evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
UNIVERSITAS MASARYKIANA BRUNENSIS
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Platforma průmyslové spolupráce

CZ.1.07/2.4.00/17.0041

Název

Modern open source Java EE-based process and issue tracker

Popis a využití

- systémy pro sledování problémů
- analýza a srovnání systémů a design nového systému pro sledování problémů a procesů
- studijní materiál pro systémy pro sledování problémů (issue tracking system)
- výuka: pokročilá Java

Jazyk textu

- anglický

Autor (autoři)

- Monika Gottvaldová

Oficiální stránka projektu:

- http://lasaris.fi.muni.cz/pps

Dostupnost výukových materiálů a nástrojů online:

- http://lasaris.fi.muni.cz/pps/study-materials-and-tools

Contents

1 Introduction

In all companies, employees work on some tasks, cooperate within the team and go through certain processes, making it advantageous to track their work. Many systems enabling such a tracking can be used by these companies, and are called 'issue tracking systems'. Users are able to track their tasks, share them with other team members, add intended details, attachments, set defining attributes, set the state of the issue regarding the performed action etc. Many companies want to use the system not only for task tracking, but also for bug reporting during the test phase of a product, for project management and arbitrary company corporate processes, and want the systems to be very customizable.

Comparison of few issue tracking systems helps to define the features, which the system should include. The defining quality is customization - features should be customizable, and it is therefore an advantage for the system to be created as open source using Java EE technologies for the developers to customize the code. Systems should be also easy to use and have intuitive and user friendly graphical user interface (GUI). They should be applicable to numerous processes and usable by many companies to map these processes.

This thesis describes a number of processes in Red Hat and their mapping in Bugzilla, which in turn describes a detailed issue lifecycle and its connection to real work. There are differences in all the processes, so it is paramount for the system users to be able to adapt the system to their needs. Also, Red Hat has its own hiring process, for which this system can be used as well.

Another objective is to design a system that satisfies many problems. Wicket, CDI and JPA are used for the implementation, Wicket and CDI for the functionality and JPA for the data model. The usage of Wicket is very easy and intuitive, this framework enables the creation of functional classes and their HTML pages, both in object-oriented way. It is extendable and reusable because of usage of components that can be easily added into the code. The JPA ensures the mapping of the objects to the database tables. The system interface is designed to be user friendly and intuitive and the possible user actions have to be restricted by roles and permissions, which allows the users to be sure what they can do.

There are many plans for future development to include more advantages in the system, such as the possibility to add plugins, which would support the adaptability, full text search for more effective searching, cus-

tom stream dashbord to enhance the team cooperation, time management feature to track the time spent on the task and integration with many other exsisting tools.

2 Issue Tracking Systems

An issue tracking system is an important software used by many companies. Is it used mostly for its benefits of easy management and issue prioritization, which enables effective work planning and valuable input when defining the product road map, simple team cooperation and communication and great project organization and clear overview to generate a report from.

There are several ways to use this tool. The main usage is to track, manage and maintain lists of issues reported and being watched by employees or customers. Employees working on a project can track their own tasks, or tasks of a team, they can share them, comment on them, provide appropriate attributes and details and review the progress of the task and its state at their leisure. Yet another possibility is the usage by the Quality Engineering (QE) team. The responsibility of the QE team is to test a project release and report bugs for the tested project into the system. These bugs are then tracked by management and developers and subsequently solved. This system can be also called 'bug tracking system'. The usage of issue tracking systems for customer support call centers, help desks or support teams is also widespread. Customers, end users or employees in the support teams create tickets for the issues containing the description, steps to reproduce the issue and most important information and the tickets are then solved by the support team. The support team can quickly locate the end user through the ticket and has all the information in one place, simplifying the work. [36]

The tracking systems don't have to be mapped to issues and tasks exclusively, but also to various company processes involving job applications, tracking individuals and their progress of seeking employment, and also management processes and everything that goes through a sequence of events or activities in general.

Issues

All the units that are being tracked in the system, e.g. bugs, tasks, processes etc. are called 'issues'. The issue itself has several attributes, which need to be maintained. It is important to create a short and apposite summary of the problem and explain every known detail in the description along with an eventual workaround. There are many fields in the issue. One field reflects the severity and overall importance of the issue. The severity level can be e.g. minor, medium, or major. Also included are the priority, type

of the issue, date of creation, product and its version, and others relevant to the type of the issue. Each issue maintains the history of every change. The issues are also passing through workflow during their lifecycles, the workflow being posibly unique for every project.

Workflow
Workflow is an important part of every tracking system and every issue has to go through it. Workflow consists of several connected steps, the order of which depends on the circumstances of the issue and the workflow's settings.

Basically, it is a set of statuses and transitions the issue goes through during its lifecycle. Workflows typically represent business processes. Status represents a state of the issue at a specific point within the workflow. An issue can only be in one state at a time. The transitions represents a passage from one status to another and the name of the transition should reflect the action of passing the issue from one state to another. Only when a transition in workflow between two statuses has been created can the issue be shifted from the first to the second state. The issue can only be reversed to the former status if another transition from the second status to the first exists. Every state represents a certain operation or action to be accomplished by an individual and the move represents an end of one specific action and the beginning of another. Creating a new workflow means creating a set of transitions between pairs of statuses.

Existing Issue Tracking Systems
There is an abundance of issue tracking systems currently in existence, each with its advantages and disadvantages. Here is the list of some of them and their comparison.

2.1 Bugzilla

Bugzilla is a bug tracking system allowing to effectively keep track of outstanding bugs in a product. It is free but has many good features and became very popular in the industry. The bugs and code changes can be tracked by the system, which also allows an effective communication within the team, patch-review and submission and quality assurance management and organization. Bugzilla is still being developed and tested and the users are allowed to report bugs, which are then followed up by a dedicated team. Bugzilla has been successfully installed under Solaris, Linux, and Win32. It

is written in Perl, therefore requiring the user to run Perl on his machine. It uses MySQL as a back-end. Bugzilla is currently used by 1268 companies, organizations, and projects.[1]

Advantages Bugzilla is used by many companies, it is free and open source, it is extendable, and there are many plugins to add. The GUI is arranged well, contains crucial links for the search and creation of issues in the upper head alongside browsing the projects details or user info. That similar bugs can be found before the insertion to prevent a duplication is another good feature. When submitting a new issue a possibility exists to limit the visibility to special groups of users. A possibility to connect an issue with another one from a different bug tracker is also in place. Time tracking can be used to set the deadline, hours spent on the problem, and hours left.

Disadvantages Bugzilla is designed exclusively as a bug tracker. However, the Bugzilla code also has potential to be used as a ticket system, task and project management tool, and maybe such a plan exists for the future. The existence of a customizable workflow is its clear advantage, it consists of a list of statuses that can be edited, deleted, or created. Unfortunately, when updating the bug, it offers all the statuses, not only those that are possible to be moved to based on the workflow. So it is only described in some manual how the workflow should work. A possibility to add custom fields is also present, but only within the scope of the whole company, not specific to the projects. The GUI is quite complicated and contains a profusion of fields and flags in the issue details, also present when searching for an issue, and is not very intuitive to use.

Security The Bugzilla team pays a lot attention to security. Perl's "taint" mode prevents SQL Injection, and has a very careful system in place to prevent Cross Site Scripting. Bugzilla patches the security vulnerabilities ongoingly and the system is designed to be really secured at all times. It is also very careful concerning information leaks, it enables to use groups when a user wants to add a comment or attachment, it can be viewed only by a certain group and nobody else receives the information. It is protected from spammers because email addresses of users are not available to logged out users. Aaccounts are also locked after a few attempts of guessing a password.[2]

Purchase Bugzilla is free and open source and doesn't offer any form of

commercial support. However, there are public forums to request assistance publicly so that the information is available to others as well. Additionally, the team members can be asked by email directly. There are also contractors and businesses who have declared themselves available for Bugzilla consulting work and users can purchase their support.[3]

2.2 Trac

Trac is a web-based, open source software written in Python. Its back-end database can be SQLite, PostgreSQL or MySQL. Apart from issue tracking, it also provides wiki and integration to subversion and git, or other version-control system. Trac uses a minimalistic approach to web-based software project management. Trac allows wiki markup in issue descriptions and commit messages, users are enabled to create links and references between bugs, tasks, files and wiki pages. There is a timeline showing all current and past project events in order, which enables an elegant overview of the projects and of the system.[4] The system is being used by about 217 companies or projects listed on the Trac page.

Advantages Trac is not only a bug tracker, but can also be used for project tasks, feature requests or software support issues, and contains a Wiki. The integration with SCM systems is part of the tool, so users can browse the code. Track is a pluggable system with many available plugins with several features, e.g. for Concurrent Versions System, timing and estimation, plugin supporting multiple user projects and so on.[5] There are customizable workflows, the user can add a transition with the beginning and end states and add resolution to the transitions by editing one of the existing worflows or choose from the list. New issue is created in a status *new* and when updating, only possible workflow statuses are offered.

Disadvantages Workflows are customizable, but one workflow is then used for the whole system. Trac basically doesn't support multiple projects, although there's a plugin for it, but this functionality should be default behavior. There are more features that should also be part of the system. The GUI is easy and intuitive, but not too many features are included. The workflow when updating an issue is not very user friendly as well. When an issue is created for the first time, the status is set to *new*, when updating, only statuses with possible transitions

are presented, which is good. But instead of a dropdown, there are few checkboxes to choose a state from by an action name, which is not very intuitive.

Security Trac uses a simple, case sensitive permission system to control what users can and cannot access. Non-authenticated users accessing the system get the permissions set for the user called 'Anonymous'. There are two groups which have a list of permissions: 'anonymous' and 'authenticated' and permissions can be grouped together to form roles.[6]

Purchase Trac is free and open source. There is a TracWiki page for users to contribute to and participate by reporting bugs, asking questions, suggesting improvements or discussing the solutions.

2.3 JIRA

JIRA is a tracking system for project management, bug tracking, tasks and issue tracking, work assigning and team activity followup. Users can prioritize their work. It supports code integration and enables to keep track of important issues.[7] It is written in Java. MySQL, PostgreSQL, Oracle, and SQL Server can be used as a back-end. It is not open source.[8] JIRA is created by Atlassian and there are about 450 companies or projects using this issue tracker.

Advantages JIRA is not only a bug tracking system but can also be used for task tracking, project management and the service desk. It contains code integration for a version control system like Git, Subversion and others, where user can view a code changes or progress. Custom workflow can be used for an issue by importing it either from the machine or from the JIRA marketplace. The workflow can be set per project and is very good. Depending on permissions, the user can change the status of the issue only to statuses that have transition from the current one, JIRA doesn't offer all the statuses from the workflow. A list of similar issues is shown at issue creation to prevent duplication. Time tracking is present and the due date can be set. There are custom fields per project and it is possible to define their behavior as needed.

Disadvantages JIRA is not open source nor free. There is a feature called 'agile board', it shows which issues are closed, which are not, and also

queries in progress, but it can be a resource demanding query when too many issues are present, as it would take a long time to load.

Security JIRA provides a flexible security system which defines who can access JIRA and carry out the actions in the system. Five types of security are possible. Global permissions are used for the system access and project permissions are applied to individual projects and its issues. An issue can have a security level set which would restrict the access only to people on the same level, also depending on the permissions defined for the project where the issue is created. Comment visibility can restrict visibility of intended comments and work-log visibility can restrict visibility of individual work-log entries.[9]

Purchase JIRA is neither open source nor free, and the cost increases with the number of users using the system. It begins with $10/month per 10 users and ends with $1000/month per 2000 users. Interested users can try it for free for 30 days. The exact purchase:

10 users	15 users	25 users	50 users	100 users	500 users	2000 users
$10	$50	$100	$200	$300	$500	$1000

[7]

2.4 Mantis

MantisBT is a free web-based bug tracking system. It is written in PHP scripting language and works with MySQL, MS SQL, Oracle (experimental), DB2 (in progress) and PostgreSQL databases and a webserver.[10] It contains various features like email notification, custom fields, built-in reporting, customizable issue workflow, source control integration (GIT, SVN and CVS) or wiki integration.[11] There are about 138 companies or projects using the Mantis bug tracker.[12]

Advantages Mantis provides a mobile interface for iPhone, Android, and Windows Phone to access the tracking data over the phone. It has an API plugin and there are many valuable plugins available, like the email reporting plugin, which allows the creation of an issue by email, the reminder plugin which allows the sending of reminder emails based on the due dates and on the feedback status and so on.[13] There are simple and advanced search filters, as well as a full text

search. When creating an issue there is a field to set how often can the bug be reproduced. The new issue is created in a *new* state and the workflow is configurable per project. Time tracking can be used for the issue.

Disadvantages Mantis is used only as a bug tracker, and it is not possible to track other types of issues, e.g. tasks. The user interface is not very well arranged, many blending colored fields are making it counterintuitive. The workflow offers all the statuses, the list is not based on the workflow transitions.

Security There is a possibility to set permissions for the roles assigned to the users. If anyone wants to set or change some permissions he needs to change a declaration in a configuration file.[14] Users can report security issues in the bug tracker and the team will follow up on them.

Purchase Mantis is open source and therefore a free software. Users can seek help mainly in the public forums, which also helps other users to solve their problems. Mailing lists and IRC channel for the support are also available, and users can also report a bug in the system. Wiki page can offer support as well. Commercial support is also provided, and users can ask for features that best fit their needs.

2.5 BugTracker.NET

BugTracker.NET is a free, open source, web-based bug and issue tracking tool for general purpose. The application is written in ASP.NET, C#, and Microsoft SQL Server (or free SQL Server Express) is used as back-end. It is integrated with a source code management tool and contains a good email integration, with the email thread concerning a bug being tracked with the bug. It is highly configurable, there are custom fields, custom workflows and allows the user to use CSS or his own custom HTML.[15]

Advantages A special feature is the integration with the phone. The phone number can be set to record the bugs. When customer calls, the new bug is created with the attachment of sound recording and the text description. This feature is not free, however.[16] The system as such is free and has some interesting features, e.g. time management. The list of the bugs can be exported into excel. Full text search is also nice. There is a possibility to add custom fields and graphical view of bugs, which enables the display of bugs created at a given date, the ratio of different bug types, division by priority and other attributes.

Disadvantages When creating a bug, a list of statuses of a workflow is provided and the new bug can be created in any other state than *new* and updated to any other state regardless of the workflow. Custom fields can be added by the administrator but are applied to all the projects. Generally, there are not so many interesting features. Time tracking should be available, but is not intuitive to set.

Security BugTracker.NET allows to assign permissions based on projects or organizations. Because of these two items, user may need some experimentation to set the permissions properly. The system also makes a difference between internal and external users. There are permission levels to set for user access or explicitly set the permisisons to users.[17]

Purchase The system is free with the exception of certain features, e.g. phone integration.

2.6 Redmine

Redmine is a flexible project management and bug tracking system. Programming language used is Ruby on Rails and SQLite, MySQL and PostgreSQL can be used as back-end. Redmine is free and open source. It offers some basic features like a support for multiple projects, per project wiki, time management or issue creation via email. It supports several version control systems.[18] Redmine is being used by about 90 companies and projects.

Advantages Redmine is a bug tracking system, which can also be used for issue and task tracking and project management. It has many valuable features, e.g. time management, integration with version control systems, issue creation via email, custom fields of several types, which can be created for the issue and for project shown in the project overview, for users, groups and other categories. The user can set a start and due date in the time management feature, add the exact amount of time spent on a selected task from the dropdown menu. There is also an issue calendar, where the issues are listed, the date when the issue was created is listed under the given day and again under the due date day with special marking. The project page can be customized in an easy way. A possibility to create subtasks also exists.

Disadvantages The list of all the projects is shown on the page arranged according to the 1st letter, is not very well arranged and in case of too

many projects can be too resource-demanding to query. The upper head contains only few links, and when the user wants to search for an issue or create a new issue, he has to find an intended project first. When creating a new issue, the status can be set arbitrarily. When updating, workflow doesn't constrain the order.

Security Users can report any security vulnerability in the form of a ticket if they find any. The system enables role-based access control and is able to protect sensitive data. There is a possibility to set the permissions for 'non member' and 'anonymous' roles. Administrator can create groups and add the users into the groups.[19] But some Redmine versions are affected by XSS vulnerability.[20]

Purchase Redmine is open source and free. There is a public forum where the users can get help or discuss any issues and questions they have and an IRC channel is also available.

2.7 FogBugz

FogBugz is a tracking tool developed by Fog Greek software. It is an integrated system featuring issue and bug tracking, project management as well as wikis, evidence based scheduling, outline tasks and reporting. It supports source code integration as Git or automated error reporting.[21] This tool was originally written in VBScript and because it is used by customers who run it on their own servers it has to run on hundreds of thousands of web servers. The code was shipped to run out-of-the-box, so the custom language Wasabi was created, which compiles to the target platforms. It is backwards-compatible with VBScript but includes obvious improvements.[40] FogBugz is not free or open source and pricing is based on user licenses, where each logged-in user must have one license. The price is about $25/month per user. The system is used by about 51 known companies and in summary the system in trusted by over 20 000 companies in 120 countries.

Advantages The software is paid, but there is a 90 day trial period and when the user is not satisfied, he receives the full refund. The system is used for many purposes outside of bug tracking, like issue and task tracking, project management, wikis, and evidence based scheduling which means that assumed reliability of a user is based on his actual estimation history.[21] There is a simple option to outline tasks and manage the time estimation and due dates on the issues and the time estimation of issues is easy to set. Managers can also see charts

concerning the project's progress, what is missing and what time the team has left. There is a wiki to insert any article into and the system saves the entire version history. The issue manipulation is quite straightforward, the task can be created for a user with just one click and the assignment to another person is also very quick, right on the dashboard with list of issues. Everything is really easy.

Disadvantages FogBugz is not open source and free. Part of the tracking system is the integration with version control system, but that is another product, and needs to be bought with the tracker.

Security Fog Creek team provides very good security for FogBugz. They keep the products security up to date because of the sensitive data. Users can report a security vulnerability if there is any and the support will consider the report with the highest priority. A defence against attack is provided by numerous networks and each server serving a different purpose. One for the database and one for the web servers. The communication between the networks has to follow strict rules. The access to the system is also very restricted, so only FogBugz On Demand systems administration team and the FogBugz Development Lead can access it. Users' data is backed up daily and can be recovered again even after a server crash. Employees are also not allowed to examine the customer's data without his permission.[23]

Purchase FogBugz is not open source or free, but there is a possibility to test it free of charge for 45 days. The price per user is $25/month. When the user is not happy with the product during the first 90 days, he will receive the full refund. Between 24-150 users the price is $599/month, between 151-1000 users the price is $999/month and unlimited number of users costs $2400/month. There is another project part named Kiln, which enables the integration with Git or Mercurial version control system, but it has to be bought separately.

24 - 150 users	151 - 1000 users	Unlimited
$599	$999	$2,499

[22]

	Implementation	Open source	Full text search	Custom field / project
Bugzilla	Perl	yes	yes	no
Trac	Python	yes	yes	no
JIRA	Java	no	yes	yes
Mantis	PHP	yes	yes	no
BugTracker.NET	ASP.NET	yes	yes	yes
Redmine	Ruby on Rails	yes	yes	yes
FogBugz	Wasabi	no	yes	no

	Custom workflow / project	Bug tracker only	Usage	Purchase
Bugzilla	no	yes	1268	free
Trac	yes	no	217	free
JIRA	yes	no	450	paid viz 2.3
Mantis	yes	yes	138	free
BugTracker.NET	no	no	33	free
Redmine	no	no	90	free
FogBugz	yes	no	51	paid viz 2.7

Note: The number of users is not exact. Most of the issue trackers are free and the download count doesn't need to match the list of users. Users can subscribe on some of the pages, reporting their usage of the system, but in reality, the user count can be much higher.

Table 2.1: Comparison of issue tracking systems

3 Analysis of Relevant Processes in Red Hat

There are several projects in Red Hat divided into several phases. They are project planning, development, test and launch phases. Regarding the issue tracking systems, I will mainly discuss the processes in various teams (JBoss Quality Engineering (QE), RHEL QE or Fedora QE) and their mapping in the Bugzilla, which is used throughout each phase as a main issue tracker.

This is because the main motivation of creating new issue tracking system is to gather information about these processes and collect requirements for the system.

3.1 RHEL 6 QE

In general, 4 phases need to be done before the product release.

Planning	Development	Test	Launch

3.1.1 Process Phases Description

Planning Phase At the beginning, the list of bug fixes and feature requests is collected from several teams like support, QE or management. Product management reviews the input and assesses the priorities, then proposes subset of the list to be included and fixed in the upcoming release. In the second half of the phase, it must be decided which bugs will really come into the release. Product management, development and QE must together review and agree on the bug fixes to be applied according according to their capacity to manage everything by the release time and availability of particular resources. If at least one of them doesn't agree, the bug will not be fixed in this release. The priority and severity of the bug is set.

Development Phase All the tasks that were agreed upon are being fixed by development. The are several issues typical of RHEL QE. The code is going to a source code management system for fixed bugs. The list of bugs being fixed is called 'errata'. It is created during the development phase and it consists of fixed bugs. Erratum is created for one bug fix consisting of a given component, list of bug fixes and/or feature requests and a brief description. The public beta is released at the end of this phase.

Testing Phase QE starts testing the component and verifying the bug fixes – errata. If the tests fail the bug returns to the development task. If it succeeds the status of errata is updated to indicate that it's been verified. 5 snapshots are created approximately every 2 weeks and are being tested during this phase. Usually many new bugs are found, reported and fixed and the bug fix is included in the next snapshot.

Launch Phase During the last phase the documentation is being edited and candidate and final releases take place. When a bug is found at this point, it is called a 'blocker'. Blocker issues are either resolved or moved to the next release to prevent regression.

3.1.2 Bugzilla Process

Planning Phase The list of bug fixes and feature requests is proposed during the planning phase. Product management, development and QE must review every issue to be included in the release. This means the issues must be acknowledged. In Bugzilla, there are fields called 'flags'. Their values can be set to:

- " " = not reviewed or proposed
- "?" = proposed
- "+" = approved
- "-" = denied.

If the bug is considered for a release the priority has to be set to urgent, high, medium, or low, according to the urgency of the bug to the customer. There must be a flag pointing to the release version, e.g. rhel-5.5.0, set to "?". 3 flag fields exist, and the following is called the '3-ack process'. First flag is set by the product management to the arranged value. It depends whether the bug is in an approved component, or which bug fixes are needed by customers. After that, there is a development on to set the second flag value. They decide if the bug fixes and feature requests can be done on time and if they have the capacity. And the last one is QE setting the value of their flag. They also must have the resource and time for the verification. If all the flags are set to "+", the release flag will be set to "+" indicating it is approved for this release. Bug fixes have a life cycle. They are going through a certain workflow depending on which project are they part of. Now

the bug is in the state *new*. At the end of this phase, all the bug fixes have to be at least approved by the flags, but can be processed further.

Development Phase When the bug comes into this phase, its state is set to *assigned*. It means that the bug is assigned to a developer and he is working on the fix. At the end the bug fixes are pushed into the source code management system and an erratum is created for every fix. The state can now be set to *post*, which means that a patch for the bug has been posted for review. This state is not compulsory. Another state is *modified*. Some basic testing was executed and the fix was commited to a branch.

Testing Phase ow the state is moved to *on_qa* and the QE team can start with the verification of the fixes. Testers move the Bugzilla status back to *assigned* if the test fails, or to *verified* if the test is successful.

There are other important flags indicating if the bug fix is low and medium priority bug fix in components which are not often updated. Bug fix or feature request is an exception that was proposed after the planning phase, when all requirements were gathered. Or if it's in unapproved component, or a bug fix or feature request can be called a 'blocker', that means it would block target release until resolved.

At the end of the phase, bugs have to be set at least to *verified*.

Launch Phase The bugs are moved to the state *release_pending*, which means that a package is verified by QE and the release engineering team has to handle the errata delivery on Red Hat Network (RHN). Bugzillas are being closed at this point. A Blocker bug can be found in this phase, and a special flag has to be set in the Bugzilla to indicate it is a blocker.

3.2 Fedora QE

The process of Fedora QE is similar, but involves subtle changes. There is an executive team responsible for setting strategic goals, making decisions for the project and helps to achieve the goals. There are 4 phases as well, each covering several issues. The phases overlap, since when one release of Fedora is being tested, another release is already being developed. When one release is launched, the bugs from the release 2 versions back are closed. They can be reopened again if someone finds them still relevant.

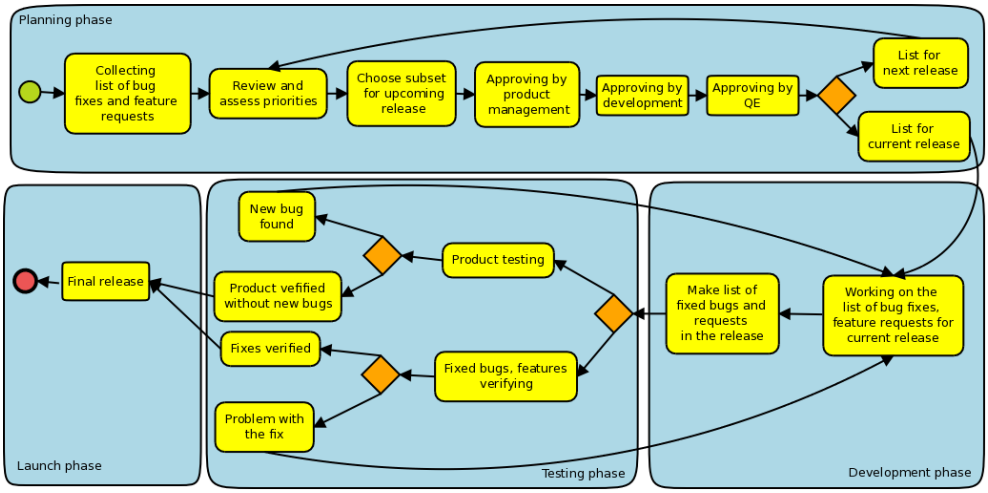

Figure 3.1: Rhel 6 process diagram

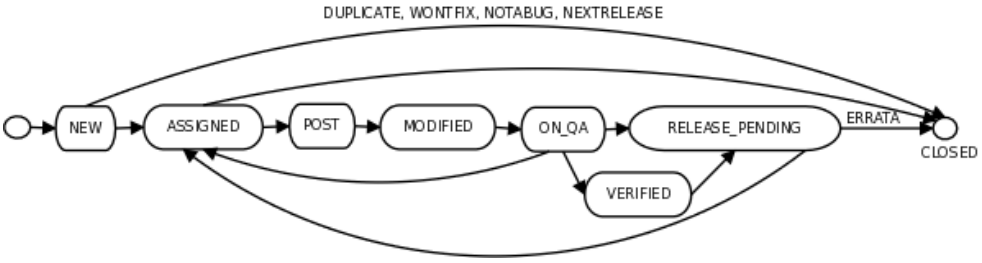

Figure 3.2: Rhel 6 Bugzilla Workflow

3.2.1 Process Phases Description

Planning/Development A list of new features, called 'changes', is created at the beginning. A change is proposed on a wiki page, it must be easy to submit early on and the feedback and review consists of several phases. The accepted change set can be used for the preparation of external materials like release notes. The final release doesn't necessarily need to reflect all the proposed changes. A person responsible for the process of accepting new changes is available. A point during the development, when all the features are done and ready to test, is called 'feature freeze' and ends the development. Another part of development are the Fedora packages which are being developed and maintained by a modular web-based system that enables publishing Fedora package updates. They can be added/updated/deleted. The packages have to meet the quality control requirements for Fedora and are reviewed by a liable person.

Testing/Launch Testing team is working on the whole release. There are several minor releases before the main one. Usually 2 weeks before the release, the change freeze occurs to stabilize the release. Only blocker or freeze exception bugs can be included when their proposal is accepted. Freeze exceptions are small changes that the team wants to be included in the release. They wouldn't be included even when they are approved only if they could cause a regression. No blocker bugs can be known at the time of the candidate release.

3.2.2 Bugzilla Process

Planning/Development phase When the proposed changes are accepted a Bugzilla is created and linked with the corresponding wiki page, where the change is described. The Bugzilla is in the *new* state. When the state is changed to *assigned* a given developer can start working on it. Another states are again *modified* when the bug change is done, and *on_qa* when ready for testing. The only case for using flags is not for the planning, but just as part of process of importing new packages into Fedora. The Whiteboard field is used to set several properties for the package. The difference between RHEL QE process is that the special fields that team agrees upon the changes list is not used here during the planning and different fields are used for many settings.

Testing/Launch During testing bugs are being reported and during the re-

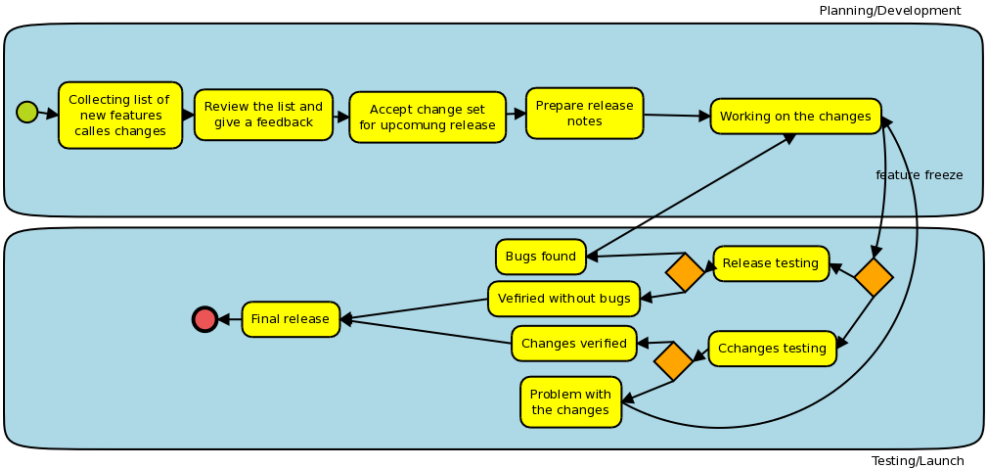

Figure 3.3: Fedora process diagram

lease of Alpha and Beta only blocker bugs and freeze exceptions can be included. That means if the Bugzilla is created, depending on the release, the Whiteboard field is set to indicate that the bug is e.g. (Alpha)Blocker, or a (Alpha)FreezeException, when there is any.

3.3 Middleware QE (EAP, JBoss Fuse, ..)

3.3.1 Process Phases Description

Planning Phase In the first phase, the future features are collected for the release, documentation and test plans are being created, the Support document can be written, and the Product Requirements document is developed. Teams estimate the documents and have to approve that they are able to work according to the plan. At the end there is a meeting concerning the whole process plan.

Development Phase DDevelopers are working on the issues, possible blocker bugs can be reported, and after the alpha release, there is a feature freeze. A check takes place at the end of this phase, verifying that all the features are included and that the majority of them is documented. After the first engineering release, the process is moved to the testing phase.

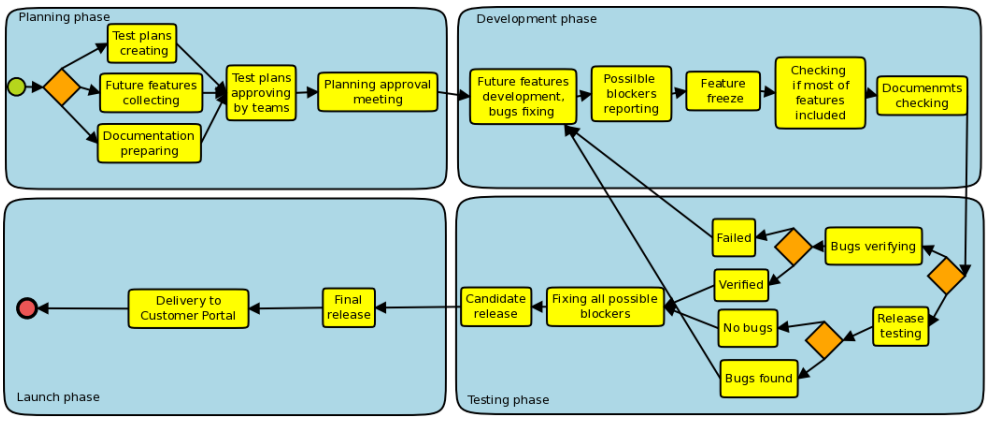

Figure 3.4: Moddleware process diagram

Testing Phase All the bugs are validated, and during the beta release, all new bugs have to be marked as blockers. When all the blockers are fixed, the candidate release can be launched.

Testing/Launch Phase Launch phase is very short, but the testing is also in progress in its course. The product is delivered to the customer portal.

3.3.2 Bugzilla Process

Planning Phase All the future features can be tracked in Bugzilla, but are usually being processed outside of it. More formal requirement tracking takes place in Jira. The spreadsheets are also being used. In this phase, the 3-ack process is not being used in Bugzilla.

Development Phase When a blocker bug is found it is proposed as a blocker bug in Bugzilla by the 3-ack process and, when approved, all three flags are set to "+" and the Bugzilla is set as a blocker. It means the special field value is set to indicate e.g. (alpha)blocker. If not approved, it is opened for the next release.

Testing/Launch Phase This phase is the same as the development phase, possible blockers can be proposed and approved by the 3-ack process and the special field. And finally the major release is out.

3.4 RHEL 7 QE

3.4.1 Process Phases Description

Planning In the first phase, the bug fixes and feature requests are collected from several teams and have to be approved for application in the current release.

Development The development proceeds upstream, as the alpha release corresponds to the Fedora release. The alpha1, 2, 3 is released in this phase. Only bugs that are Requests for Feature Enhancements (RFE) or blockers are tracked internally.

Testing The releases are tested in this phase and new bugs can be found. All the RFEs are tested. Blocker bugs can also be found and the process of verification is more complicated than in the case of normal bugs, and has to be manually reviewed.

Launch During this phase, many issues need to be prepared for the major release. The engineering team did almost all its work in the previous versions and teams like sales or marketing and now many others have to synchronize and make the release ready. Any customer can have the release set and assess it.

3.4.2 Bugzilla Process

Planning List of RFEs is created and confirmed to be part of the current release.

Development During the development phase, Bugzilla is used only for tracking the RFE or (alpha)blocker bugs. No 3-ack process is used, but the flag indicating the bug type can be set to "+" or "-" indicating if the bug is approved, the same for the release flag approving the release version for the bug fix.

Testing In the testing phase, the 3-ack process starts to be used. The product manager, development and QE team have to agree that they are able to process the RFEs in such a way to be part of the intended release by setting the corresponding flags to proper value.

All the bugs have to be verified at the end of the testing phase, the state in Bugzilla is set to *verified*.

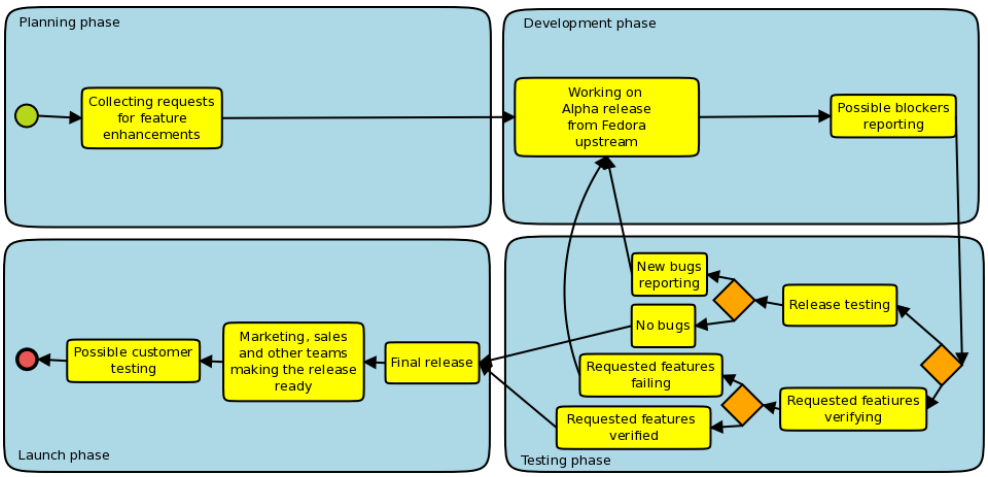

Figure 3.5: Rhel 7 process diagram

Launch Possibly some minor testing can be done and if a blocker is found, the 3-ack process is used to confirm that the bug is in fact a blocker. Bugs are verified and the final release is launched.

3.5 Process Diagram Applied to Workflow

The picture 3.6 displays the process diagram of how the process of finding a new bug is mapped to the issue tracker workflow.

3.6 Bugzilla Processes Evaluation

All the projects and processes are similar in basics with some small changes per project. But the mapped Bugzilla process differs in some ways.

Workflow First problem can be that there is a single workflow for every project. There are 7 states defined, all with special meaning, sometimes making the use counterintuitive. Project manager can't create his own workflow for a project and is obliged to work with the existing one. A major problem is also with permissions: e.g. when a developer gets an *assigned* Bugzilla, makes the changes and should change the state to *post* or *modified* according to the workflow, nothing forces him to do that. In Bugzilla, he is allowed to change the state to any he wants to, even if it's in conflict with the given workflow.

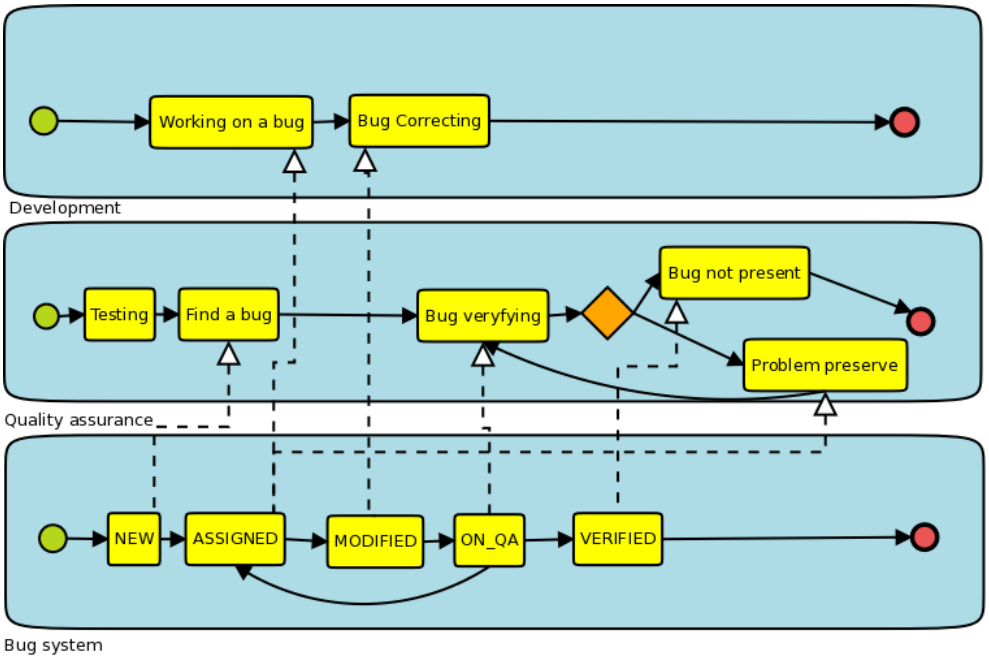

Figure 3.6: Basic workflow applied to bug reporting process

Flags and Fields In Bugzilla, there are several fields of special nature. Every team makes use of them in its own way. Some teams use flags during the planning phase, also for blocker and exception bugs, but some use different field for setting these bugs. And there are many different fields that every team uses in its own way.

4 Motivation for New Tracking System

As mentioned in chapter 2 there are already some issue and process tracking systems, but there is a strong motivation to create a system, which would be really reliable, extendable, and primarily customizable, as these systems are widely used by many companies, all with their own specific requirements.

4.1 Modern Java EE 6 Open Source

The fundamental motivation is to make the issue tracking system modern and open source. This project is dedicated to be divided between many developers, so the code needs to be readable. Since it is an open source application, a major advantage is that other developers can contribute to the code to meet their own requirements and needs, since it's customizable and extendable already from the basics. When many developers are working on the software, there is a great probability for the code to be of high quality with many innovating new features. When anyone is deciding which software to use, it's always advantageous to try the application for the first time and then start using it, this is easy if it is an open source project. Also, since it is free, the issue tracking system is very accessible to every company, and can be widely used.

There may be few good issue tracking systems, which are usually coded well but are of an older date and are not using current technologies. In case anyone would like to read the code he could be faced with a challenge, as the technology has changed since. Java EE 6 is a good choice to use, being truly widespread among developers, very popular, quite easy to use, capable, and containing many advanced features that can be used for every project. And since it is new, it will be more solid and secure than the older technologies.

4.2 Customizable Workflow

Every issue or process goes through the set of statuses and transitions during its lifecycle. Every status shows a state in which the issue is at the given moment and represents a particular point in the workflow. The passing from one status to another is called a 'transition' and represents the meaning of this passing. Basically every issue tracking system has its own workflow created to be used through all the projects. But every project has its

own needs, managers can have different points of view for every task, and the basic workflow doesn't need to meet all the requirements and differences. What is needed is a workflow editor where every project can create its own workflow. When creating a project, there has to a be possibility to add own proper statuses and create transitions from any status to another to best fit the needs of the given project. When no new workflow is created the basic one is used by default.

Another customizable feature concerning workflow is to set a permission role for every transition. Not all employees can do all the work and they have their own tasks assigned to them, so it is appropriate that a specific job is dedicated to person with a certain role and nobody else can pass the issue through the given transition.

The disadvantage of some of the issue tracking systems is that a workflow is created but is only specified in an external documentation. A user might read a very good documentation to see what transition he can make and to which status he can pass the issue, but the restriction doesn't in any way carry over to the real usage of the system. If he wants or if he makes a mistake, he can set the status to one that is not allowed by the workflow, making the possibility of the issue being forgotten more real, as nobody is working on the issue because of a bad transition. What we want is to limit the choice in the drop down menu only to the statuses befitting the transitions of the project so that nothing similar could happen.

Another possibility for passing an issue into another state is using conditions. Issues can have some fields which, when accomplished, allow the user to set a new status.

4.3 Custom Fields

Every project might have some special attributes apart the workflow needing to be defined during the issue lifecycle. If there are special fields for one project, there is a possibility for them to stay empty in another one or there is a need of some new attribute field making the empty one being used in a different way, all of which can cause quite a confusion. We acknowledge that every manager leads the project in the way that is the best for him, so he wants the way to be exactly projected into the used tool. Good way to solve this problem is the possibility to add a list of special fields to every project. These are called 'custom fields', and consist of the name and type of the field. Then when the issue is created in the given project it will contain these custom fields as a default part of itself and the values can be gradually

entered.

4.4 Socialization

Because of the need of an ffective, easy to use, and user friendly tool, it is important to make the application more social by making all the resources and information the users need immediately available. That is why every user in the application should have the option of having his own editable dashboard with all the necessary information. User could follow any projects he is interested in and see the main changes and updates of these projects. The same applies for the list of issues the user is interested in or is working on. When there is a change in any of these issues, he can view important information, e.g. date of the change, its author and all the necessary details on his dashboard. He can also give votes or likes to the issues and those with most votes will be displayed as best. The changes in an issue are being saved in an individual table consisting of a date and a user who made the modification and provided all the necessary details so a user can see the whole history of all the relevant issues. This can considerably improve the work as such because the user can also follow other users to see their work on a task very quickly.

Another good feature is to receive email notifications in case of any activity. It should be possible either to set email notifications on every modification of all issues that are a part of the user's dashboard or to choose those the user will be getting notifications for.

4.5 Security

Security is a crucial part of every application. The solution design in this issue tracking system is to separate users, roles and permissions. Permissions are defined in a basic way as CRUD (Create, Real, Update, Delete) operations that are really included in every used application. Every user has a list of these permissions and it's easy to define them to everyone when the application is being used by a few people. But the problem is the management of permissions when the size of the application becomes unwieldy with huge amount of users. An update of this list would take a really long time. The solution is to create a list of groups, each having its own list of permissions, every user being a member of one or more of these groups. Afterwards the management of permissions and their changes will become more simple, because it's only executed in a particular group and all users

that are a part of this group will be affected. Another part of security in the system are roles. The role has a list of permissions to perform an operations in the system. The users have assigned required roles to be able to execute what is needed to their task. Again when the users belong to a group, the roles can be assigned to a specific group and when there is a need of an update, it's executed in the role itself or the group.

4.6 User Friendly Interface

Interface of every application has to be intuitive and easy to use. The most important part in issue tracking system is the issue creation. When a user wants to create a new issue, he should know what to do. The function of all the fields and buttons should be clear. Also all the settings should be obvious.

4.7 Search

Searching for an issue includes search based on metadata (summary, description, titles) by default. An additionally planned part of the issue tracking system is the full text search. In this manner the search engine explores all the words in every issue possibly bound with some filter and searches if the user input matches, so there is a high probability to find the given issue faster. Users usually search for similar issues or the same again and again, so it is nice not to have to keep populating all the fields of the form. Therefore the user should be able to save his favorite search queries and next time use only the saved references.

4.8 Process Tracking

Issue tracking systems are mainly used by quality engineering teams for tracking bugs or during working on a task to track the progress. But there are many other processes that are needed to be tracked and there aren't many tools which are usable for so diverse processes and that's why employees usually use some tools which don't provide such a functionality to track the whole process, like a spreadsheet or note sticks. It is easy to use but there is no workflow and nobody can see the progress in an obvious way. The motivation for this task is that since one company can have very diverse processes it would be nice to track all of them in one tool. Therefore many parts in the new issue tracking system are really customizable,

and every department can create a workflow that fits its needs best and can adjust all the fields properly for its use, e.g. it can map a process needing some action and then to send an email, so after the given action is completed, there can be a matching state created and a person responsible for sending the email can see the process in such a state and can send the corresponding email. All the processes can be tracked in this way.

5 Analysis, Design and Implementation

5.1 Specification Of An Issue Tracking System Prototype

The system allows the insertion of project with basic attributes – id, name, description, owner – and then extending attributes as lists of individual entities with their own basic attributes – component, project version, custom field and workflow. A project usually has several components and versions that can be inserted simultaneously with the project. Optionally it can have its own customized fields that enable setting values of required attributes of a given project. If no workflow is created for the project, there is a basic one to be used, but every project can have its own. Workflow consists of list of states and transitions, every transition has beginning and end state. User can input the issue state names. The page to update single projects enables to add or remove components, project versions and custom fields and user can view a list of the projects. Another page for editing the project workflow is also present. The main entity of the system is the issue. It consists of attributes – id, name, description, summary, issue type, project, component, project version, creator, priority, resolution, status, file location, list of watching users, comments, relationships and custom fields. When inserting an issue, name, summary and description of the issue there is a dropdown choice of a project, and based on the chosen project, the dropdown choices of components and project versions are modified and the user can choose from possibilities corresponding to the given project and project's custom fields are shown to set the appropriate value. An issue can have an attachment and a file location can be set by the file chooser. A list of predefined priorities is available. When saving the issue, its detail is shown on the page. There is a button by which the user can watch the issue by being added to the list of the watchers, and can also comment on the issue. When there is an issue that the inserted one depends on, or is in any relation with, user can fill the id of that issue and set the relationship type. The status of the created issue is automatically set to *new*, and depending on the project the issue pertains to, the workflow of the project, if present, is added to the issue, if not, the basic one is added. The user can choose another status into which he wants to move the issue from the dropdown list. There is a list of statuses that the user can move the issue to, depending on the transitions of the given workflow, and only these statuses are offered in the list

The scope of the practical part of this thesis is a prototype implementation of new issue tracking system in Java EE that is modern, customizable,

and includes innovative features.

5.2 Use-Case Diagram

A use-case diagram of intended issue tracking system is displayed in the picture 5.1. There are basic functions as team member creates an issue and developer gets it assigned. Both can update an issue by adding a comment, update a (custom) field, make a transition into another state, watch the issue or close the issue when needed. They can also search for an issue with a basic search by filling a text contained in the name or description, setting project, component, version and so on. Search is also possible as a full text search. Every user can utilize a customized stream dashboard by adding desired projects, issues or users on the dashboard. A manager is able to create and update a project and required custom fields for it and also create a customized workflow and assign it to the project..

5.3 Data Model

There are up to 20 entities in the project, they will be divided into special groups and shown in the data diagrams.

5.3.1 Data Model of Issue Tracking System Security

There are four main entities as part of security:

- Permission
- Role
- Group
- User.

Users are supposed to belong to groups that have specific roles. Every role has an assigned list of permissions enabling it to perform given operations. Diagram in the picture 5.2 shows the relationship between these entities.

5.3.2 Data Model of Project, its Elements and Wokflow

The entities are:

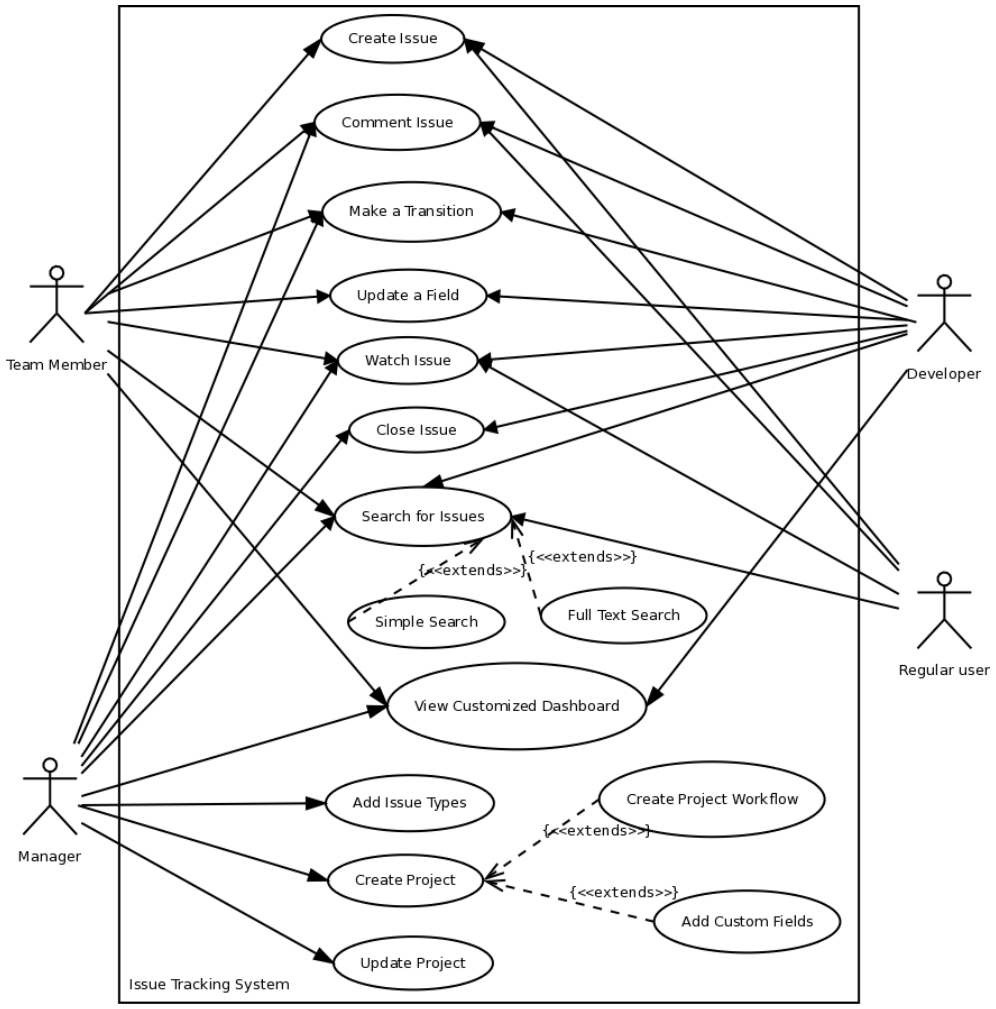

Figure 5.1: Use-case diagram

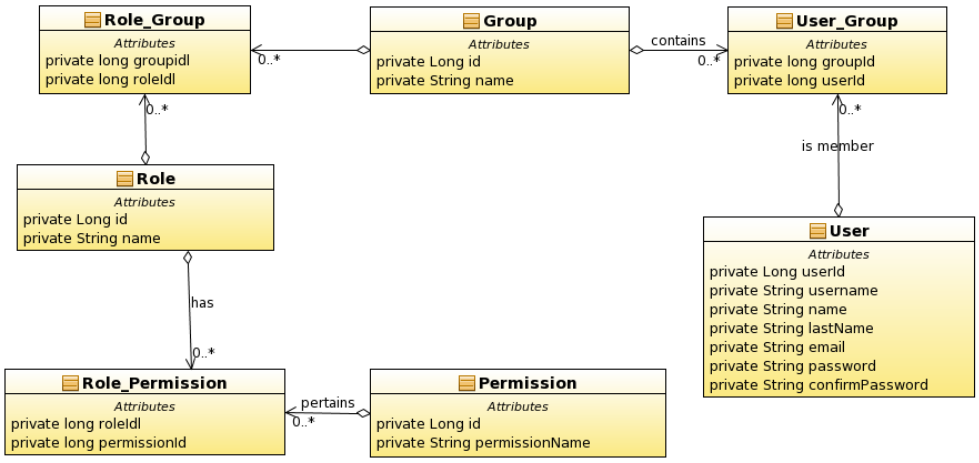

Figure 5.2: Data model diagram of security

- Project
- ProjectVersion
- Component
- Workflow
- CustomField
- Status
- Transition.

Every project has several components and project versions and possibly contains custom fields. A project has either the basic workflow or a new custom workflow can be created and assigned to the project. It consists of statuses and transitions, each transition having the beginning and end state.

5.3.3 Data Model of Issue Entity and its Components

When creating a new issue, one project entity is assigned to the issue. Project contains its components and versions and issue is created for a specific component and version for a given project. Also the project can contain arbitrary amount of custom fields and depending on it, issue has these fields to

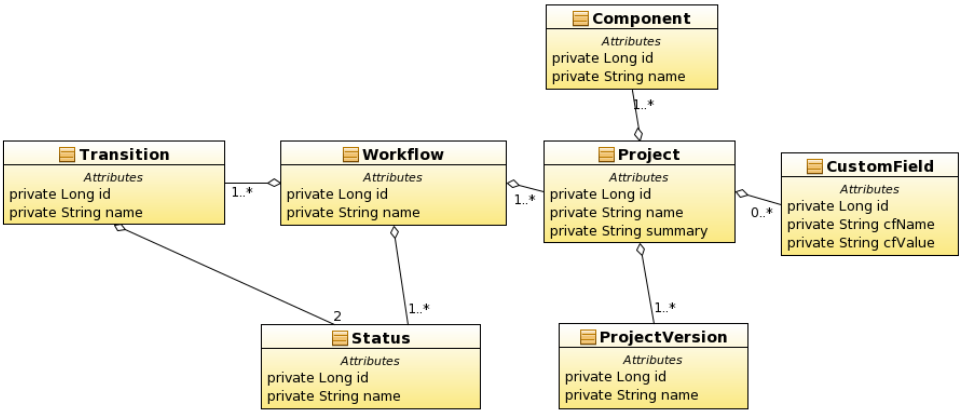

Figure 5.3: Data model diagram of project and its components

fill. When the project has its own custom workflow created, it is assigned to the issue. Every issue is of a chosen type and users can add comments.

The entities are:

- Issue
- IssueType
- Comment
- Project
- ProjectVersion
- Component
- Workflow
- CustomField
- Status.

5.4 Implementation with Wicket

Because one of the reasons for creating a new issue tracker is the modern Java EE technology, the Apache Wicket framework is used for the implementation. There are many parts of code divided into singular components,

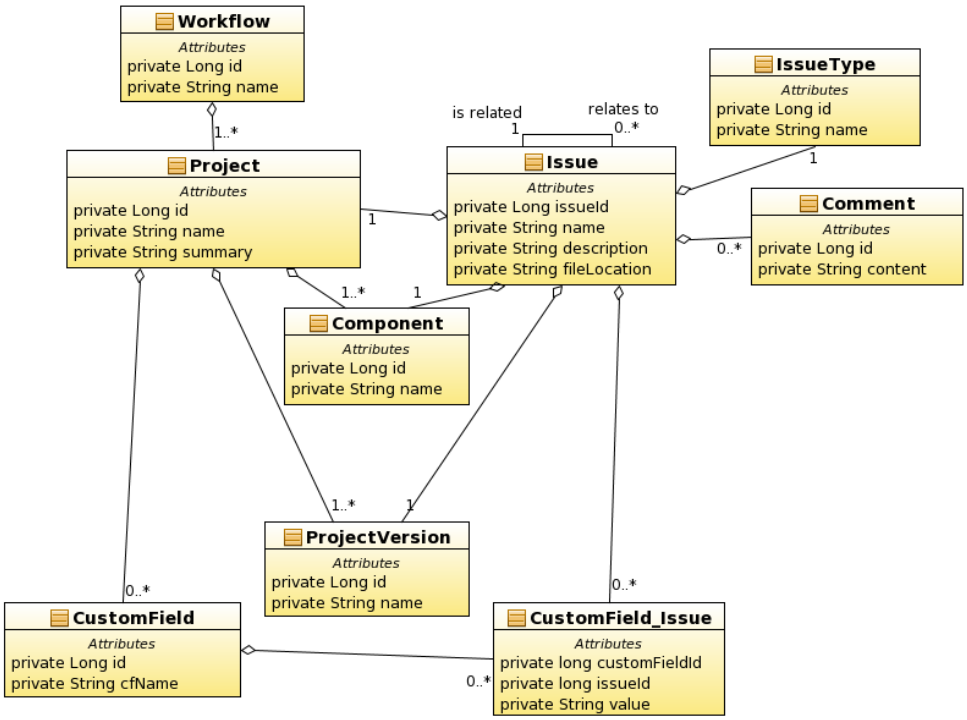

Figure 5.4: Data model diagram of the issue and its components

offering code reduction and reusability. Every page is created as a single component consisting of many smaller components. E.g. in the page for creating a project, a possibility to add several project versions or project components exists. These lists are implemented as a ListView, which form a single component in a diferent package. These values are listed in the page, so the ListView components are used again. They list the values depending on a model they get in constructor and conditions can be used to enable or disable some fragments that need to be displayed depending on the page. Another page regarding the issues also use the separate components, especially the forms. In the page showing a detail of the created issue, there is a form that enables to write and save comments, and another one to change the state of the issue and possibly connect the issue to another one that it relates to. These forms could be reused again in another page when needed. ListView of custom fields and its values is used on this page. Login page also uses the login form component, and there are even two links also as separated components. This makes the code easier and arranged well.

5.4.1 Wicket Advantages

Wicket is a modern Java EE framework, created to improve existing drawbacks in development based on experience. There are many advantages in using Wicket for the implementation.

The development with Wicket is quite easy and intuitive in comparison with other existing frameworks. There aren't any configuration XML files, every page has its own interconnected HTML and Java class and all the objects in the Java class have their references in the HTML markup, and both classes are located in the same package and have the same name, so it is obvious which HTML page goes with intended functionality.

Wicket solves the form inputs validation by its validators. Any validator can be created for a special purpose containing all the conditions for the inputs and can be used for every corresponding input.

Use of the Ajax in the page is also easier. Wicket encapsulates the entire JavaScript in special Ajax methods, which look like normal Java functions, so all the work is done in background. When refreshing a component by clicking,for example, all the components intended to be processed can be added to the Ajax target parameter of the method in an easy fashion.

The code can be very reusable. All the pages consist of components, each can be created as a separate Java class with its own HTML page as a panel added into the page and linked by id to the page's HTML markup class.

Wicket does much of the HTML work for us. It is easy to add a com-

ponent into the code and reference it in the HTML wherever needed. The single components refer to the kind of HTML syntax without the need of understanding the code, just all the components have to be referenced.

Developers can use detachable models to reduce the session memory. When using big values such as lists, the data can be loaded when needed and after the request completion, the data are removed from the session.

5.5 Application Composition

The application can be presented in two parts, back-end and front-end. The back-end stands for the data and methods processing this data and these methods are divided into interface and implementation classes. This serves an easy integration with the front-end that uses only the interface. The front-end presents the graphical user interface and its functionality. This is done by the Wicket pages and its HTML pairs. It enables all the functionality like creating and editing the projects, its components and versions, custom fields or workflows, creating and editing the issues and its attributes, types, comment the issues or move them through the workflow by changing the statuses.

5.6 Used Technologies

5.6.1 JPA

Java Persistence API is Java EE standard for ensuring data persistency by Object Relational Mapping (ORM) or converting object data to database tables. Objects representing data are POJOs (Plain Old Java Object) that don't need any container and it is working with them like with usual objects. These objects are completed with the aid of annotations with helping information defining a way of mapping them to the relational database tables.[32]

Entity
is compulsory. @Entity declares the class as an entity and @Id declares the identifier property of this entity. @Table is set at the class level and allows the developer to define the table, catalog, and schema names for the entity mapping. If no @Table is defined the default values are used, mandatorily: the unqualified class name of the entity. The @Table element contains a schema and catalog attributes, if they need to be defined. Developer can also define unique constraints to the table using the @UniqueConstraint

annotation in conjunction with @Table. Columns' names are defined with @Column annotation, exactly @Column(name="columnName"). If not, the default attibute name is used as a table column name.[33] There is also the defined mapping between entities. Single Valued Mappings are a mapping from source entity to target entity where the cardinality of the target entity is "one". @OneToOne and @ManyToOne are examples of Single Valued mappings. Collection Valued Mappings are a mapping from source entity to destination entity where the cardinality of the target entity is more than one. @OneToMany and @ManyToMany are examples of Collection Valued Mappings. Cardinality defines the number of entities that exist on each side of the relationship. Another mandatory annotation is @JoinColumn. A Join Column in JPA is a column in the owner entity (the entity that has a join column is always the owning entity) that refers to a key (usually a primary key) in the non-owner or inverse entity. JoinColumns are nothing but a foreign key columns. JPA calls them Join Columns, possibly because they are more verbose in what their actual role is, to join the two entities using a common column.[34]

5.6.2 Entity management - JPQL/Criteria API

Java Persistence Query Language (JPQL) ensures the entity management for JPA. Its syntax is very similar to the syntax of SQL. The main difference between SQL and JPQL is that SQL works with relational database tables, records and fields, whereas JPQL works with Java classes and objects. JPQL query can retrieve and return entity objects. That makes JPQL more object oriented friendly and easier to use in Java. Besides a few exceptions, JPQL is case insensitive. Another possibility is to use JPA Criteria API. The JPA Criteria API provides an alternative way for defining JPA queries, which is mainly useful for building dynamic queries the exact structure of which is only known at runtime.[31]

JPA Criteria API vs JPQL
JPQL queries are defined as strings, similarly to SQL. JPA criteria queries, on the other hand, are defined by instantiation of Java objects representing query elements. A major advantage of using the criteria API is that errors can be detected earlier, during compilation rather than at runtime. It is proper to use when building a dynamic query based on fields that user fills at runtime in a form that contains many optional fields because it eliminates the need for building the query using many string concatenation operations.[31]

5.6.3 Context and Dependency Injection

Context and Dependency Injection (CDI) is a standard for dependency injection (DI) and interception (AOP). CDI is a foundational aspect of Java EE 6 and can be embedded in any application. CDI enables objects to have their dependencies provided to them automatically but adding an @Inject annotation, instead of creating them or receiving them as parameters. CDI also manages the lifecycle of these dependencies. Context enables to bind the lifecycle and interactions of stateful components to well-defined but extensible lifecycle contexts. Dependency Injection enables to inject components into an application in a type-safe way and to choose at deployment time which implementation of a particular interface to inject. CDI redefines the concept of a bean where a bean is a source of contextual objects that define application state and/or logic. A Java EE component is a bean if the lifecycle of its instances may be managed by the container according to the lifecycle context model defined in the CDI specification. A bean can have a scope defined that means that a bean must be able to hold a state during the interaction with an application for a specified scope. The scope can be defined for request, session or the whole application which means the container always get the same instance when it is injected into the application, all done by annotations. There are many other annotations for possible use as Qualifiers. [30]

5.6.4 WICKET

Wicket is a lightweight component-based web application framework for the Java programming language. Wicket applications are trees of components, which use listener delegates to react to HTTP requests against links and forms.

There are many choices to use for Java application, so developer has to choose one that best satisfies his needs. Wicket is a good choice because it is really easy to work with, is customizable, the code is extedable and reusable. There are no configuration files to learn in Wicket. It is a simple class library with a consistent approach to component structure. Wicket uses a special HTML attribute to denote components, enabling easy editing with ordinary HTML editors and has a first-class component model.

Another advantage of Wicket is that the code is secure, all logic in Java with maximum type safety and it is easy to integrate with Java security. URLs do not expose sensitive information and all component paths are session-relative.

Managing the Server-Side State
Wicket makes it easy and often transparent to manage server-side state. Every server-side state is automatically managed and is associated with components. Each server-side page component holds a nested hierarchy of stateful components, where each component's model is, in the end, a POJO. Wicket maintains a map of these pages in each user's session. User deal with simple, familiar Java objects and Wicket deals with things like URLs, session ids and GET/POST requests. This makes it really easy to deal with the back button problem. It supports configurable page version management. When user submits a form or follow a link from a page they accessed with the back button in their browser, Wicket is able to revert the page object to the state it was in when the page was originally rendered. So it supports the back button with very little work. Wicket has been designed to work with POJO persistence frameworks such as JDO or Hibernate, so it's simple to write database driven web application.

HTML Code
Many frameworks add some form of special syntax into the HTML code, changing the nature of HTML in the process. This can be more difficult to work with but Wicket doesn't do it. Instead, it extends HTML in a standards-compliant way via a Wicket namespace that is fully compliant with the XHTML standard. Every Wicket component has its id and it matches the special HTML attribute. The components attach to the HTML. When we have a CSS stylesheet append we are able e.g. to set the HTML tag values directly in the Wicket page code to the components. We can get and set a value to the tag or append the tag in the code which will be displayed in the web page. The markup contains only simple tagging, no programming semantics.[28]

Wicket Models
One of the basic Wicket parts are models and their purpose is to hold a value for a component, so the model and the component are two different parts. There are many ways of holding the value which depends on the model implementation. The model entity has to implement the IModel interface, where the values can be set and get by methods. The types of the values depend on the components, so e.g. when there is a Label component, the model value has to be an object able to be converted into the string, regarding ListView component, the value has to be of the List type. The Wicket models enable to retrieve a model value from a resource file or

from the database only when needed. This models can be used with every component. The IModel interface can reduce session memory consumption and is particularly useful with large values such as lists, because the data can be retrieved when it is really needed and when the request is complete, the data is removed from the session. It is not a duty to provide a model to every component, the value can be provided to the component and it wraps it into a default model, but it will then be static. On the contrary, when providing a model it is dynamic because it can get data from the database and there can be a different value everytime.[29]

5.6.5 Test Driven Development with Wicket

Application testing is a very important part of the development. Testing should improve the development, reveal errors in the code before running the application. Basically it is better to spend some time by test development because every code change can cause changes and unexpected bugs anywhere in the code. The tests ensure that the code works as expected.

5.6.6 Junit

Junit is a testing framework for Java programming language. Developer writes a part of the code with a special functionality into a test to ensure it works correctly. Particular tests are isolated units that are independent from each other and each tests a specific functionality. The percentage of tested code is called 'test coverage' and the more code is covered by unit tests the better.

Test classes are usual Java classes and should be named after a Java class which is being tested. The test class contains several tests covering a specific functionality of a given Java class, each annotated with @Test annotation. The test class can also contain methods that are executed before the test methods: @Before, after the test methods: @After, or @BeforeClass, @AfterClass. The validation usually happens by a set of assert methods that check the expected functionality. If an assert method fails, it is recorded for the exploration.

5.6.7 WicketTester

Wicket Application consists of several pages and many components, models or behaviors that are included in the pages and all of them should be tested. The main class of Wicket testing is called WicketTester and provides

a set of methods for the tests creation, e.g. to render a component, click link, start page and check if the page is rendered correctly, if contains a given component, labels of feedback panels show what is expected, and so on, also in case of the assert methods.

When WicketTester is instantiated, it can either contain an instance of our WebApplication, or when is not defined, it will use a MockApplication. Mock objects can be created instead of real objects to emalute their behavior. When we start a page by WicketTester and check if it is rendered correctly, we can also check all the labels on this page. Further try to click a link on this page and check the labels on the target page of the link. Another issue tested can be the state of the components. It can be checked if the component is visible/invisible, enabled/disabled or required. Components can also be tested apart from the pages without starting the container page. We just start the components and can check its internal details. It is also possible to test AJAX components, behaviors or events. Other important part is testing of forms, where we can set inputs and check either error or info feedback messages or expected submit behavior. There are models in the components and it also need to be checked if they contain expected data objects. The methods usually take as input the component or the page-relative path to it.

Every Java page class has a related HTML class with the page markup and a TagTester class is used to test the components markup. TagTester checks if the generated markup contains tags with attribute and intended value. WicketTester extracts the HTML document and TagTester can be created, e.g. by this document and attribute-value pair. Then assert methods can be used for the tag's validation.[35]

5.7 Using the System for Tasks

One of the advantages of the tracking systems is that it can have multiple usages. The following is a description of Red Hat hiring process, which could be mapped to the tracking system and thereby easing the process.

5.7.1 Hiring Process

Since issue tracking system can be applied not only to bugs and issues of a project, but also to tracking diverse processes in various branches, here follows a description of the hiring process interconnected with some of the other processes. The processes are not exactly divided into phases. There is just the planning phase followed immediately by realization.

At the beginning, there is an available position in the office. Manager takes responsibility for this position, gets some budget to find and hire an appropriate candidate. Manager handles details for this position as a description, required level of candidate's expertise and the approved position can appear on the web site. The search starts, and may happen in several ways. If no one is found, the position has to be modified, e.g. by dispensing with some of the requirements. When an appropriate candidate is found, he has a meeting with the recruiter concerning the basic position. The recruiter then sends the output to the manager. Second round involves the technical team and if the candidate passes, a last meeting with the manager takes place and a job offer is created. There is also an event called 'open house'. The public and especially students can come visit the office. They can attend numerous presentations by all the teams and, if interested in a job offer, can apply for an interview. There is another event especially for students - the Open Source Contest. Students can try to solve some real tasks and when successfull, and if interested, can be hired without the technical part of the interview.

5.7.2 Part of the Hiring Process is Internship

Most of the time, students aren't hired for a full time position, but for an internship position. During planning, technical teams interested in any interns send request to the recruiter with all the necessary information for the position and a person is found for the interviews. The contacts for students interested in such a positions are collected before the hiring. Then, the invitation to the entrance test is sent to all the contacts. The technical teams are correcting the tests and deciding who to invite for another meeting. When they pass, they are allocated into the teams. The internship lasts one year and after that the decision if the contract will be canceled, renewed or the intern will be changed to the regular employee is made.

A cooperation with other processes is part of the hiring process. There are several teams involved, e.g. the coordinating team, managers and recruiters, who are entering the data into the system, the financial department, which agrees on the positions being created, facilities, which assign sitting space etc. When the new employee arrives, an onboarding process begins, consisting of the facilities' introduction and/or of an introduction given by an associate from the given team, providing the basic information about the job to the newcomer.

5.7.3 Tracking Systems Used for Hiring

During the planning of the hiring process of the interns, the online calendar tool is used to collect contacts. The students have access to it and write a suitable date for the test next to their name. Most of the work on this process is done manually. Students are being tracked by the Wiki pages, where all the information about them is stored, and the scanned tests and the comments from the technical teams are being preserved. For the employment hiring, there is an Applicant Tracking System (ATS) used for tracking the candidates. The managers, recruiters, and financial department have access to the system. After the meetings, the interviewer hands over the information to the manager through the ATS. When the contract is being created, the financial department and the manager have to give a permission again through the ATS. The applicants have access to the system and are being tracked up to the state *hired*.

6 Conclusion

The main objective of this thesis was to analyse and design an issue tracking system, compare the advantages and disadvantages of existing systems, analyse several processes in Red Hat that are applied to such a system and use this information to find a way to create a good system that is both reliable and efficient. An additional objective is to create a prototype implementation of such a system using modern Java EE technologies.

The comparison of seven existing systems showed many features and benefits that should be gathered and included in a new system, and also drawbacks that shouldn't. One of the main disadvantages is a bad GUI. A GUI has to be easy and intuitive for users to work with, avoiding a long search for every feature and link. The GUI should be arranged well. It is also necessary to make the system very customizable, as it needs to be applicable to many processes and used by many different teams, since they all approach their work differently, even if the task is very similar. Customizable objects exist in many systems, and sometimes they are applicable to the whole system only, sometimes they don't work in an intuitive way. All the processes should customize their system to fit their specific needs, but the change shouldn't affect other, uninvolved teams. Another important point is the restrictions. In some systems users are technically able to undertake an action which they are not permitted to undertake, with the proper usage only being described in a document. Groups, roles, and permissions should be applied to such actions and restrict the usage of the system to make everyone involved's work easier. Eventually, a tool containing a business logic and rules to make such restrictions should be included.

The system prototype is implemented in a modern way, which all the developers know or at least can learn easily, with the Java EE being a good choice. Frameworks used are JPA, CDI and Wicket. The main framework is Wicket, which is used for the functionality implementation. The development with Wicket is easy, component-based and it is simple to add components into the existing code, because they can be separated and just added into the pages. They are reusable end the code is extendable.

The analysis of a few Red Hat processes showed that there really are many differences during the phases in the teams and the mapping to Bugzilla differes in some ways. Teams use different fields, different workflow, and have a distinct way to process their tasks, all of which should be applicable by the system to set the most proper solution.

There are many plans about extending the implementation to include

more features.

Full Text search Basic search queries based on setting a project, component, its version, possibly type, workflow state or severity of the issue is quite effective, but the search proceeds only based on metadata, so there are many advantages of using full text search instead. Full text search is based on the indexes and so it is really faster with the same amount of data than the basic search and is efficient especially when the database contains a lot of data. The operators, e.g. AND, OR, NOT, can be used for the search based on keywords and phrases, also some functions, e.g. contains(), which helps to clarify the query.

Security by PicketBox and PicketLink PicketLink provides support for identity management which allows to manage identity objects like users, roles or groups. Identity manager also provides a possibility to manage user's credentials and other attributes and other properties for objects. PicketBox provides the authorization, authentication, audit and other security functionality. In PicketLink, there is a PartitionManager that allows identity partitioning and so the web application can be really secure. It also provides the core Identity Model API classes upon which an application's identity classes are defined to provide the security structure for the application.[38]

Custom Stream Dashboard Users appreciate when they have the possibility to view all the necessary aspects for the first time. That is why they can give votes and likes to the issues and then see them on their customized dashboard. When there is any change or update of the issue, it will be displayed with all the update's details like date, author of the change, and all necessary items in the main stream, allowing them to track what they need. They can also follow the projects, its statistics and issues. The user can change and customize his settings at any time to ease his work.

Plugin Support The requirement of today's applications is that they need to be very customizable, and the possibility of adding a plugin into the application meets this requirement. Plugins add some special functionality or new features and every user can combine whichever he needs and the applications can be able to search for these plugins itself. The third-party developers can extend the application by easily adding new features and its size and complexity is reduced by not including all the functionality at once. The OSGI framework will be

used to make the application pluggable. It is a technology that comprises a set of specifications that define a dynamic component system for Java. It enables to create an application composed by several components. Components are communicating through services that are shared between the components and so the implementation of the components is hidden to each other. The use of OSGI will be used because it reduces the complexity in the development, the code is easier to write and test and is reusable.[27]

In Wicket, there is also a possibility to make the UI pluggable in an easier way because of Wicket's component and object-oriented character.

Time Management Working on a task takes some time and every user can figure what time he probably spends with the task. Every issue includes some work to be done until the issue is closed or moved to another state. It is therefore advantageous to have a tool which is able to set the estimated duration of the task, watch the time remaining and see the entire load of a given employee. As a future expansion, every issue will have a field to set all this information. At the beginning, the employee can estimate the time to finish his task and set the duration starting from a specific date. Every day he will be able to see the remaining time and the end date to finish the task. When his manager wants to see all that he is working on he will have the review of all employees tasks, its duration and the dates of the accomplishment, so he can count his utilization and know the time for new task.

Apache Camel A very beneficial feature of an application is the integration with another application. There may be a need of a message exchange with some important data and information. It may be an information concerning an issue or project insertion, change of an issue and so on. Users can catch the event and route the message to the destination they want. The basic and the most known usage is an email sending corresponding to the events.

Camel provides possibility to implement most of the design patterns from the book Enterprise Integration Patterns (EIP). In a Camel-based application, users create the endpoints and connect these endpoints with routes. Camel provides support for endpoints technologies like a JMS queue, a web service, FTP server, email address and so on. Camel creates a component based on the type of the endpoint and then creates the endpoint and it enables to exchange a messages between the

endpoints. Everything happens through the Camel context, where the endpoints and routes are added and then it is possible to start the context to send the messages.[24]

BRMS When the issue is supposed to be moved into another state, it has to satisfy given requirements. Usually, users can do the transition with one click, but it has to be assured that all the requirements are accomplished. So there should be a system, which allows the transition to be made only when it is really possible regarding the defined process rules and conditions.

Business Rules Management System (BRMS) is a software used to define, deploy, execute, monitor and also maintain the complexity of decision logic used by a system. The logic applies to business rules and includes policies, requirements, and conditional statements that are used in the systems to determine if an action can be carried out. BRMS allows to externalize the business logic from the application code, tools allowing business experts and developers to define and manage decision logic and a runtime environment, allowing applications to invoke decision logic managed within the BRMS and execute it using a business rules engine.[25]

ABRT When there is any error or exception in the system the application can crash. As a future expansion a plan exists to make an integration with an Automatic Bug Reporting Tool (ABRT). ABRT can handle errors or crashes of the system. It is a collection of tools that enable collecting, analyzing and reporting application issues. One of the main components of ABRT is abrtd, which is a deamon that runs at all times, watching for some crash. If there is any, it runs analyzers on it, a needed action, and sends broadcast messages.[26] If the user wants to report the crash, ABRT offers him to choose where to send a report, the destination can be the logger, that saves the report to local hardware, RHTSupport, which sends the report to the Red Hat support centre, Bugzilla, which creates a ticket in the system, upload it using FTP/SCP, email it, or write it to a file.[39]

Bibliography

[1] Individual bugzilla.org contributors. *What is Bugzilla?* [online]. [cit. 19.11.2012]. <http://www.bugzilla.org/about/>.

[2] Individual bugzilla.org contributors. *Features* [online]. [cit. 10.5.2012]. <http://www.bugzilla.org/features/#security>.

[3] Individual bugzilla.org contributors. *Support* [online]. [cit. 10.5.2012]. <http://www.bugzilla.org/support/>.

[4] *Trac Installation Guide for 1.0* [online]. [cit. 19.9.2013]. <http://trac.edgewall.org/wiki/TracInstall>.

[5] *Trac Plugins List* [online]. [cit. 15.11.2013]. <http://trac.edgewall.org/wiki/PluginList>.

[6] *Trac Permissions* [online]. [cit. 10.9.2013]. <http://trac.edgewall.org/wiki/TracPermissions>.

[7] *Jira Overview* [online]. [cit. 2013]. <https://www.atlassian.com/software/jira>.

[8] Wikipedia contributors. *Comparison of issue-tracking systems* [online]. [cit. 4.12.2013]. <http://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems>.

[9] *Configuring Security* [online]. <https://confluence.atlassian.com/display/JIRA/Configuring+Security>.

[10] *Mantis Bug Tracker* [online]. <http://www.mantisbt.org/>.

[11] Individual bugzilla.org contributors. *Mantis Feature List* [online]. [cit. 25.9.2012]. <http://www.mantisbt.org/wiki/doku.php/mantisbt:features>.

[12] *Support* [online]. <http://www.mantisbt.org/testimonials.php>.

[13] *MantisTouch* [online]. <http://www.mantistouch.org/>.

[14] MantisBT Team. *Mantis Bug Tracker Administration Guide* [online]. [cit. 2013]. <http://www.mantisbt.org/docs/master-1.2.x/en/administration_guide.html>.

[15] Corey Trager. *About BugTracker.NET* [online]. <http://ifdefined.com/bugtrackernet.html>.

[16] Corey Trager . *Phone integration* [online]. <http://ifdefined.com/doc_bug_tracker_phone.html>.

[17] Corey Trager. *User permissions* [online]. <http://ifdefined.com/doc_bug_tracker_permissions.html>.

[18] Jean-Philippe Lang. *Redmine* [online]. [cit. 2013]. <http://www.redmine.org/>.

[19] Russ McRee. *Implementing Redmine for Secure Project Management* [online]. [cit. 27.1.2013]. <https://www.sans.org/reading-room/whitepapers/testing/implementing-redmine-secure-project-management-34122>.

[20] *Netsparker identifies a High Risk XSS Web Vulnerability in Popular Redmine* [online]. <https://www.mavitunasecurity.com/XSS-vulnerability-in-Redmine/>.

[21] FOG-CREEK. *FogBugz* [online]. <http://www.fogcreek.com/fogbugz/>.

[22] FOG-CREEK. *FogBugz* [online]. <http://www.fogcreek.com/fogbugz/pricing.html>.

[23] FOG-CREEK. *Support* [online]. <https://www.fogcreek.com/fogbugz/on-demand-infrastructure.html>.

[24] Individual bugzilla.org contributors. *Getting Started with Apache Camel* [online]. <http://camel.apache.org/book-getting-started.html>.

[25] Wikipedia contributors. *Business rule management system* [online]. [cit. 18.3.2013]. <http://en.wikipedia.org/wiki/Business_rule_management_system>.

[26] *Automatic Bug Reporting Tool (ABRT)* [online]. [cit. 2013]. <https://access.redhat.com/site/documentation/en-US/Red$_{H}at_{E}nterprise_{L}inux$/6/*html*/*Deployment_Guide*/*ch* – *abrt.html*>.

[27] Individual bugzilla.org contributors. *The OSGi Architecture* [online]. [cit. 2013]. <http://www.osgi.org/Technology/WhatIsOSGi>.

[28] *Support* [online]. [cit. 2012]. <https://wicket.apache.org/meet/introduction.html>.

[29] *Working with Wicket models* [online]. <https://cwiki.apache.org/confluence/display/WICKET/Working+with+Wicket+models>.

[30] Oracle and/or its affiliates. *Using Contexts and Dependency Injection for the Java EE Platform* [online]. [cit. 2012]. <http://docs.oracle.com/cd/E24329_01/web.1211/e24368/cdi.htm>.

[31] ObjectDB Software. *Java persistence api* [online]. <http://www.objectdb.com/java/jpa/query>.

[32] FI WIKI. *Java persistence api* [online]. [cit. 5.2011]. <http://kore.fi.muni.cz:5080/wiki/index.php/Java_Persistence_API.>.

[33] Red Hat Inc. and the various authors. *Mapping Entities* [online]. [cit. 2004]. <http://docs.jboss.org/hibernate/annotations/3.5/reference/en/html/entity.html#entity-mapping>.

[34] Anuj. *JavaWorld* [online]. [cit. 28.5.2013]. <http://www.kumaranuj.com/2013/05/jpa-2-understanding-relationships.html>.

[35] The Apache Software Foundation. *Free Online Guide for Apache Wicket framework* [online]. [cit. 2013]. <http://wicket.apache.org/guide/guide/chapter20.html>.

[36] The Apache Software Foundation. *Issue tracking system* [online]. [cit. 24.11.2013]. <http://en.wikipedia.org/wiki/Issue$_{t}racking_{s}ystem$>.

[37] *Configuring Workflow* [online]. <https://confluence.atlassian.com/display/JIRA/Configuring+Workflow>.

[38] *Identity Management - Overview* [online]. <http://docs.jboss.org/picketlink/2/latest/reference/html/ch03.html>.

[39] *Automatic Bug Reporting Tool (ABRT)* [online]. <https://access.redhat.com/site/documentation/en-US/Red$_{H}at_{E}nterprise_{L}inux/6/html/Deployment_{G}uide/ch-abrt.html$>.

[40] Joel Spolsky. *Automatic Bug Reporting Tool (ABRT)* [online]. [cit. 1.9.2006]. <http://www.joelonsoftware.com/items/2006/09/01b.html>.

A Content of enclosed CD

The enclosed CD is part of the thesis. It contains:

- Source code of issue tracking system prototype
- Text file with example data
- The thesis in PDF format
- Readme file with installation instructions