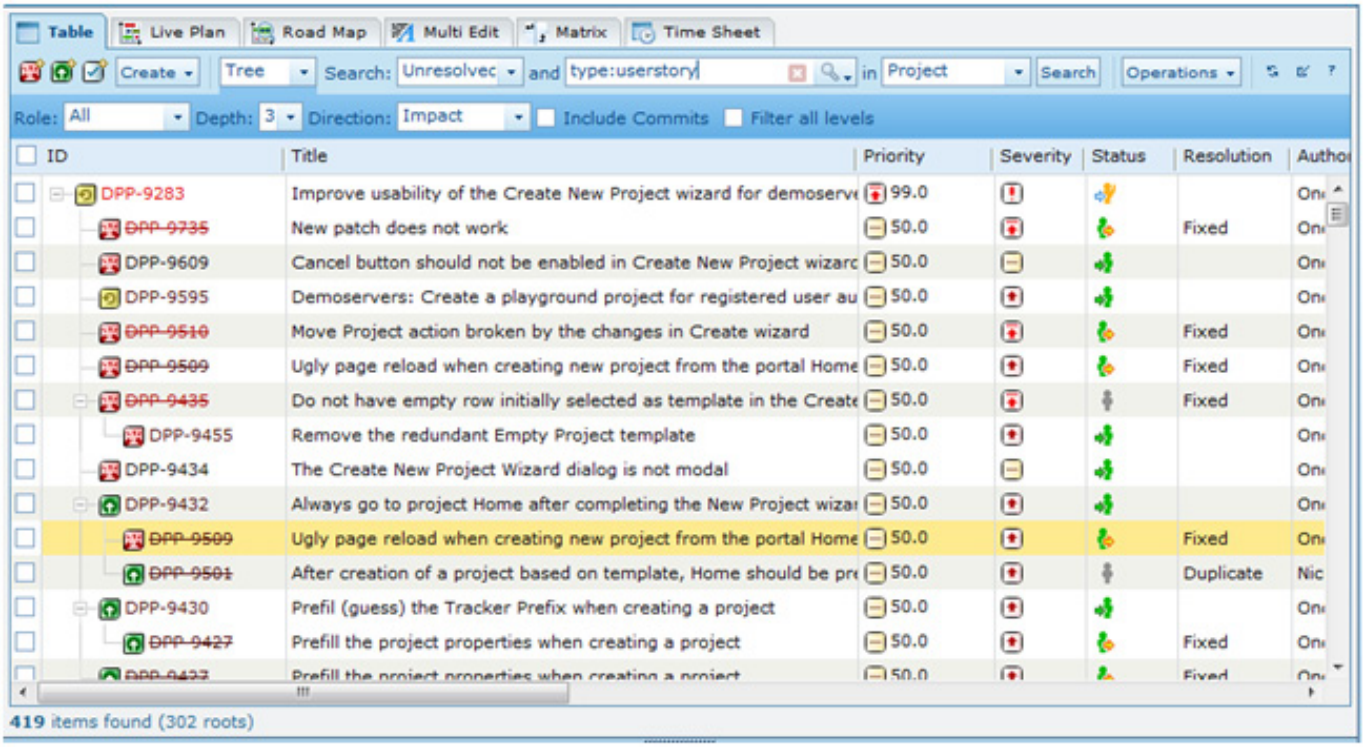

Figure 4 Example of Work Breakdown Structure with our configuration

Product Backlog

Typically the Product Owner writes up items for the Product Backlog in a Word or Excel document and then simply reshuffles them according priority. This approach could easily cause all kinds of problems except for the fact that Polarion ALM enables efficient and coordinated management of such artifacts.

Composing User Stories

Across all owners and stakeholders, we use 3 ways of composing User Stories:

- Through Polarion Web UI ("Create Work Item")
- Through Email sent to the Polarion Maillet
- Through LiveDocs, Polarion's exclusive office document synchronization feature.

Regardless of the authoring method, created work items appear in the Tracker and it's relatively easy for all stakeholders to find them using Polarion's Query Builder. (Such a query might be "**type:userstory AND backlog:usability**"). We often embed queries into Wiki pages so stakeholders don't have to formulate queries. The one shown in Figure 6 below collects all backlogs and displays the top items.

Prioritizing of User Stories

Here our process differs from typical Scrum. We have several relatively independent stakeholders, all committed to common goal, but still in pursuit their own targets (sound familiar?) Therefore, each backlog is prioritized separately by the backlog Owner, who defines the threshold of his or her items to flag those which must appear in the Product Backlog and, ideally, should be discussed by the team.

Outstanding

Owner	Nick Entin
Description	N/A
ID	oustanding
Threshold	severity = critical or blocker
Work Items (unresolved/total)	18 / 53

Unresolved Work Items sorted by severity and priority

(backlog:outstanding AND resolution:######NULL) AND (project.id:"PolarionSVN")

ID	Title	Status	Severity	Priority	Initial Estimate	Time Point	Categories
DPP-9814	Provide the status description of the mailet feature, so we know if it can be published as extension.	Open		Medium [50.0]			
DPP-9813	Configure polarion demoservers so the eval users do not see hundreds of other eval projects	Open		Medium [50.0]			
DPP-9543	Find solution for lack of disk space: Rotate or delete old logs	In Progress		Medium [50.0]	4h	i57 (2009-06-16)	Jobs and Scheduler, Administration
DPP-9194	Provide load/stress test of Polarion	Open		Medium [50.0]			
DPP-8245	VS Plugin in OpenSource - publish version, without Login-trick on the Polaron Community server	Open		Medium [50.0]			
DPP-8229	Plugable functions in query bar	Open		Medium [50.0]			Querying and Index
DPP-7339	Remove Jetspeed from Polarion	Open		Medium [50.0]			
DPP-4799	Restore use of LivePlan for internal projects	Open		Medium [50.0]			
DPP-9605	Documentation for 3.3.1	Open		Medium [50.0]			Documentation
DPP-9604	Documentation for 3.4	Open		Medium [50.0]			Documentation

Showing 10 items of 18 found More

Figure 5 A Stakeholder Backlog

Extracting from Stakeholder Backlogs to Product Backlog

Our next step is to collect all the required items for the Product Backlog. In Polarion ALM, it's quite easy to create a Wiki page which collects all the "top" (i.e. highest priority) items from various backlogs – we just embed an instance of the **{workitems}** macro, supplying correct query to fetch items from the backlogs maintained in the Tracker.

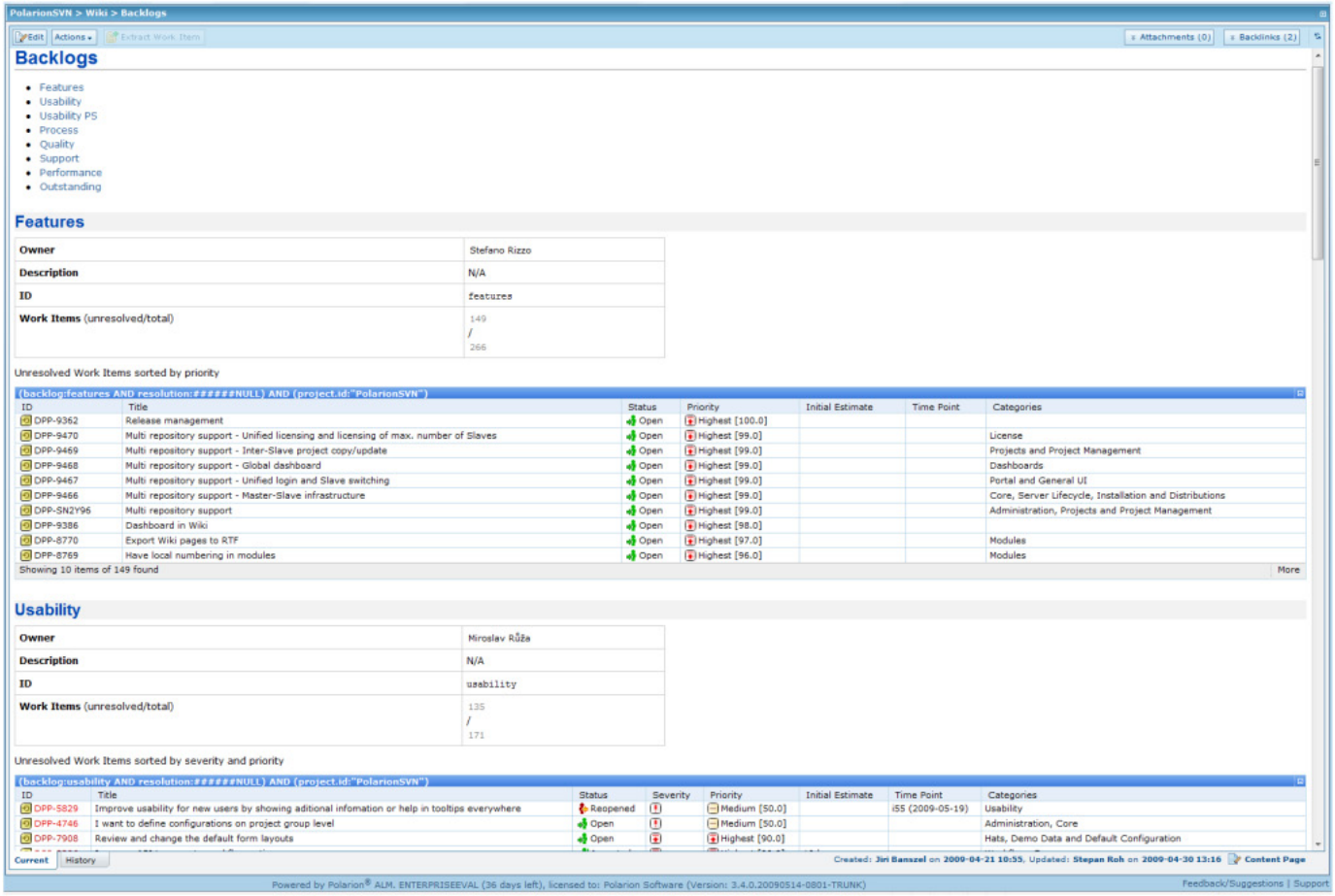

Figure 6 Backlogs wiki page with embedded queries displaying top items different backlogs

Next, the Product Owner prioritizes the list. We've defined a custom integer field "Product Backlog Priority" (PBP)which the Product Owner uses to sort the items accordingly.[3]

NOTE: A highly useful Polarion feature lets you click "More" in a backlog table embedded in a Wiki page, which opens the Work Items table in the Tracker (a prime example of Polarion's integrated approach to tools).

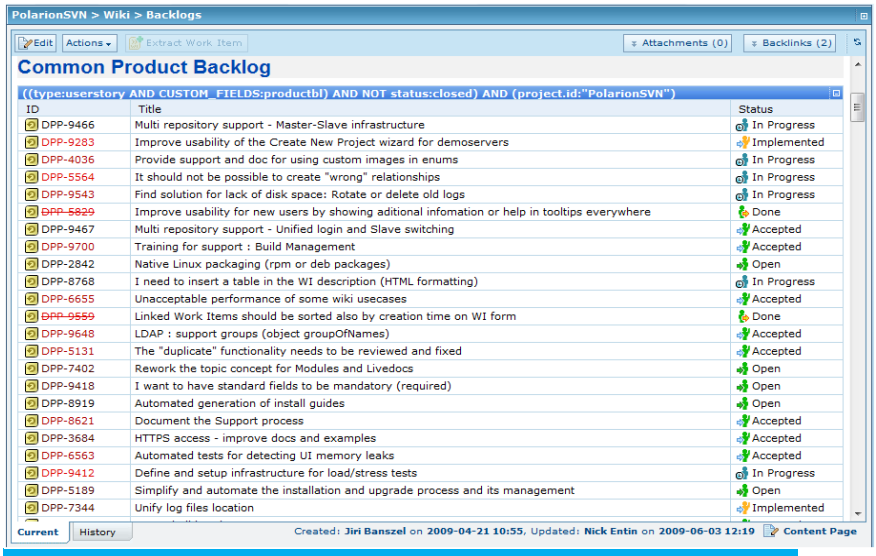

Figure 7 Tracker items comprising the Product Backlog displayed in a Wiki page

The PBP attribute also helps to track down if there were some changes in a particular backlog that are not yet reflected in the Common one. For example, a query that retrieves all the "important" User Stories which don't have the PBP field set:

Important Backlog items, not entered to the Product Backlog

((type:userstory AND NOT CUSTOM_FIELDS:productbl AND NOT status:closed AND resolution:######NULL) AND ((backlog:features A

ID	Title	Status	Backlog	Severity	Priority
DPP-9814	Provide the status description of the mailet feature, so we know if it can be published as extension.	Open	Outstanding		Medium [50.0]
DPP-9813	Configure polarion demoservers so the eval users do not see hundreds of other eval projects	Accepted	Outstanding		Medium [50.0]
DPP-9758	Install Polarion as Service on Windows	Open	Support		Highest [90.0]
DPP-9801	Support Subversion 1.6	Open	Support		High [70.0]

4 items found More

Figure 8 Wiki page section with query for potentially missing backlog entries

Additional tips from our development process

We actively use Polarion's Auto-assignment feature when creating new work items. This enables immediate assignment to a Senior Developer, who will potentially lead the implementation. This developer gets an email notification and sees the new item assigned to him. This encourages early review of posted user stories, provides read-filtered input for the planning meetings, etc.

To simplify prioritization the "weight" or "initial estimate" of a User Story is important, and automatic assignment helps to get initial review and communication going even before the planning meeting.

Also, we've configured the User Story work item type to aggregate the values of Remaining Estimate fields from any child items.

So by brainstorming, and reaching agreement at the Planning Meeting, we identify an Initial Estimate value for each User Story. Later however, when it is decomposed into to Improvements, Tasks and Defects, we can often spot variations – that particular work actually takes less time, or some additional task was not anticipated, and was discovered after implementation began. This data is extremely helpful for re-planning of any User Story that was not finished to then next Sprint.

Tips

- Use Auto-assignment for new work items
- Configure the User Story work item type to aggregate the values of Remaining Estimate fields from any child items

The Sprint: Meetings

Meetings are possibly the most important assets of Scrum. Meetings are when the team commits to the Product Owner on the amount of work (features) they will address over each sprint. They discuss the progress in *Daily Scrums* and, finally, access results at the last meeting. In this section and the next, I'll describe how we manage those meetings with the Polarion development team.

The Planning Meeting

The goal of the Planning Meeting is to ensure that the team fully understands the Product Backlog items, to commit the team to implementing agreed-on items in upcoming sprint, and to ensure proper distribution of work among team members. During the Planning Meeting dependencies between teams are also identified to allow as much parallel work by the teams as possible, keeping the same focus for the iteration.

The planning entity for the sprint is the User Story. Each one has a customer (the person who formulated the requirement) and an owner – typically a Senior Developer, who then follows the User Story through the full lifecycle.

Typically the meeting is split to two parts. The first part involves the Product Owner, and possibly other stakeholders, to ensure common understanding of things to be done and commit the team to some work items.

The second part is a rather internal meeting, where the team decides who will implement what and splits the User Stories into concrete Task and Improvement work items, validating capacity of the team using the Polarion ALM's LivePlan feature.[4]

Normally the User Stories in the Product Backlog have been inspected and time-estimated by developers in advance. Team members come prepared with questions, and perhaps concerns about conflict with some agreements or principles, or inconsistencies.

Out of the Planning Meeting come the User Stories selected for the sprint, which becomes a Time Point assignment in Polarion ALM. Results of the planning meeting are presented in a special wiki page showing the agreed-upon Sprint Backlog:

Sprint Backlog

((type:userstory AND timePoint.id:i60)) AND (project.id:"PolarionSVN")

ID	Title	Status
DPP-9768	Service Release 3.3.1	In Progress
DPP-10018	Remove "Beta" from VS Integration, check compatibility with 3.3.0 and upload repackaged versions	In Progress
DPP-8825	Generate SDK documentation automatically	Implemented
DPP-4036	Provide support and doc for using custom images in enums	Reopened
DPP-5564	It should not be possible to create "wrong" relationships	Implemented
DPP-9737	RPM packaging, implementation research	Accepted
DPP-9953	Add Operation to WI Table to move all displayed WIs to Module	Accepted
DPP-9386	Dashboard in Wiki	In Progress
DPP-8768	I need to insert a table in the WI description (HTML formatting)	In Progress
DPP-9412	Define and setup infrastructure for load/stress tests	In Progress
DPP-9648	LDAP : support groups (object groupOfNames)	Implemented
DPP-10052	Branch documentation sources and builds	Accepted
DPP-3471	Define the benchmarks for Polarion data load and (related) performance	In Progress
~~DPP-9814~~	Provide the status description of the mailet feature, so we know if it can be published as extension.	Done

14 items found More

Figure 9 Sprint Backlog in the integrated wiki

Slipping User Stories

We pay special attention to User Stories committed to a Sprint, but not completed. It's very natural to slip to the next Sprint because "it's just taking a little longer".

One expects that as soon it's moved to next Sprint, it will be done on the first day. No! Experience shows that developers often leave unfinished User Stories to end of iteration because they are easy to complete. But in reality - they get behind with other tasks, and the slipped User Story remains unfinished and slips even further into next Sprint.

A burning question in our Planning Meetings is: "If this User Story was not addressed on last Sprint, how can we ensure that our new commitment to this User Story will actually be realized?"

Daily Scrums

Daily Scrums might be the most complicated part of Scrum because it requires changing perceptions. Too many of us interpret meetings as means of getting tasks and reporting back, but Scrum in general, and Daily Scrums particular, are about helping the team to understand the current situation and to adjust if necessary. Daily Scrums let team members to synchronize and all can check whether sprint goals are still feasible and if not, take decisions about what to change. No reports are made, and the Scrum Master poses one simple question: “Are we sure we’ll meet our Sprint Goals? Please show/explain how we do that!”

We use the Wiki Task Board to track progress of our sprint execution. :[5]

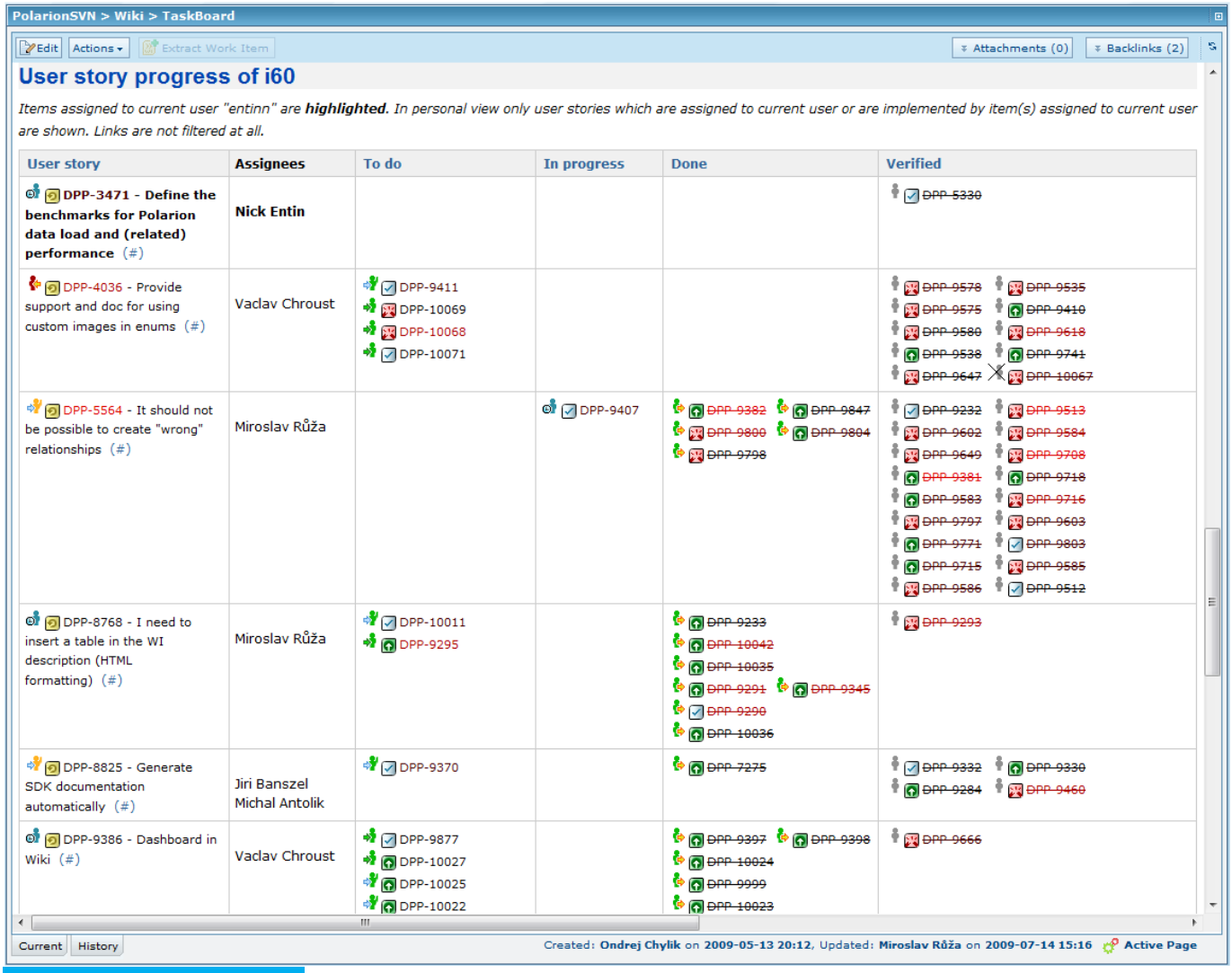

Figure 10 Task Board in the wiki

Since our teams are small and we adjust daily Scrums to the lunch hour :). When the most important questions are answered the team may go to lunch and discuss very low level details, if needed.

The Assessment Meeting

Every iteration ends with an Assessment Meeting, where every developer presents his/her work, either as a document (if the task was to "specify" , etc.), or as a demo of the implementation in the product. Each User Story should already have been tested by QA.[6]

For the Assessment Meetings we check only those items marked as "Done". [7] As input for the Assessment meeting we use yet another, more compressed, variant of the Task Board:

Figure 11 Task Board Summary on the Wiki

The end part of the Assessment Meeting is for introspection and Lessons Learned. Ideally, it is a time to discuss how we could optimize the process to implement more over a sprint, but typically we find ourselves trying to identify things that were less than perfect in our process and/or the implementation of some feature implemented in the sprint. We also try to identify additional synchronization risks, problems of communication, involve additional people to show them our dependency, which was not fulfilled, and other subjects.

The Sprint: Development

During a sprint, the development team continuously integrates all changes, and updated versions of the product are installed on the internal servers daily to prove stability and allow earlier testing of new functionality by other people (testers, doc writers, etc.)

Every developer should know his/her personal plan, which matches the team plan set during the planning meeting. Developers track tasks via…

- Personal queries, like "assigned to me in current TimePoint"
- E-mail notifications of newly assigned work items
- The LivePlan chart and corresponding Wiki pages in our Polarion system

Burn-Down Charts

We configure the LivePlan view to show only entities assigned to a TimePoint (usually the current sprint). It shows only leaves of the work item work-break structure (i.e. it doesn't render User Stories on the plan if it has child items – improvements, tasks or defects).

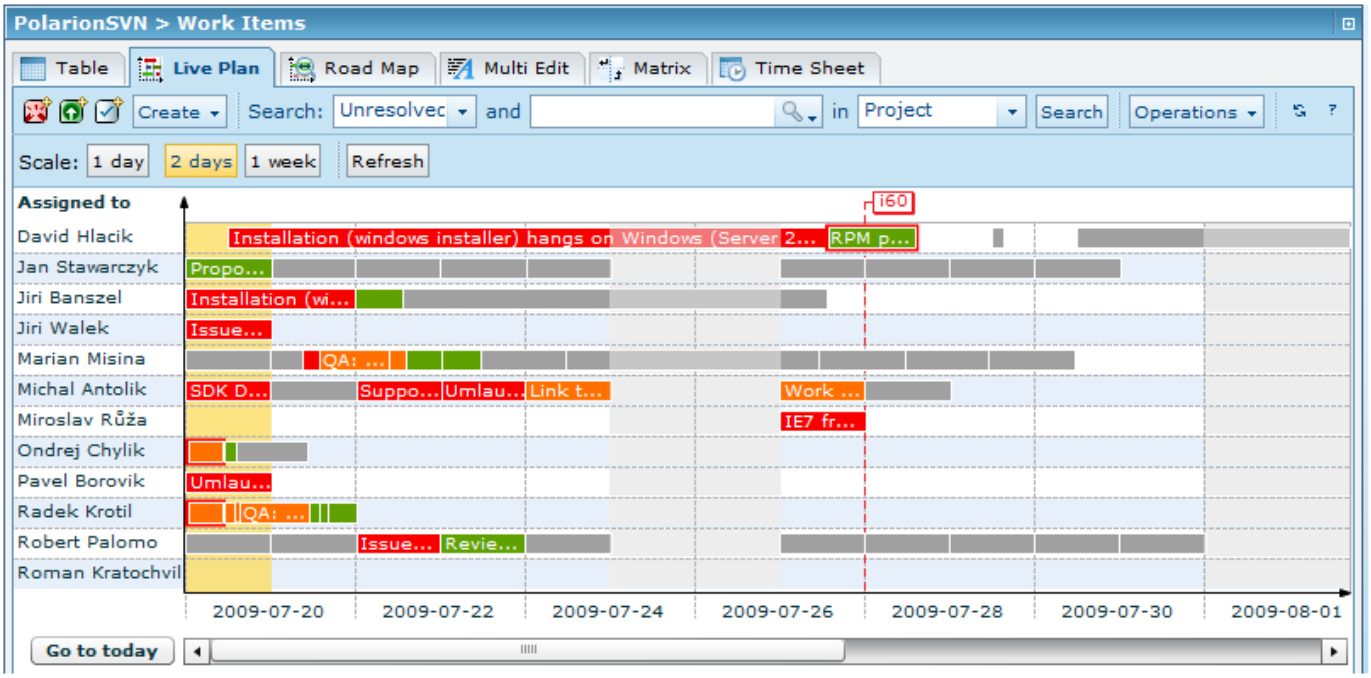

Figure 12 Burn-down chart in the Live Plan

The LivePlan also reflects all non-working days (configured in the global working calendar), and personal days off (configured in developers' personal working calendars). We get very clear information whether or not the Sprint goals are still achievable in the sprint time frame. The plan is ordered by priorities and severities in the way developers have agreed upon in the Planning Meeting (part 2) . Correspondingly, less important items should be at the end of the plan. We also have Wiki pages that highlight the progress of the team and remaining time:

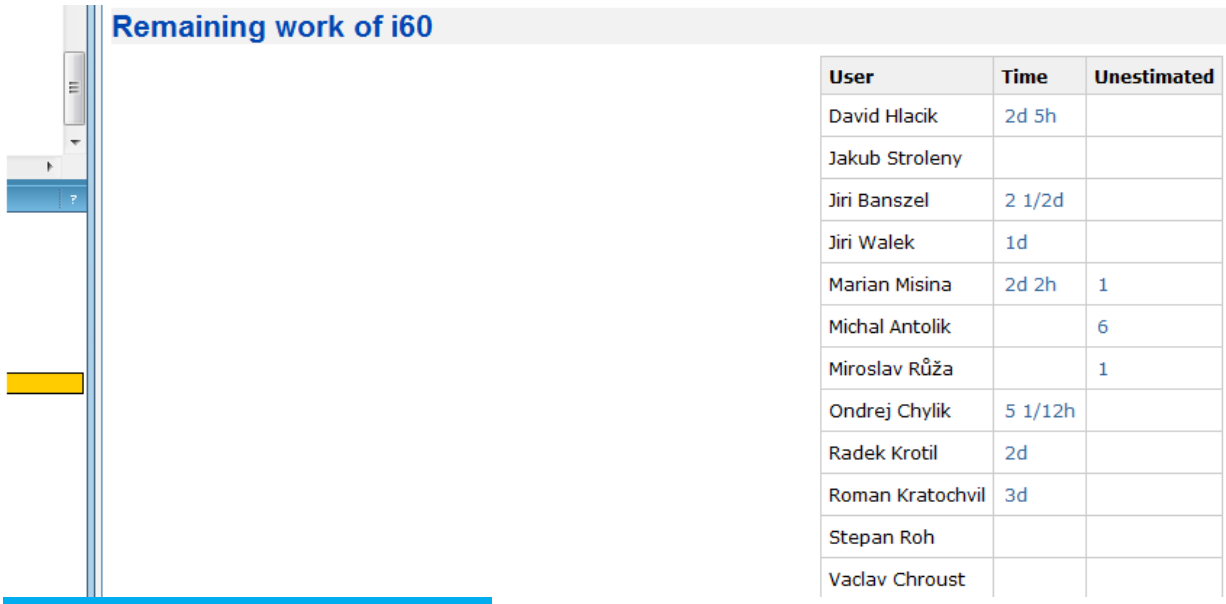

Figure 13 Remaining time estimate in the Wiki

Polarion's Road Map view also gives a clear picture of the work items in tabular form:

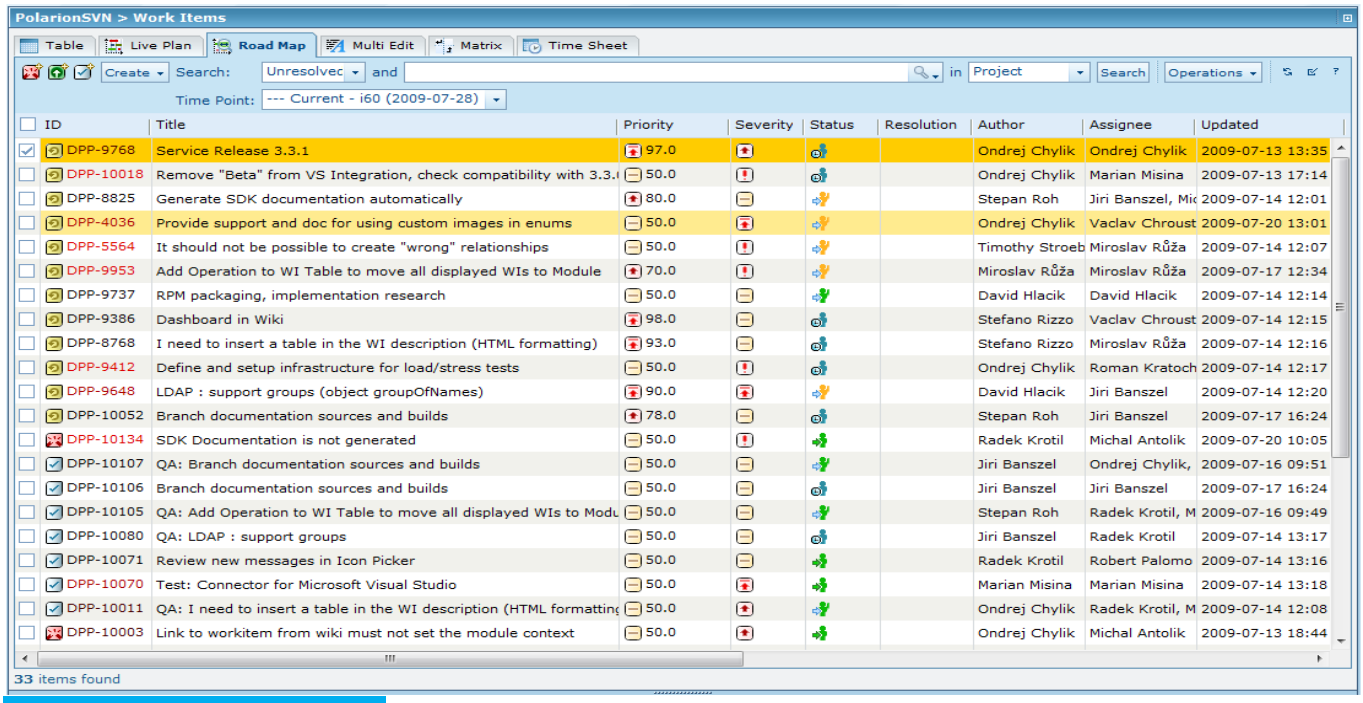

Figure 14 Polarion ALM Roadmap View

Testing and Documentation

As implementation enters the "Implemented" state, it should be taken over to QA and Documentation. There is automatic testing through unit tests and so on, but every feature should pass QA control to ensure consistency of the implementation, acceptable levels of usability, that licensing and configuration for different product lines is correctly implemented m and common user acceptance.

Typically QA and Documentation starts in parallel with development – based on a specification document (normally a wiki page). Final definition of the test cases and documentation typically happens at the point where implementation is really considered done, and first round of review (also by QA) is passed.[8] The User Story is populated with corresponding QA and Doc Tasks (might be several of each) and the flags, set by the User Story owner, that proper QA and documentation are done.

Conclusion

I hope you have gained some insights into how Scrum and Polarion ALM can work in conjunction. In closing, let me suggest some possible next steps for you.

- If you are new to Polarion ALM, I highly recommend our Test Drive Server where you can explore Polarion ALM as much as you like without any time limits. For information and to create your user account, visit www.polarion.com/products/alm/demo.php.
- http://extensions.polarion.com – here you can find a constantly growing cadre of extensions for Polarion, including examples of the Task Board I have referred to , workflow functions, integrations with third-party solutions, project templates, and more.
- http://forums.polarion.com – here you can ask questions and discuss with other customers your approach to Polarion.

Did You Know?

Polarion Software's integrated ALM solutions manage requirements engineering, team collaboration, bug tracking, version control, CMMI compliance, Agile process, and more for many kinds of projects with **over 500,000 users worldwide** in broad range of industries. Visit **www.polarion.com** for more information.

About the Author

Nick Entin is VP for Research & Development at Polarion Software. Nick oversees the development of all Polarion requirements management, application lifecycle management, and team collaboration software products. He is a member of the Scrum Alliance and a Certified ScrumMaster. You can read his profile at http://www.polarion.com/company/people/index.php.

Notes

PAGE 4

[1] This is a string field, because it should allow values like "before version 3.2.1" or "after nightly build Apr 12th 2008"

[2] Other work item types also have the Customer attribute, but it reflects who has requested the feature or proposed an Improvement, so it's role is less important than in the Defect type.

PAGE 7

[3] You might also configure special "Hat" *Product Owner*. I personally use my own table settings to expose the PBP column, so I can easily reshuffle items.

PAGE 9

[4] LivePlan reveals over/under- tasked people, potential bottlenecks, etc as soon as we plug in our tentative plan data.

PAGE 10

[5] The Wiki Task Board is a free extension available on Polarion POP – the Polarion extensions portal at http://extensions.polarion.com.

PAGE 11

[6] Documentation may not happen in parallel with each iteration. Reality is that documenting all things together is not always possible. We use special a User Story "UNDONE: release X.Y.Z" to link all the things to be addressed in a Stabilization Sprint before a corresponding release.

[7] From a workflow point of view, User Stories are marked as "Implemented" (programming is finished), "Done" (QAed, Documented), "Verified-Done" (when corresponding stakeholder agrees that this functionality is really what was requested and expected). Those lacking documentation are still marked as "Done", anticipating completion of documentation in a Stabilization Sprint.

PAGE 13

[8] Otherwise unforeseen issues might cause implementation to vary from the original specification and lead to refinement or even change of the specification. Of course the Product Owner or corresponding stakeholders are the ones who ultimately decide any change of specification.

Europe, Middle-East, Africa: Polarion Software GmbH
Lautlinger Weg 3 — 70567 Stuttgart, GERMANY
Tel +49 711 489 9969 - 0
Fax +49 711 489 9969 - 20
www.polarion.com - info@polarion.com

POLARION

Americas & Asia-Pacific: Polarion Software, Inc.
406 Tideway Dr. Alameda, CA 94501, USA
Tel +1 877 572 4005 (Toll free)
Fax +1 510 814 9983
www.polarion.com - info@polarion.com