DIGITAL INDUSTRIES SOFTWARE

Polarion application lifecycle management

Leveraging DevOps as a natural component of ALM

Executive summary
This white paper describes how Siemens Digital Industries Software builds Polarion™ software using a DevOps approach. It provides an overall picture of the toolchain we use and how we guide our process, while outlining several examples you can easily implement in your production environment.

SIEMENS

Contents

Abstract

It has become common for analysts to replace the term application lifecycle management (ALM) with phrases like enterprise agile planning tools (EAPT) and software lifecycle management (SLM). Some analysts suggest there is no need for ALM and everything can be done via popular DevOps tools like GitLab or GitHub. However, DevOps can be a natural component of ALM, depending on how well an ALM tool implements the DevOps domain and integrates with its established solutions.

The process for complex development

A complex development requires a complex process. The process must ensure the plans are realistic, correspond with quality guidelines, ensure security, enable teams to collaborate effectively and so on. In this section we'll discuss the problem statement and the problem solution.

What we have:

- A complex product
- The release cadence is known in advance and must be followed
- Every release should have substantial functional and quality increments
- A continuous demand to change or adopt a new architecture and update the user interface (UI) to align with other Siemens products and implement the best UI practices and components
- Scrum teams that integrate different components to the product line and must be aware of crossover dependencies
- Cross-product maintenance tasks (defect fixing, performance and scalability improvements) that typically affect many areas of the application. These are not the responsibility of a single team and their impact on the codebase can be far-reaching
- Continuously changing prioritizations (new/funded projects, customer escalations and estimate changes) that may lead backlog reprioritization

What we need:

- An infrastructure that enables several teams to work in parallel without disturbing each other, especially during the integration phases
- A system that can track how the execution of multiple topics progresses for reporting and synchronization purposes
- Developed features that are thoroughly tested. This should be done locally by the development team and again after integration. This ensures the general stability of the release and eliminates possible regressions
- Multi-level integration of the source code that provides traceability between tasks/requirements and the code (enables the code review process to make sure that all changes can be audited

- A useable collaboration platform for teams to effectively consult with each other. It should help facilitate:
  - Discussion threads with an easy way to find notes and conclusions
  - Requesting and receiving specific expertise from the entire community
  - On-site customer support. This typically means having a product manager and/or owner continuously available for ad hoc consultancy
- The ability to report progress continuously and rapidly

Big Picture

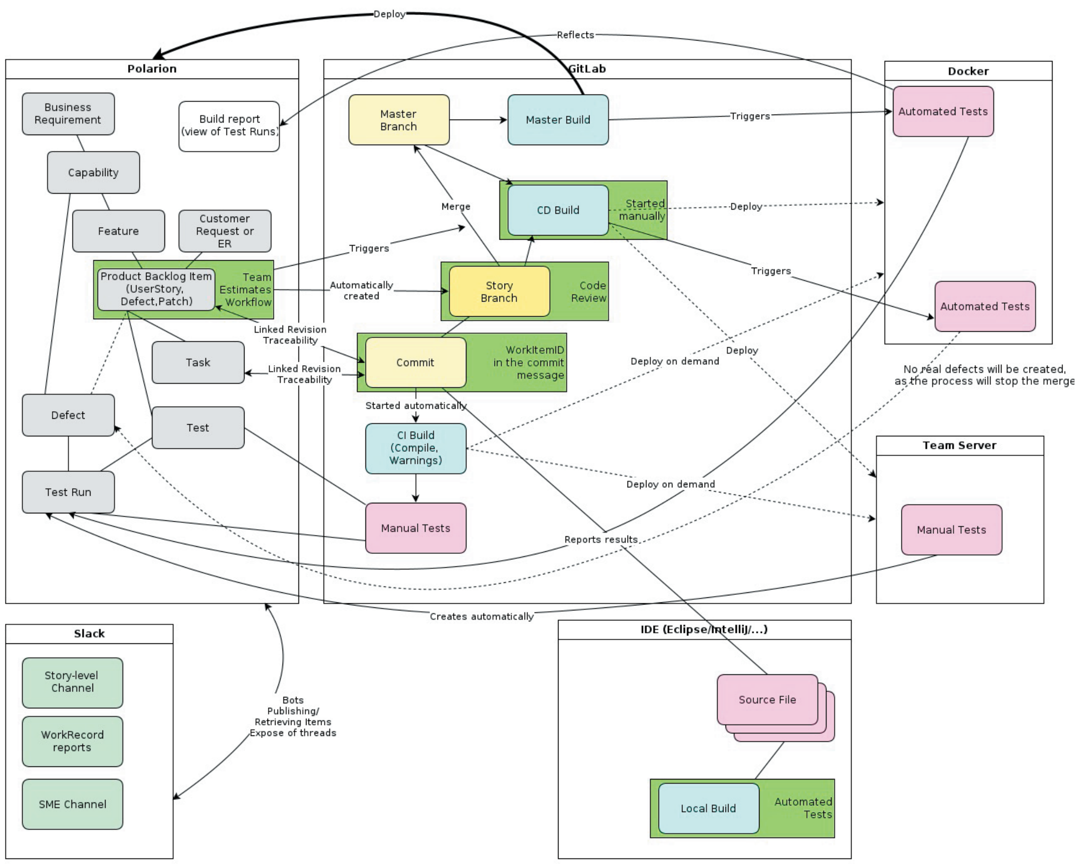


Figure 1. Polarion's R&D process in visual form.

Siemens' toolset and infrastructure:

- Polarion — central access point:
  - General process orchestration
  - Artifacts (requirement documents, stories, tasks, defects, tests, etc.) lifecycle management, including full traceability of changes and workflows as the standard operating procedure (SOP) driver
  - Estimation, prioritization and planning
- GitLab — build management and a continuous integration (CI) and continuous delivery/deployment (CD) infrastructure:
  - Branching and merging
  - Compiling and building
  - Test automation execution
- Hardware and software virtualization:
  - A set of servers and containers for local and global test environments
  - Team-specific test servers
  - Version-specific reference servers to reference, for example, how a feature worked in a specific past product version or to replicate an issue
  - Monitoring, auto deployment, etc.
- Development/engineering tools:
  - Java IDEs — Eclipse, IntelliJ, VisualStudio, etc.
  - Profiling tools and frameworks — JProfiler, etc.
  - Test automation tools/frameworks — Selenium, Junit, Cucumber, etc.
  - Documentation tools and frameworks — X-cat, Oxygen, Jabber, etc.
- Collaboration tools:
  - Instant messaging (IM) — Slack, MS Teams, etc.
  - Filesharing — OneDrive, SharePoint, etc.

Polarion is part of the Xcelerator portfolio, the comprehensive and integrated portfolio of software and services from Siemens Digital Industries Software.

Business requirements

Developing a great product begins with constructing innovative ideas on how to address existing or projected customer needs. Ideas can be new or based on improving on existing functionalities and they can be used to implement established solutions in the real world. Typically, business requirements are represented by a set of documents that describe the problem statement and a proposal of what must be addressed. These requirements will be implemented in our environment following the Scaled Agile Framework (SAFe) and by using Scrum/Kanban on the team level.

SAFe is a knowledge base of proven integrated principles, practices and competencies for lean and agile methodologies and DevOps, enabling large enterprises to idealize, plan and execute big projects that have dependencies, business constraints, etc.[1]

Capabilities

A capability is a higher-level solution behavior that typically spans multiple Agile Release Trains (ARTs). Capabilities are sized and split into numerous features to facilitate their implementation in a single program increment (PI). A typical capability in our context will be a significant portion of a functionality, for example, related to a particular domain or commonly used set of services and frameworks, or a new architecture approach.

Capabilities get grouped into epics to enable a higher level of aggregation and strategic planning. An epic is a container for a significant solution development initiative that captures the more substantial investments in a portfolio. Due to their considerable scope and impact, epics require the definition of a minimum viable product (MVP) and lean portfolio management (LPM) approval prior to implementation.

Typically, epics and capabilities require the most attention from top management, product management and developer leads because these managers control how the budget is aligned across teams, corresponding capacities given, the execution plan is drafted and the risks identified. Often, the capabilities are not linked directly to a customer commitment and serve as a platform for implementing many of the features described below.

Features
A feature is a service that fulfills a stakeholder need. Each feature includes a benefit hypothesis and acceptance criteria, then is sized or split as required so that it can be delivered by a single ART in a PI. For us, a feature may represent a business case, which is a sellable, functional and self-efficient implementation.

Customer request — enhancement request
An enhancement request (ER) is recorded when a business customer requests the enhancement of an existing functionality. Typically, they are usability or functional additions to what was delivered out-of-the-box (OOTB) and should increase productivity. These items are prioritized by support and product management, then added to the development backlog.

Product backlog item
A product backlog item is anything in our process that must be scheduled in a sprint. User stories, product-wide defects and patches are all product backlog items. We initially referred to all of them as user stories, but as our process evolved, we split them into additional categories because different stakeholders prioritize different things. For example, defects are triaged and prioritized by a committee. Once that is complete, the team's product owner determines an appropriate sprint priority. On the other hand, patches are decided by product management. Once decided, patch creation and distribution to customers typically gets passed to a team. This team might not have had any involvement in fixing the defects addressed by the patch.

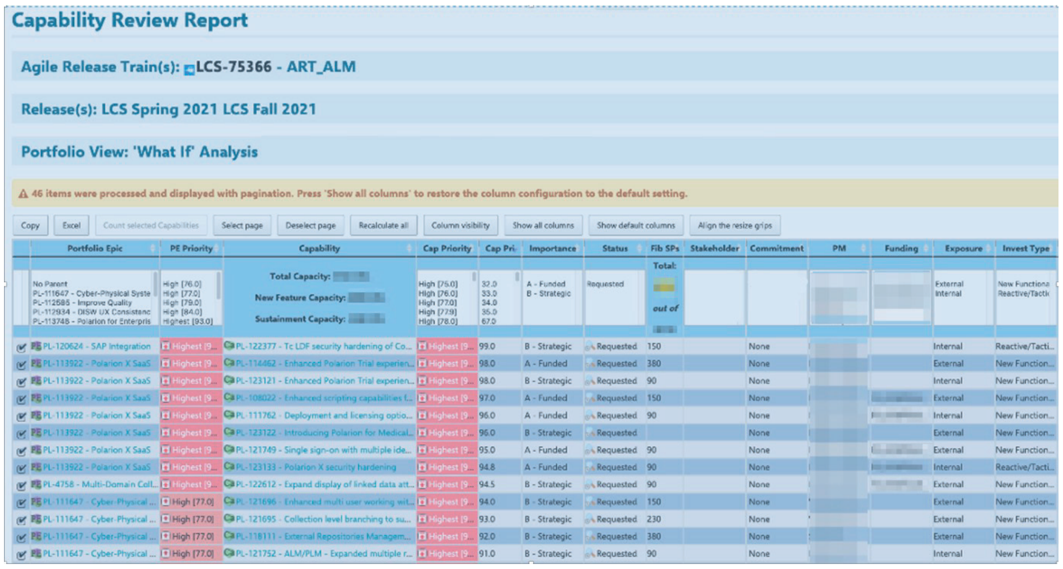
A user story is the most widely used agile item for capturing needs and requirements. Its purpose is to capture the natural conversation surrounding what must be built in the product from the user's perspective. It should initiate and track the discussion between who wants the feature and the developers that are tasked to build it. It is essential that developers understand the feature's intended use and create the best possible solution in architectural and technological boundaries. When the development team understands why a user wants it and what the user wants to achieve, they can come up with a set of possible solutions.[2]

Task
A task is a piece of work that brings the user story toward its implementation. Usually, several tasks are created for a user story to identify how much of a team's involvement is required and whether other parties should be involved in the sprint.

Test
Apart from the Definition of Done (DoD) and the acceptance criteria of a user story, a set of tests can be defined to provide repeatable evidence that a delivered functionality works as intended in current and future contexts. Many of the tests are written in code (test automation) and do not require individual authoring as a work item for a user story. However, when the automatic test is executed, the corresponding object is automatically created and the execution results are tracked in Polarion for each test run.

Test run
A collection of tests executed to prove a selected product area functions correctly.

Planning
While planning strategically, capabilities are prioritized and assigned to their corresponding departments. Then, they are estimated and provided with the relevant capacities to establish their completion.



Capability Review Report

Agile Release Train(s): LCS-75366 - ART_ALM

Release(s): LCS Spring 2021 LCS Fall 2021

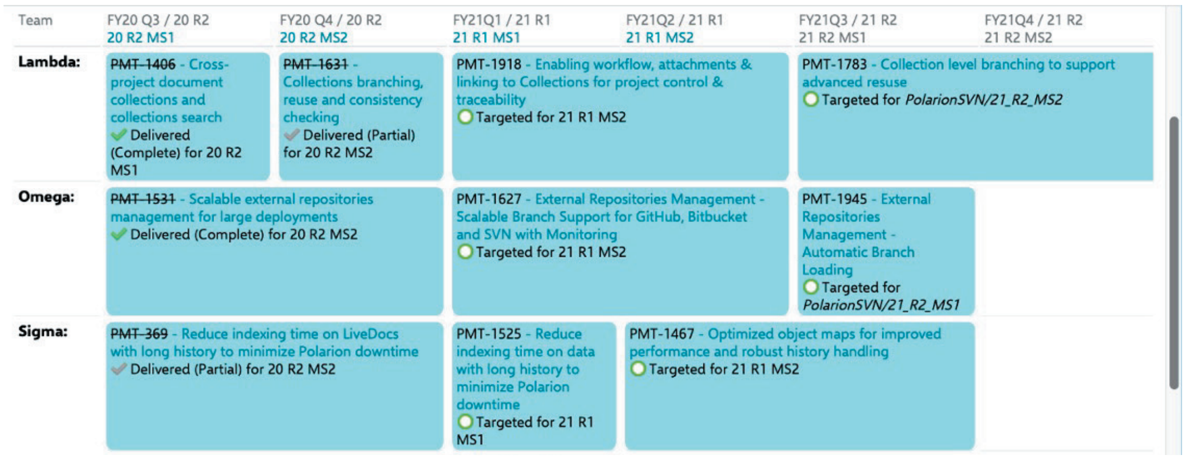
Portfolio View: 'What If' Analysis

⚠ 46 items were processed and displayed with pagination. Press 'Show all columns' to restore the column configuration to the default setting.

	Portfolio Epic	PE Priority	Capability	Cap Priority	Cap Pri	Importance	Status	Fib SPs	Stakeholder	Commitment	PM	Funding	Exposure	Invest Type
	No Parent PL-111647 - Cyber-Physical Syste PL-112585 - Improve Quality PL-112934 - DISW UX Consistenc PL-113748 - Polarion for Enterpris	High [76.0] High [77.0] High [79.0] High [84.0] Highest [93.0]	Total Capacity: [redacted] New Feature Capacity: [redacted] Sustainment Capacity: [redacted]	High [75.0] High [76.0] High [77.0] High [77.9] High [78.0]	32.0 33.0 34.0 35.0 67.0	A - Funded B - Strategic	Requested	Total: [redacted] out of [redacted]			[redacted]	[redacted]	External Internal	New Functiona Reactive/Tacti
✓	PL-120624 - SAP Integration	Highest [9...	PL-122377 - Tc LDF security hardening of Co...	Highest [9...	99.0	B - Strategic	Requested	150		None	[redacted]		Internal	Reactive/Tacti...
✓	PL-113922 - Polarion X SaaS	Highest [9...	PL-114462 - Enhanced Polarion Trial experien...	Highest [9...	98.0	A - Funded	Requested	380		None	[redacted]		External	New Function...
✓	PL-113922 - Polarion X SaaS	Highest [9...	PL-123121 - Enhanced Polarion Trial experien...	Highest [9...	98.0	B - Strategic	Requested	90		None	[redacted]		Internal	New Function...
✓	PL-113922 - Polarion X SaaS	Highest [9...	PL-108022 - Enhanced scripting capabilities f...	Highest [9...	97.0	A - Funded	Requested	150		None	[redacted]	[redacted]	External	New Function...
✓	PL-113922 - Polarion X SaaS	Highest [9...	PL-111762 - Deployment and licensing optio...	Highest [9...	96.0	A - Funded	Requested	90		None	[redacted]	[redacted]	Internal	New Function...
✓	PL-113922 - Polarion X SaaS	Highest [9...	PL-123122 - Introducing Polarion for Medical...	Highest [9...	96.0	B - Strategic	Requested			None	[redacted]		External	New Function...
✓	PL-113922 - Polarion X SaaS	Highest [9...	PL-121749 - Single sign-on with multiple ide...	Highest [9...	95.0	A - Funded	Requested	90		None	[redacted]	[redacted]	External	New Function...
✓	PL-113922 - Polarion X SaaS	Highest [9...	PL-123133 - Polarion X security hardening	Highest [9...	94.8	A - Funded	Requested	90		None	[redacted]	[redacted]	Internal	Reactive/Tacti...
✓	PL-4758 - Multi-Domain Coll...	Highest [9...	PL-122612 - Expand display of linked data att...	Highest [9...	94.5	B - Strategic	Requested	90		None	[redacted]	[redacted]	External	New Function...
✓	PL-111647 - Cyber-Physical ...	High [77.0]	PL-121696 - Enhanced multi user working wit...	Highest [9...	94.0	B - Strategic	Requested	150		None	[redacted]		External	New Function...
✓	PL-111647 - Cyber-Physical ...	High [77.0]	PL-121695 - Collection level branching to su...	Highest [9...	93.0	B - Strategic	Requested	230		None	[redacted]		External	New Function...
✓	PL-111647 - Cyber-Physical ...	High [77.0]	PL-118111 - External Repositories Managem...	Highest [9...	92.0	B - Strategic	Requested	380		None	[redacted]		External	New Function...
✓	PL-111647 - Cyber-Physical ...	High [77.0]	PL-121752 - ALM/PLM - Expanded multiple r...	Highest [9...	91.0	B - Strategic	Requested	90		None	[redacted]		Internal	New Function...

Figure 2. Capability review report.

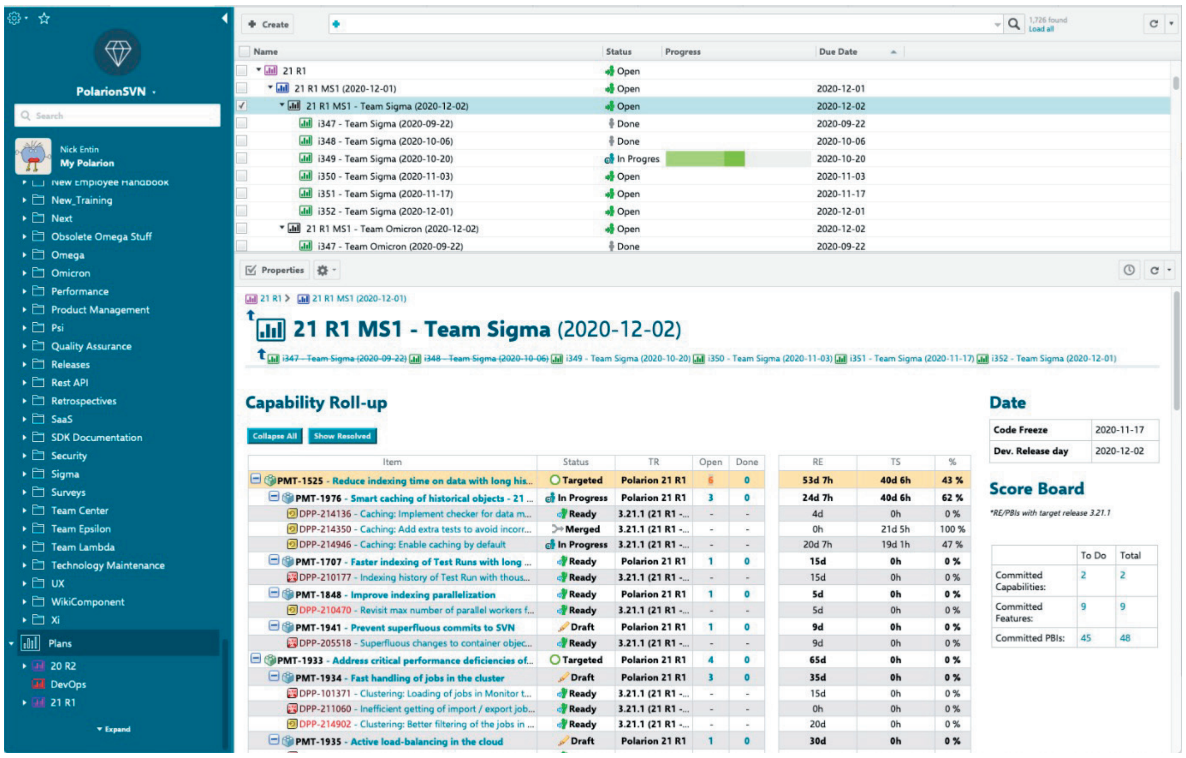
On the product/project level, a plan may be distributed among Scrum teams to make sure the work is distributed appropriately and required synchronization is identified.



Team	FY20 Q3 / 20 R2 20 R2 MS1	FY20 Q4 / 20 R2 20 R2 MS2	FY21Q1 / 21 R1 21 R1 MS1	FY21Q2 / 21 R1 21 R1 MS2	FY21Q3 / 21 R2 21 R2 MS1	FY21Q4 / 21 R2 21 R2 MS2
Lambda:	~~PMT-1406~~ - Cross-project document collections and collections search ✓ Delivered (Complete) for 20 R2 MS1	~~PMT-1631~~ - Collections branching, reuse and consistency checking ✓ Delivered (Partial) for 20 R2 MS2	PMT-1918 - Enabling workflow, attachments & linking to Collections for project control & traceability ○ Targeted for 21 R1 MS2		PMT-1783 - Collection level branching to support advanced reuse ○ Targeted for *PolarionSVN/21_R2_MS2*	
Omega:	~~PMT-1531~~ - Scalable external repositories management for large deployments ✓ Delivered (Complete) for 20 R2 MS2		PMT-1627 - External Repositories Management - Scalable Branch Support for GitHub, Bitbucket and SVN with Monitoring ○ Targeted for 21 R1 MS2		PMT-1945 - External Repositories Management - Automatic Branch Loading ○ Targeted for *PolarionSVN/21_R2_MS1*	
Sigma:	~~PMT-369~~ - Reduce indexing time on LiveDocs with long history to minimize Polarion downtime ✓ Delivered (Partial) for 20 R2 MS2		PMT-1525 - Reduce indexing time on data with long history to minimize Polarion downtime ○ Targeted for 21 R1 MS1	PMT-1467 - Optimized object maps for improved performance and robust history handling ○ Targeted for 21 R1 MS2		

Figure 3. Distribution of plan among Scrum teams apability review report.

When the planning reaches the Scrum team level, capabilities are broken down into features and then into user stories — defects, patches, or other relevant project backlog increments (PBIs). On each level, related activities need to be planned and have their progress evaluated over time. For example, a capability must be aligned with the capacity of the assigned team(s) and features. It should also be planned so it can be delivered to a customer by the target date. Teams should be able to assess a user story's level of complexity based on the number of story points assigned to it.



Name	Status	Progress	Due Date
21 R1	Open		
21 R1 MS1 (2020-12-01)	Open		2020-12-01
✓ 21 R1 MS1 - Team Sigma (2020-12-02)	Open		2020-12-02
i347 - Team Sigma (2020-09-22)	Done		2020-09-22
i348 - Team Sigma (2020-10-06)	Done		2020-10-06
i349 - Team Sigma (2020-10-20)	In Progres	🟩	2020-10-20
i350 - Team Sigma (2020-11-03)	Open		2020-11-03
i351 - Team Sigma (2020-11-17)	Open		2020-11-17
i352 - Team Sigma (2020-12-01)	Open		2020-12-01
21 R1 MS1 - Team Omicron (2020-12-02)	Open		2020-12-02
i347 - Team Omicron (2020-09-22)	Done		2020-09-22

21 R1 > 21 R1 MS1 (2020-12-01)

21 R1 MS1 - Team Sigma (2020-12-02)

~~i347 - Team Sigma (2020-09-22)~~ ~~i348 - Team Sigma (2020-10-06)~~ i349 - Team Sigma (2020-10-20) i350 - Team Sigma (2020-11-03) i351 - Team Sigma (2020-11-17) i352 - Team Sigma (2020-12-01)

Capability Roll-up

Item	Status	TR	Open	Done	RE	TS	%
PMT-1525 - Reduce indexing time on data with long his...	Targeted	Polarion 21 R1	6	0	**53d 7h**	**40d 6h**	**43 %**
PMT-1976 - Smart caching of historical objects - 21 ...	In Progress	Polarion 21 R1	3	0	**24d 7h**	**40d 6h**	**62 %**
DPP-214136 - Caching: Implement checker for data m...	Ready	3.21.1 (21 R1 -...	-	-	4d	0h	0 %
DPP-214350 - Caching: Add extra tests to avoid incorr...	Merged	3.21.1 (21 R1 -...	-	-	0h	21d 5h	100 %
DPP-214946 - Caching: Enable caching by default	In Progress	3.21.1 (21 R1 -...	-	-	20d 7h	19d 1h	47 %
PMT-1707 - Faster indexing of Test Runs with long ...	Ready	Polarion 21 R1	1	0	**15d**	**0h**	**0 %**
DPP-210177 - Indexing history of Test Run with thous...	Ready	3.21.1 (21 R1 -...	-	-	15d	0h	0 %
PMT-1848 - Improve indexing parallelization	Ready	Polarion 21 R1	1	0	**5d**	**0h**	**0 %**
DPP-210470 - Revisit max number of parallel workers f...	Ready	3.21.1 (21 R1 -...	-	-	5d	0h	0 %
PMT-1941 - Prevent superfluous commits to SVN	Draft	Polarion 21 R1	1	0	**9d**	**0h**	**0 %**
DPP-205518 - Superfluous changes to container objec...	Ready	3.21.1 (21 R1 -...	-	-	9d	0h	0 %
PMT-1933 - Address critical performance deficiencies of...	Targeted	Polarion 21 R1	4	0	**65d**	**0h**	**0 %**
PMT-1934 - Fast handling of jobs in the cluster	Draft	Polarion 21 R1	3	0	**35d**	**0h**	**0 %**
DPP-101371 - Clustering: Loading of jobs in Monitor t...	Ready	3.21.1 (21 R1 -...	-	-	15d	0h	0 %
DPP-211060 - Inefficient getting of import / export job...	Ready	3.21.1 (21 R1 -...	-	-	0h	0h	0 %
DPP-214902 - Clustering: Better filtering of the jobs in ...	Ready	3.21.1 (21 R1 -...	-	-	20d	0h	0 %
PMT-1935 - Active load-balancing in the cloud	Draft	Polarion 21 R1	1	0	**30d**	**0h**	**0 %**

Date

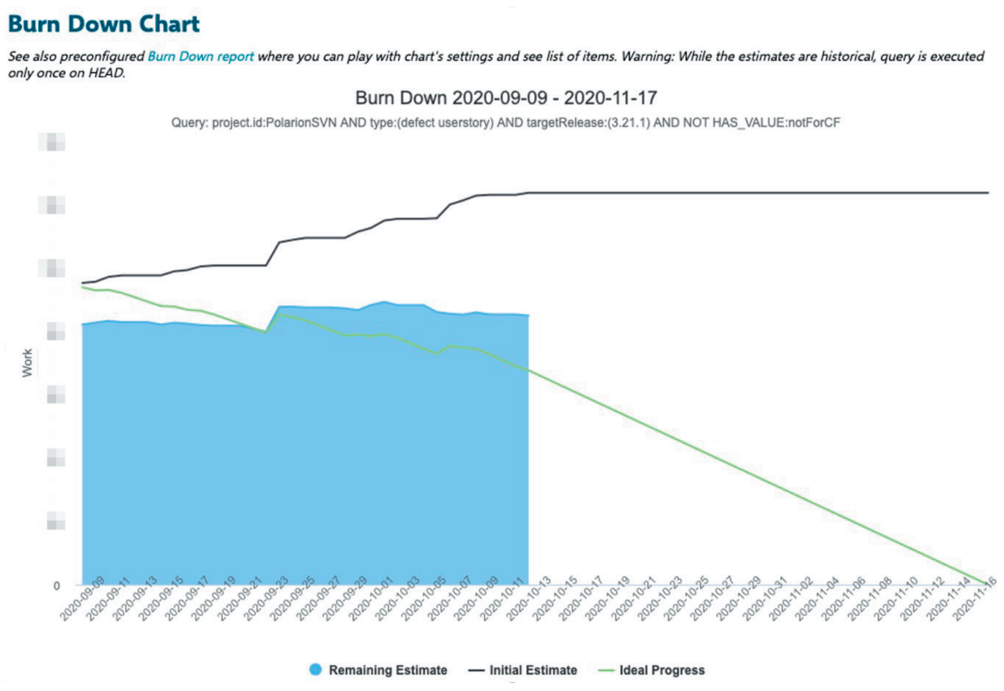
Code Freeze	2020-11-17
Dev. Release day	2020-12-02

Score Board

**RE/PBIs with target release 3.21.1*

	To Do	Total
Committed Capabilities:	2	2
Committed Features:	9	9
Committed PBIs:	45	48

Figure 4. Relevant activities for an upcoming PI for a Scrum team.


Burn Down Chart

See also preconfigured Burn Down report where you can play with chart's settings and see list of items. Warning: While the estimates are historical, query is executed only once on HEAD.

Figure 5. The corresponding execution progress can be monitored via a burn down chart.

The graph above reflects our level of agility. After starting a PI and planning and estimating, the estimates continue to change, usually rising. The gap between the remaining estimate and the ideal progress is expected because we only burn points after the planned PBIs are completely done. This usually takes a little time before it is reflected in the burn down chart.[3]

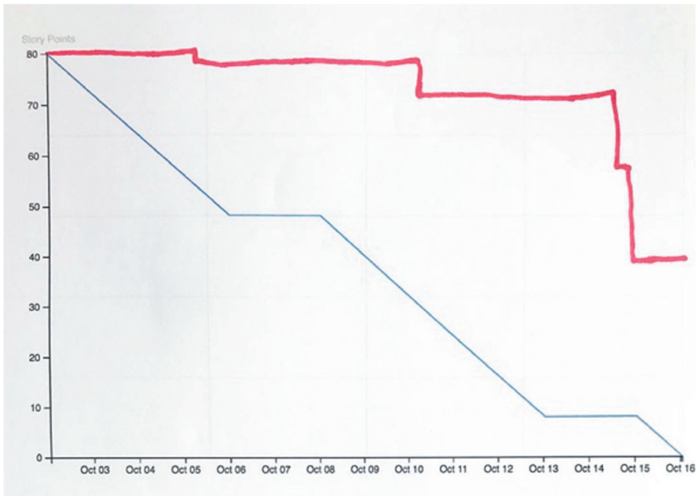
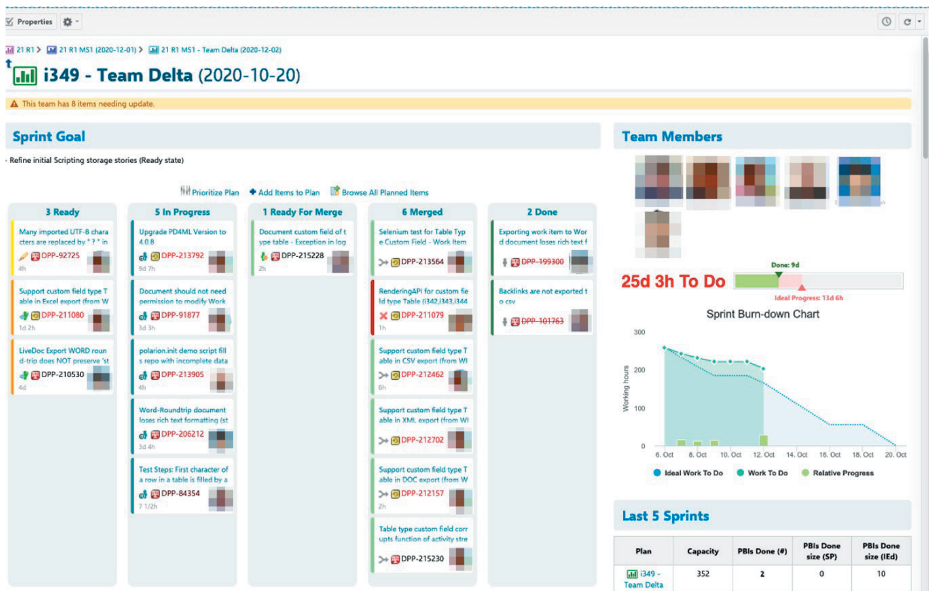


Figure 6. Possible burn down chart.



21 R1 > 21 R1 MS1 (2020-12-01) > 21 R1 MS1 - Team Delta (2020-12-02)

i349 - Team Delta (2020-10-20)

This team has 8 items needing update.

Sprint Goal

· Refine initial Scripting storage stories (Ready state)

3 Ready
- Many imported UTF-8 chara cters are replaced by " ? " in — DPP-92725 — 4h
- Support custom field type T able in Excel export (from W — DPP-211080 — 1d 2h
- LiveDoc Export WORD roun d-trip does NOT preserve 'st — DPP-210530 — 4d

5 In Progress
- Upgrade PD4ML Version to 4.0.8 — DPP-213792 — 9d 7h
- Document should not need permission to modify Work — DPP-91877 — 3d 3h
- polarion.init demo script fill s repo with incomplete data — DPP-213905 — 4h
- Word-Roundtrip document loses rich text formatting (st — DPP-206212 — 3d 4h
- Test Steps: First character of a row in a table is filled by a — DPP-84354 — 7 1/2h

1 Ready For Merge
- Document custom field of t ype table - Exception in log — DPP-215228 — 2h

6 Merged
- Selenium test for Table Typ e Custom Field - Work Item — DPP-213564
- RenderingAPI for custom fie ld type Table (i342,i343,i344 — DPP-211079 — 1h
- Support custom field type T able in CSV export (from WI — DPP-212462 — 6h
- Support custom field type T able in XML export (from WI — DPP-212702
- Support custom field type T able in DOC export (from W — DPP-212157 — 2h
- Table type custom field corr upts function of activity stre — DPP-215230

2 Done
- Exporting work item to Wor d document loses rich text f — ~~DPP-199300~~
- Backlinks are not exported t o csv — ~~DPP-101763~~

Team Members

25d 3h To Do

Done: 9d

Ideal Progress: 13d 6h

Last 5 Sprints

Plan	Capacity	PBIs Done (#)	PBIs Done size (SP)	PBIs Done size (IEd)
i349 - Team Delta	352	2	0	10

Figure 7. A concrete sprint.

Software development lifecycle

With Polarion, most of the code is written in Java, so we use the following integrated development environments (IDE) to code for it: Eclipse, IntelliJ and others. These IDEs are well integrated with the revision control systems (RCS) we use (GitLab) and enable additional functions like static code analysis or the execution of automated tests directly on newly written code.

Source code branching and committing

One of our most important best practices is to only make changes to the code base when there is a compelling reason to do so. These changes are always done via a PBI. When a user story or defect transitions to in progress a GitLab branch is automatically created as part of the workflow.

Polarion enables the automation of these procedures by defining workflow functions for the corresponding work item types.

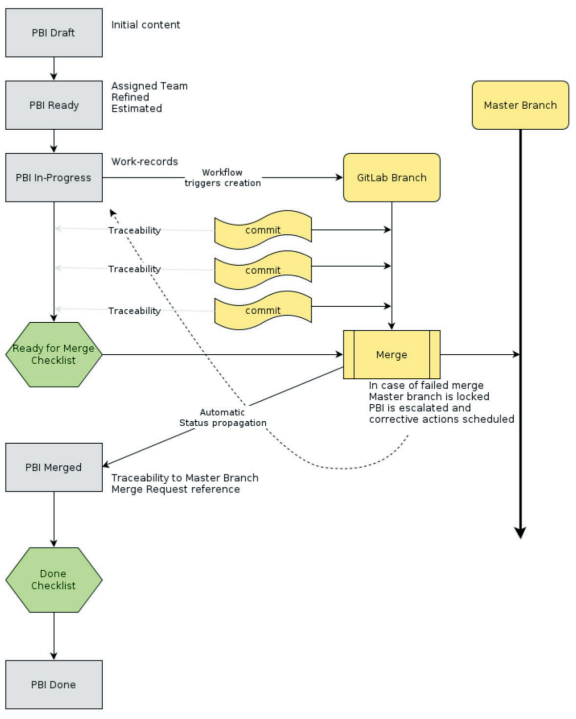
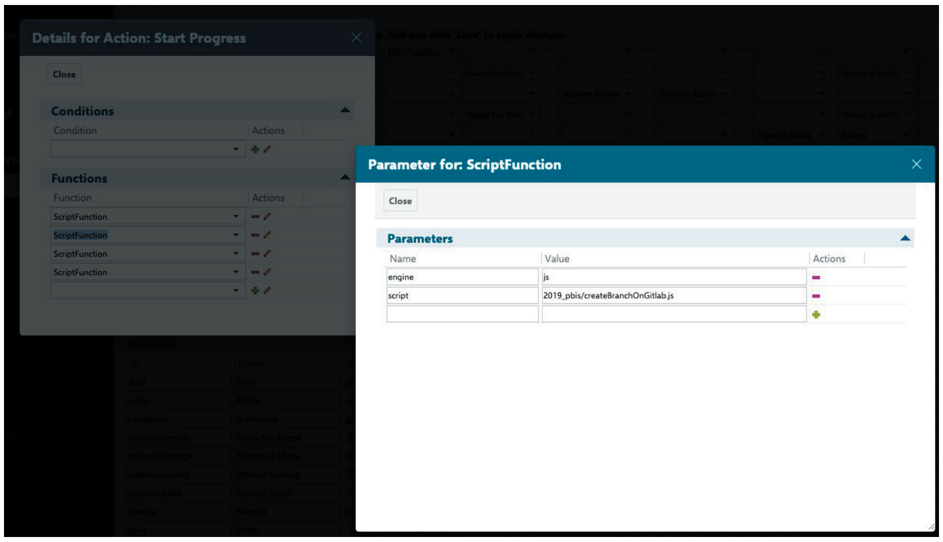


Figure 8. General PBI lifecycle.



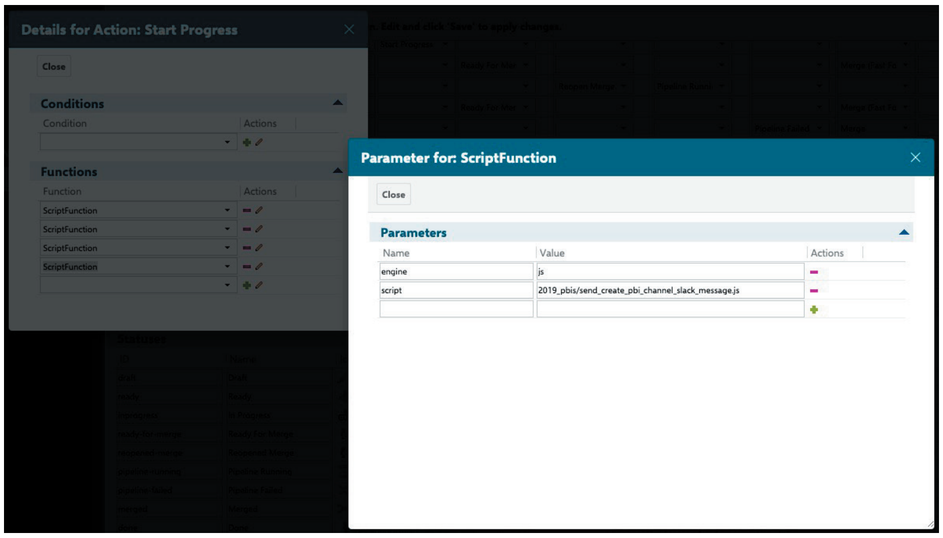
Parameter for: ScriptFunction

Parameters

Name	Value	Actions
engine	js	
script	2019_pbis/createBranchOnGitlab.js	

Figure 9. Registration of a script, which automatically creates a branch on GitLab as part of the product backlog item workflow execution.

To improve collaboration, we also automatically create Slack notifications so all team members are informed of an item's progress and can discuss issues and obstacles more organically in real time.

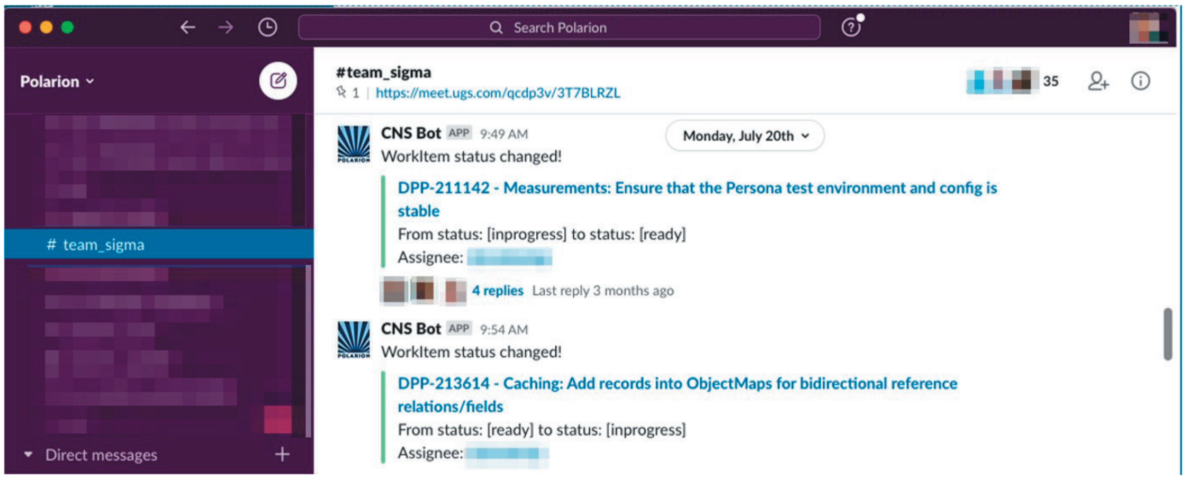


Parameter for: ScriptFunction

Parameters

Name	Value	Actions
engine	js	
script	2019_pbis/send_create_pbi_channel_slack_message.js	

Figure 10. Registration of a script, which notifies the team in a Slack channel about changes to the product backlog item.



#team_sigma
1 | https://meet.ugs.com/qcdp3v/3T7BLRZL

Monday, July 20th

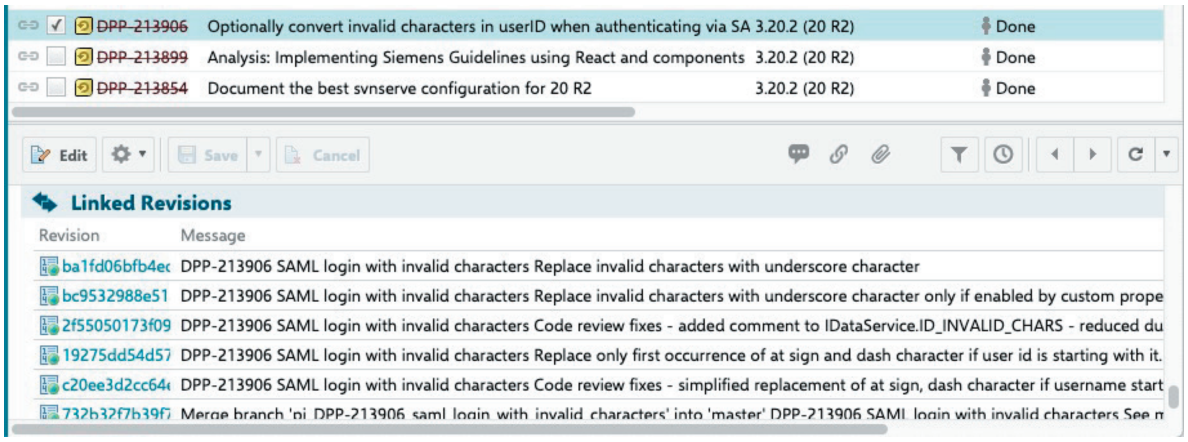
CNS Bot APP 9:49 AM
WorkItem status changed!
DPP-211142 - Measurements: Ensure that the Persona test environment and config is stable
From status: [inprogress] to status: [ready]
Assignee: [redacted]
4 replies Last reply 3 months ago

CNS Bot APP 9:54 AM
WorkItem status changed!
DPP-213614 - Caching: Add records into ObjectMaps for bidirectional reference relations/fields
From status: [ready] to status: [inprogress]
Assignee: [redacted]

Figure 11. The Slack interface.

When a developer is ready to make changes to a GitLab branch, they include the PBI's work item ID in the Git commit message. All changes are linked to the item that prompted them.

The system of IDs works differently in Polarion compared to similar tools. The prefix identifies the project where the work item is stored. DPP is a prefix for items in our production project. The second part assigns a numeric identifier that is unique to the project.



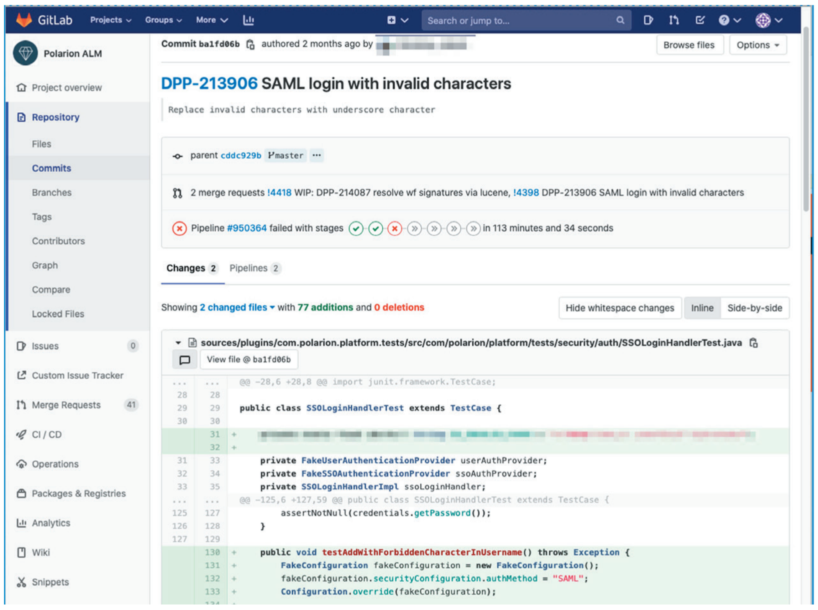
~~DPP-213906~~	Optionally convert invalid characters in userID when authenticating via SA	3.20.2 (20 R2)	Done
~~DPP-213899~~	Analysis: Implementing Siemens Guidelines using React and components	3.20.2 (20 R2)	Done
~~DPP-213854~~	Document the best svnserve configuration for 20 R2	3.20.2 (20 R2)	Done

Linked Revisions

Revision	Message
ba1fd06bfb4ec	DPP-213906 SAML login with invalid characters Replace invalid characters with underscore character
bc9532988e51	DPP-213906 SAML login with invalid characters Replace invalid characters with underscore character only if enabled by custom prope
2f55050173f09	DPP-213906 SAML login with invalid characters Code review fixes - added comment to IDataService.ID_INVALID_CHARS - reduced du
19275dd54d57	DPP-213906 SAML login with invalid characters Replace only first occurrence of at sign and dash character if user id is starting with it.
c20ee3d2cc64	DPP-213906 SAML login with invalid characters Code review fixes - simplified replacement of at sign, dash character if username start
732b32f7b39f	Merge branch 'pi_DPP-213906_saml_login_with_invalid_characters' into 'master' DPP-213906 SAML login with invalid characters See m

Figure 12. Example of commits tied to a specific work item.

Clicking on the linked revision will open the GitLab UI and display the changes.



Commit ba1fd06b authored 2 months ago by [redacted]

DPP-213906 SAML login with invalid characters

```
Replace invalid characters with underscore character
```

parent cddc929b master

2 merge requests !4418 WIP: DPP-214087 resolve wf signatures via lucene, !4398 DPP-213906 SAML login with invalid characters

Pipeline #950364 failed with stages in 113 minutes and 34 seconds

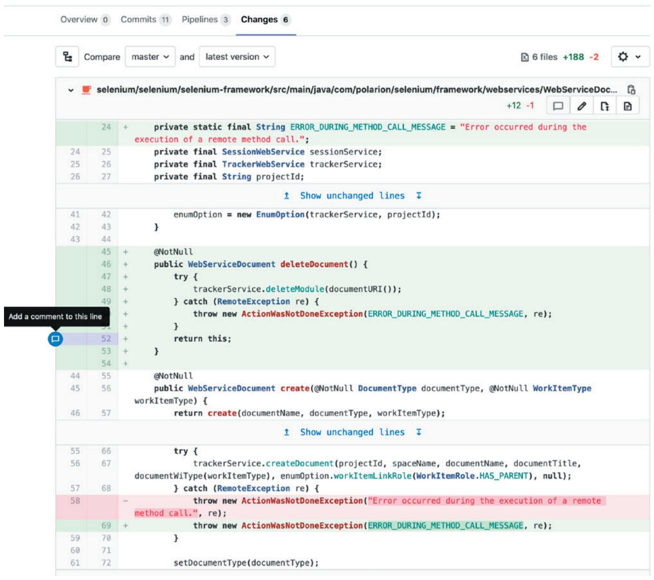
Showing 2 changed files with 77 additions and 0 deletions

sources/plugins/com.polarion.platform.tests/src/com/polarion/platform/tests/security/auth/SSOLoginHandlerTest.java

```
@@ -28,6 +28,8 @@ import junit.framework.TestCase;

public class SSOLoginHandlerTest extends TestCase {

+    [redacted]
+
    private FakeUserAuthenticationProvider userAuthProvider;
    private FakeSSOAuthenticationProvider ssoAuthProvider;
    private SSOLoginHandlerImpl ssoLoginHandler;
@@ -125,6 +127,59 @@ public class SSOLoginHandlerTest extends TestCase {
        assertNotNull(credentials.getPassword());
    }

+    public void testAddWithForbiddenCharacterInUsername() throws Exception {
+        FakeConfiguration fakeConfiguration = new FakeConfiguration();
+        fakeConfiguration.securityConfiguration.authMethod = "SAML";
+        Configuration.override(fakeConfiguration);
```

Figure 13. The review of changes in GitLab.

With the review of changes in GitLab, this facilitates the code review process.

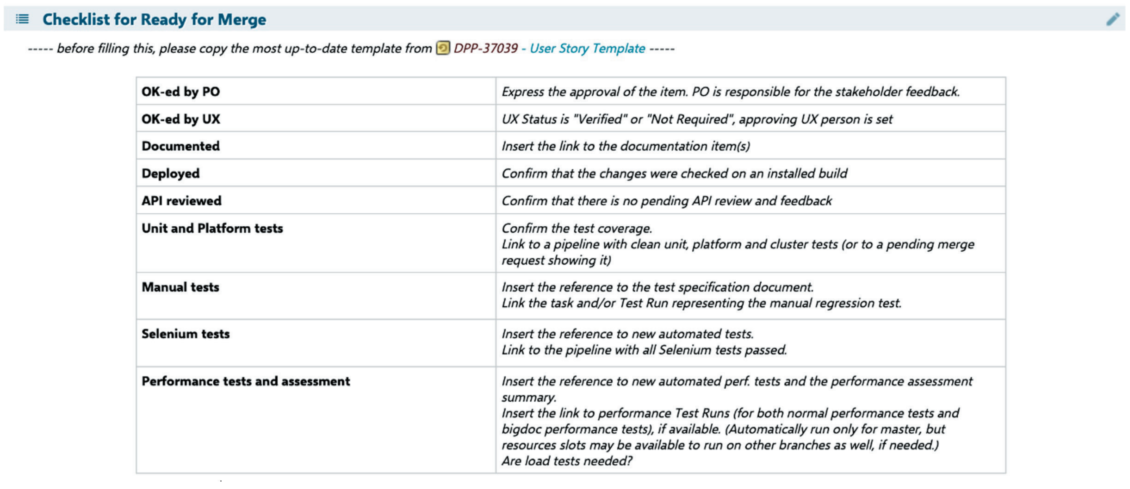


selenium/selenium/selenium-framework/src/main/java/com/polarion/selenium/framework/webservices/WebServiceDoc...

```
+    private static final String ERROR_DURING_METHOD_CALL_MESSAGE = "Error occurred during the
execution of a remote method call.";
     private final SessionWebService sessionService;
     private final TrackerWebService trackerService;
     private final String projectId;
                         Show unchanged lines
         enumOption = new EnumOption(trackerService, projectId);
     }

+    @NotNull
+    public WebServiceDocument deleteDocument() {
+        try {
+            trackerService.deleteModule(documentURI());
+        } catch (RemoteException re) {
+            throw new ActionWasNotDoneException(ERROR_DURING_METHOD_CALL_MESSAGE, re);
+        }
+        return this;
+    }
+
     @NotNull
     public WebServiceDocument create(@NotNull DocumentType documentType, @NotNull WorkItemType
workItemType) {
         return create(documentName, documentType, workItemType);
                         Show unchanged lines
         try {
             trackerService.createDocument(projectId, spaceName, documentName, documentTitle,
documentWiType(workItemType), enumOption.workItemLinkRole(WorkItemRole.HAS_PARENT), null);
         } catch (RemoteException re) {
-            throw new ActionWasNotDoneException("Error occurred during the execution of a remote
method call.", re);
+            throw new ActionWasNotDoneException(ERROR_DURING_METHOD_CALL_MESSAGE, re);
         }

         setDocumentType(documentType);
```

Figure 14. The code review process.

Commenting on a change starts a discussion. All discussions must be resolved before a PBI’s status can be changed to ready for merge.

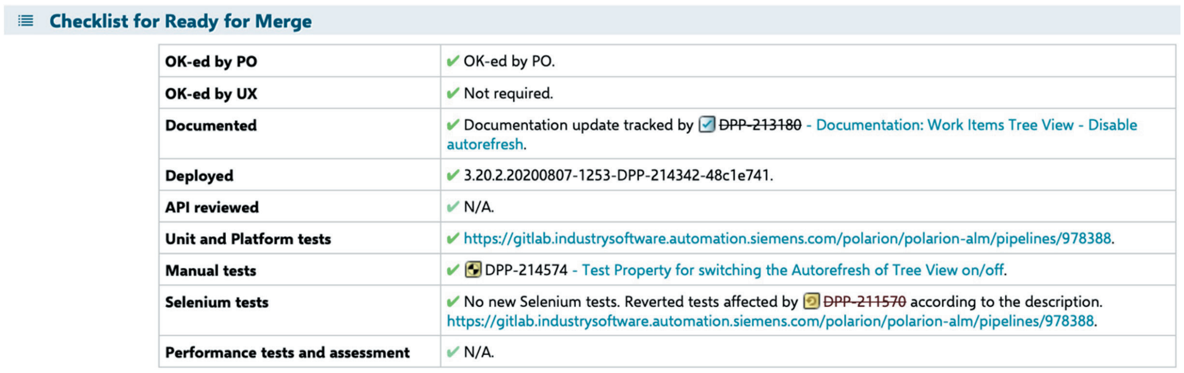


Checklist for Ready for Merge

----- before filling this, please copy the most up-to-date template from DPP-37039 - User Story Template -----

OK-ed by PO	*Express the approval of the item. PO is responsible for the stakeholder feedback.*
OK-ed by UX	*UX Status is "Verified" or "Not Required", approving UX person is set*
Documented	*Insert the link to the documentation item(s)*
Deployed	*Confirm that the changes were checked on an installed build*
API reviewed	*Confirm that there is no pending API review and feedback*
Unit and Platform tests	*Confirm the test coverage.* *Link to a pipeline with clean unit, platform and cluster tests (or to a pending merge request showing it)*
Manual tests	*Insert the reference to the test specification document.* *Link the task and/or Test Run representing the manual regression test.*
Selenium tests	*Insert the reference to new automated tests.* *Link to the pipeline with all Selenium tests passed.*
Performance tests and assessment	*Insert the reference to new automated perf. tests and the performance assessment summary.* *Insert the link to performance Test Runs (for both normal performance tests and bigdoc performance tests), if available. (Automatically run only for master, but resources slots may be available to run on other branches as well, if needed.)* *Are load tests needed?*

Figure 15. Template for the ready for merge checklist.



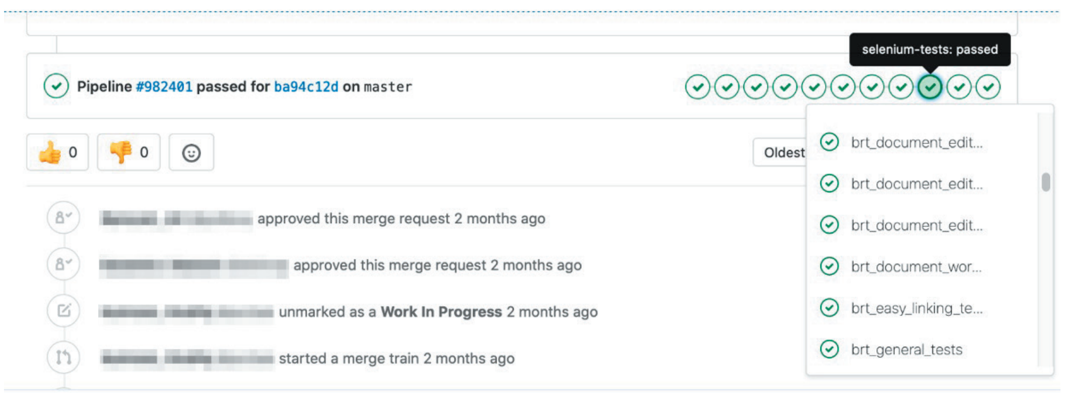
Checklist for Ready for Merge

OK-ed by PO	✓ OK-ed by PO.
OK-ed by UX	✓ Not required.
Documented	✓ Documentation update tracked by ~~DPP-213180~~ - Documentation: Work Items Tree View - Disable autorefresh.
Deployed	✓ 3.20.2.20200807-1253-DPP-214342-48c1e741.
API reviewed	✓ N/A.
Unit and Platform tests	✓ https://gitlab.industrysoftware.automation.siemens.com/polarion/polarion-alm/pipelines/978388.
Manual tests	✓ DPP-214574 - Test Property for switching the Autorefresh of Tree View on/off.
Selenium tests	✓ No new Selenium tests. Reverted tests affected by ~~DPP-211570~~ according to the description. https://gitlab.industrysoftware.automation.siemens.com/polarion/polarion-alm/pipelines/978388.
Performance tests and assessment	✓ N/A.

Figure 16. Example of the populated table.

Whenever an item is marked as ready to merge, the responsible engineer can trigger a merge pipeline. The steps are as follows:

1. Integrate the change to the master branch.
2. Compile sources and prepare binaries.
3. Run unit-tests, application programming interface (API), free and open-source software (FOSS) and other checks.
4. Deploy the binaries to a test environment.
5. Run UI-test suits on the environment.
6. Run load, stress and performance tests on a reference environment.
7. Collect the results of all test runs and report back to Polarion.
8. Prepare a shippable package.



selenium-tests: passed

Pipeline #982401 passed for ba94c12d on master

0 0

Oldest

brt_document_edit...
brt_document_edit...
brt_document_edit...
brt_document_wor...
brt_easy_linking_te...
brt_general_tests

[redacted] approved this merge request 2 months ago

[redacted] approved this merge request 2 months ago

[redacted] unmarked as a **Work In Progress** 2 months ago

[redacted] started a merge train 2 months ago

Figure 17. The pipeline execution status in GitLab.

The merge request is expected to be successful. Before it gets added to the master, the team must run the same compilation, unit and UI tests on their local branches and team servers.

One aspect that requires special attention is performance tests. They are executed on a reference environment that often differs from the development environment. As a result, performance tests that succeed on the local branch may fail on the master.

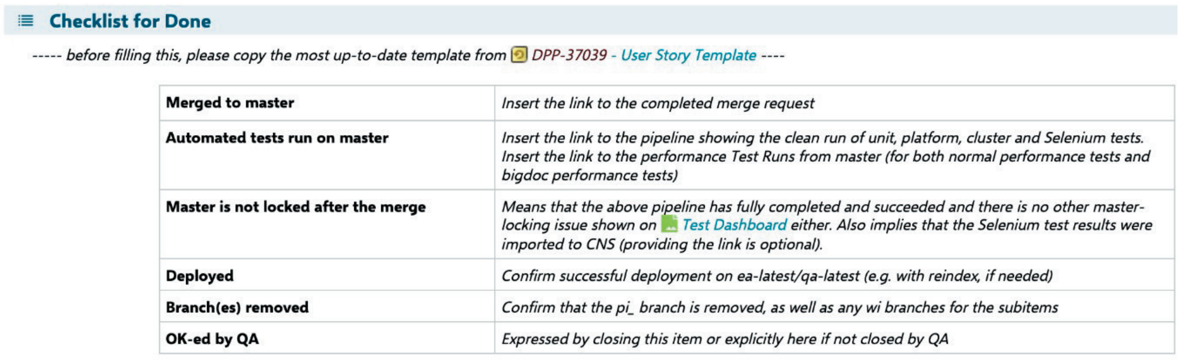
What we do when this happens

Before a PBI can be closed, we confirm there are no regressions and make sure the pipeline's performance tests pass. If any suspects are identified, we lock the master branch and no new commits are allowed until the situation is clarified. From here, we identify if it is a temporary outage, a side effect of something on the test server or a genuine regression. The team that created the suspicious merge request makes it their priority to address the problem, even if it means rolling back the commit. Then the master branch is unlocked and further commits are allowed.

Pipeline types that facilitate the different product-life-cycle phases

- Master — runs on every push to the master, all tests, distributions, installers and dockers.
- Release — runs when any tag is created. Runs all tests, distributions, installers, dockers and packages the release.
- Post-release — runs on every push to the release branch. Runs unit and platform tests, all distributions, installers and dockers.
- PI merge request — runs on every push to a PI branch with an open merge request, runs consolidated stage which is the same as consolidated custom pipeline, but each job is executed only if there are relevant changes.
- Custom — teams can create pipelines on-demand with custom parameters.

After a successful merge to the master, the following checklist must be filled out in the user story.

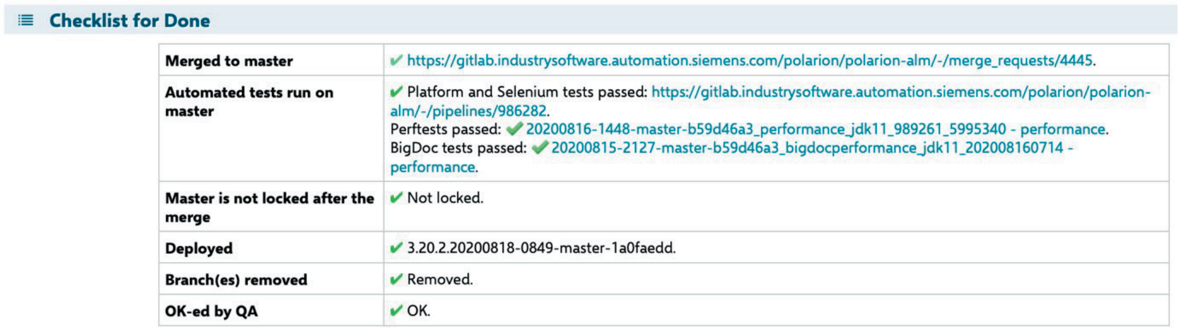


Checklist for Done

----- before filling this, please copy the most up-to-date template from DPP-37039 - User Story Template ----

Merged to master	*Insert the link to the completed merge request*
Automated tests run on master	*Insert the link to the pipeline showing the clean run of unit, platform, cluster and Selenium tests. Insert the link to the performance Test Runs from master (for both normal performance tests and bigdoc performance tests)*
Master is not locked after the merge	*Means that the above pipeline has fully completed and succeeded and there is no other master-locking issue shown on Test Dashboard either. Also implies that the Selenium test results were imported to CNS (providing the link is optional).*
Deployed	*Confirm successful deployment on ea-latest/qa-latest (e.g. with reindex, if needed)*
Branch(es) removed	*Confirm that the pi_ branch is removed, as well as any wi branches for the subitems*
OK-ed by QA	*Expressed by closing this item or explicitly here if not closed by QA*

Figure 18. An example of a completed checklist (A).



Checklist for Done

Merged to master	✓ https://gitlab.industrysoftware.automation.siemens.com/polarion/polarion-alm/-/merge_requests/4445.
Automated tests run on master	✓ Platform and Selenium tests passed: https://gitlab.industrysoftware.automation.siemens.com/polarion/polarion-alm/-/pipelines/986282. Perftests passed: ✓ 20200816-1448-master-b59d46a3_performance_jdk11_989261_5995340 - performance. BigDoc tests passed: ✓ 20200815-2127-master-b59d46a3_bigdocperformance_jdk11_202008160714 - performance.
Master is not locked after the merge	✓ Not locked.
Deployed	✓ 3.20.2.20200818-0849-master-1a0faedd.
Branch(es) removed	✓ Removed.
OK-ed by QA	✓ OK.

Figure 19. An example of a completed checklist (B).

Continuous deployment

Whenever a commit happens on a branch, the CD is configured to grab the results and deploy them to a server for debugging, testing, reference implementation or deploy the master branch results to an internal production environment. This helps us complete the first level of testing in a practical environment before customers see it. We have been doing this method called dogfooding with Polarion from the start.[4]

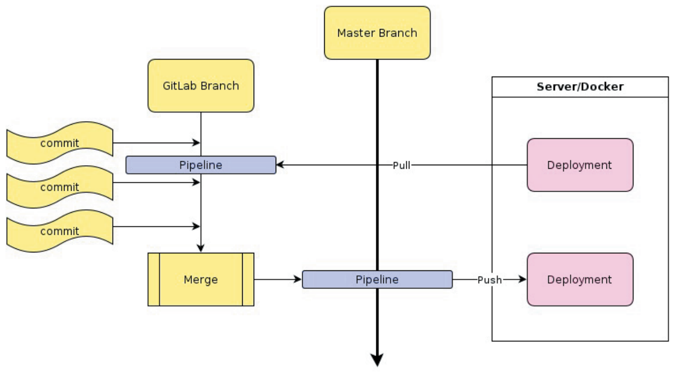
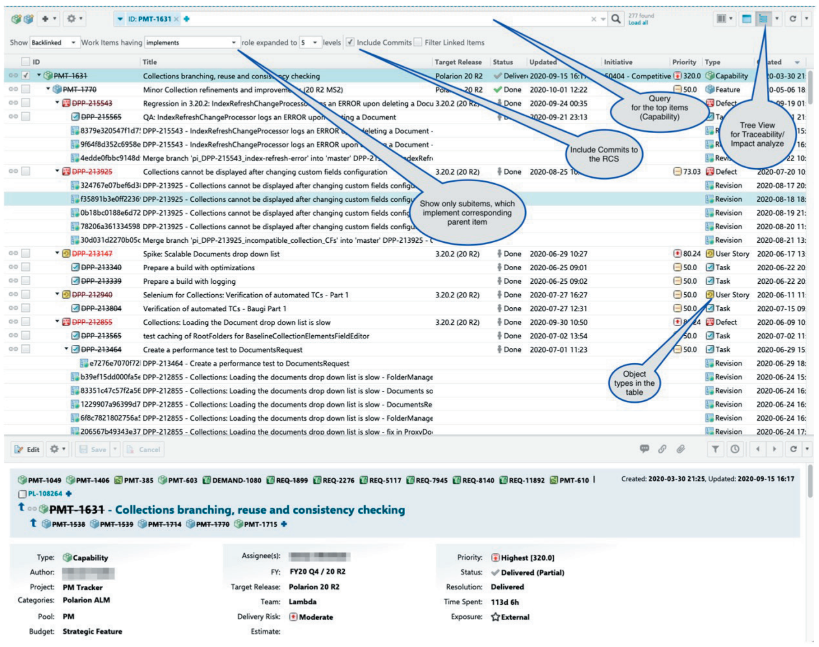


Figure 20. Lifecycle of a product backlog item in a branch of GitLab, with continuous deployment to a test environment.

A continuous deployment of the master branch is always desired. This means its pipeline ends with 100 percent positive results. However, local branches may depend on the status of the development cycle. For example, a team may wish to have a solution running and testable after each commit, so they configure one of the local pipelines to compile and immediately deploy to a team server. They may instead opt for a daily deployment model where the sever pulls the last available results overnight and deploys it for use the following day. They can always start a pipeline manually or request a new deployment via a command-line script (even to a different server or a container).

Traceability and impact analysis

Polarion offers an easy way to check the impact and traceability information on how, for example, a capability is implemented and/or tested.


ID: PMT-1631
277 found
Load all
Query for the top items (Capability)
Tree View for Traceability/ Impact analyze
Include Commits to the RCS
Show only subitems, which implement corresponding parent item
Object types in the table
Edit
Save
Cancel

Show Backlinked Work Items having implements role expanded to 5 levels ✓ Include Commits ☐ Filter Linked Items

ID	Title	Target Release	Status	Updated	Initiative	Priority	Type	ated
~~PMT-1631~~	Collections branching, reuse and consis ncy checking	Polarion 20 R2	Deliver	2020-09-15 16:1	0404 - Competitive	320.0	Capability	0-03-30 21
~~PMT-1770~~	Minor Collection refinements and improveme (20 R2 MS2)	Pola 20 R2	✓ Done	2020-10-01 12:22		50.0	Feature	0-05-06 18
~~DPP-215543~~	Regression in 3.20.2: IndexRefreshChangeProcesso s an ERROR upon deleting a Docu	3.20.2 (20 R2)	Done	2020-09-24 00:35			Defect	09-19 01
~~DPP-215565~~	QA: IndexRefreshChangeProcessor logs an ERROR upon ting a Document			2020-09-21 23:13			Ta	1 21
8379e320547f1d7	DPP-215543 - IndexRefreshChangeProcessor logs an ERROR eleting a Document -						R	15
9f64f8d352c6958e	DPP-215543 - IndexRefreshChangeProcessor logs an ERROR upon a Document -						R	16
4edde0fbbc9148d	Merge branch 'pi_DPP-215543_index-refresh-error' into 'master' DPP-21 dexRefr						Revi	22 10
~~DPP-213925~~	Collections cannot be displayed after changing custom fields configuration	3.20.2 (20 R2)	Done	2020-08-25 10		73.03	Defect	2020-07-20 10
324767e07bef6d3	DPP-213925 - Collections cannot be displayed after changing custom fields configu						Revision	2020-08-17 20
f35891b3e0ff2236	DPP-213925 - Collections cannot be displayed after changing custom fields configu						Revision	2020-08-18 18
0b18bc0188e6d72	DPP-213925 - Collections cannot be displayed after changing custom fields config						Revision	2020-08-19 21
78206a361334598	DPP-213925 - Collections cannot be displayed after changing custom fields configu						Revision	2020-08-20 11
30d031d2270b05c	Merge branch 'pi_DPP-213925_incompatible_collection_CFs' into 'master' DPP-213925 -						Revision	2020-08-21 13
~~DPP-213147~~	Spike: Scalable Documents drop down list	3.20.2 (20 R2)	Done	2020-06-29 10:27		80.24	User Story	2020-06-17 13
~~DPP-213340~~	Prepare a build with optimizations		Done	2020-06-25 09:01		50.0	Task	2020-06-22 20
~~DPP-213339~~	Prepare a build with logging		Done	2020-06-25 09:02		50.0	Task	2020-06-22 20
~~DPP-212940~~	Selenium for Collections: Verification of automated TCs - Part 1	3.20.2 (20 R2)	Done	2020-07-27 16:27		50.0	User Story	2020-06-11 11
~~DPP-213804~~	Verification of automated TCs - Baugi Part 1		Done	2020-07-27 12:31		50.0	Task	2020-07-15 09
~~DPP-212855~~	Collections: Loading the Document drop down list is slow	3.20.2 (20 R2)	Done	2020-09-30 10:50		8 24	Defect	2020-06-09 10
~~DPP-213565~~	test caching of RootFolders for BaselineCollectionElementsFieldEditor		Done	2020-07-02 13:54		50.0	Task	2020-07-02 11
~~DPP-213464~~	Create a performance test to DocumentsRequest		Done	2020-07-01 11:23		50.0	Task	2020-06-29 15
e7276e7070f72	DPP-213464 - Create a performance test to DocumentsRequest						Revision	2020-06-29 18
b39ef15dd000fa5	DPP-212855 - Collections: Loading the documents drop down list is slow - FolderManage						Revision	2020-06-24 15
83351c47c57f2a56	DPP-212855 - Collections: Loading the documents drop down list is slow - Documents so						Revision	2020-06-24 16
1229907a96399d7	DPP-212855 - Collections: Loading the documents drop down list is slow - DocumentsRe						Revision	2020-06-24 16
6f8c7821802756a	DPP-212855 - Collections: Loading the documents drop down list is slow - FolderManage						Revision	2020-06-24 16
206567b49343e37	DPP-212855 - Collections: Loading the documents drop down list is slow - fix in ProxyDo						Revision	2020-06-24 17

~~PMT-1049~~ ~~PMT-1406~~ PMT-385 PMT-603 DEMAND-1080 ~~REQ-1899~~ REQ-2276 REQ-5117 REQ-7945 REQ-8140 REQ-11892 PMT-610 | Created: **2020-03-30 21:25**, Updated: **2020-09-15 16:17**

PL-108264

~~PMT-1631~~ - Collections branching, reuse and consistency checking

~~PMT-1538~~ ~~PMT-1539~~ ~~PMT-1714~~ ~~PMT-1770~~ PMT-1715

Type:	**Capability**	Assignee(s):	[redacted]	Priority:	**Highest [320.0]**
Author:	[redacted]	FY:	**FY20 Q4 / 20 R2**	Status:	**Delivered (Partial)**
Project:	**PM Tracker**	Target Release:	**Polarion 20 R2**	Resolution:	**Delivered**
Categories:	**Polarion ALM**	Team:	**Lambda**	Time Spent:	**113d 6h**
Pool:	**PM**	Delivery Risk:	**Moderate**	Exposure:	**External**
Budget:	**Strategic Feature**	Estimate:			

Figure 21. Polarion's break-down structure of a capability. It is split into product backlog items, defects, tasks and commits to the source control management.

Conclusion

This white paper lays out our best practices in Polarion development while showing examples of how DevOps can be a natural component of ALM. If you have ideas, questions or would like to share your experience, go to the Polarion community site to access articles and an open discussion board.

References

1. https://www.scaledagileframework.com/glossary
2. https://www.scrum.org/resources/blog/user-story-or-stakeholder-story
3. https://dzone.com/articles/the-ideal-burn-down-chart
4. https://en.wikipedia.org/wiki/Eating_your_own_dog_food