MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Issue Tracking Systems

DIPLOMA THESIS

Jiří Janák

Brno, spring 2009

Declaration

I, hereby declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during the elaboration of this work are properly cited and listed in complete reference to the due source.

.....

Advisor

RNDr. Jan Pavlovič

Abstract

There have been many applications oriented around controlling and tracing the lifecycle of project, tasks, and requirements. In the beginning of this work, the author provides an introduction into the problematic of the above-mentioned tracing, mentioning also software development methods and how they relate to this theme.

In the second part of the work, the currently most popular tools for issue / bug tracking will be examined and then the reader will get acquainted with the possibility of their integration into development environments.

In the last chapter, the development of plug-in for NetBeans IDE is presented, which would make it easier for developers to use advantages of the JIRA tracking system.

Keywords

Configuration management, Requirements management, Issue tracking, Bug tracking, CMMI, Bugzilla, CodeBeamer, Rational ClearQuest, JIRA, Trac, Mylyn, Cube°n, NetBeans IDE.

Contents

1 Introduction 7
2 Modern issue tracking 9
2.1 Issue/Requirement tracking systems 9
2.1.1 What should issue tracking systems do? 9
2.1.2 Way of collecting change requests 9
2.1.3 Benefits of using issue tracking systems 10
2.2 Change requests 10
2.2.1 Composition of a Change request 10
2.2.2 Change request types 11
2.2.3 Change request workflow 12
2.2.4 Issue reporting best practices 14
2.3 Quality metrics & Issue tracking 18
2.3.1 Capability Maturity Model Integration (CMMI) 19
2.3.2 ISO standards (concerned with software development) 25
2.3.3 Issue tracking role within CMMI and ISO standards 27
3 Tracking systems overview 29
3.1 Bugzilla 30
3.2 CodeBeamer 37
3.3 IBM Rational ClearQuest 43
3.4 Trac 50
3.5 JIRA 56
3.5.1 GreenHopper plug-in for JIRA 63
3.5.2 Atlassian Bamboo 68
3.5.3 Atlassian Crucible 69
3.5.4 Atlassian FishEye 71
3.6 Head to head comparison 72
4 Tracking systems IDE integration 74
4.1 Mylyn 75
4.2 Cube°n 82
4.3 CodeBeamer ALM plug-in 86
4.4 Atlassian IDE Connector 90
5 Atlassian NetBeans Connector 96
5.1 Specification & Design 96
5.2 Deployment & Configuration 99
5.3 Conclusion 99
6 Conclusion 100
Bibliography 101

List of Figures

Figure 1: Request state transition 14
Figure 2: CMMI schema 20
Figure 3: Staged representation of CMMI and Process Areas 20
Figure 4: CMMI staged maturity levels 22
Figure 5: ISO 20000 sections schema 27
Figure 6: Bugzilla ticket workflow 35
Figure 7: Default CodeBeamer ticket workflow 41
Figure 8: Default IBM Rational ClearQuest ticket workflow 47
Figure 9: Default Trac ticket workflow 54
Figure 10: Default JIRA ticket workflow 60
Figure 11: JIRA concepts 61
Figure 12: Universal plug-in scheme 74
Figure 13: Atlassian NetBeans Connector components 96

List of Tables

Table 1: CMMI Process Areas categories 24
Table 2: Head-to-head comparison of most common issue tracking systems 73

List of Screen shots

Screen shot 1: Bugzilla use-case - Main Page 31
Screen shot 2: Bugzilla use-case - New bug report (select Classification) 31
Screen shot 3: Bugzilla use-case - New bug report (enter details) 32
Screen shot 4: Bugzilla use-case - Simple Search 33
Screen shot 5: Bugzilla use-case - Bug List 33
Screen shot 6: Bugzilla use-case - View/Edit bug report 34
Screen shot 7: CodeBeamer use-case - My Start (personal Main page) 38
Screen shot 8: CodeBeamer use-case - Project dashboard 38
Screen shot 9: CodeBeamer use-case - Project trackers dashboard (select tracking database) 39
Screen shot 10: CodeBeamer use-case - Bug list 39
Screen shot 11: CodeBeamer use-case - Add new issue details 40
Screen shot 12: CodeBeamer use-case - View issue 40
Screen shot 13: ClearQuest use-case - Login to selected requests database 44
Screen shot 14: ClearQuest use-case - Create new defect ticket 45
Screen shot 15: ClearQuest use-case - Add new defects details 45
Screen shot 16: ClearQuest use-case - List of defects (and enabled actions) 46
Screen shot 17: ClearQuest use-case - View defects details 46
Screen shot 18: ClearQuest client for MS Windows 47
Screen shot 19: ClearQuest Designer - modifying ticket workflow 48
Screen shot 20: Trac use-case - Project welcome page (Wiki) 51
Screen shot 21: Trac use-case - Add new ticket 51
Screen shot 22: Trac use-case - View tickets (select searching criteria) 52
Screen shot 23: Trac use-case - List of tickets 52
Screen shot 24: Trac use-case - View/Edit ticket details 53

Screen shot 25: JIRA use-case – Dashboard 57
Screen shot 26: JIRA use-case – Create new issue (selecting project and issue type) 57
Screen shot 27: JIRA use-case – Create new issue (adding details) 58
Screen shot 28: JIRA use-case – Projects Dashboard 58
Screen shot 29: JIRA use-case – Issue navigator (list of issues) 59
Screen shot 30: JIRA use-case – Creating a new search filter 59
Screen shot 31: JIRA use-case – View ticket (enabled actions on the left) 60
Screen shot 32: JIRA GreenHopper plug-in - Planning Board 63
Screen shot 33: JIRA GreenHopper plug-in - Task Board 65
Screen shot 34: JIRA GreenHopper plug-in - Chart Board 66
Screen shot 35: JIRA GreenHopper plug-in - Summary view of the ticket 67
Screen shot 36: JIRA GreenHopper plug-in - Card view of the ticket 67
Screen shot 37: JIRA GreenHopper plug-in - List view of the ticket 67
Screen shot 38: Atlassian Bamboo - "My activity" dashboard 68
Screen shot 39: Atlassian Bamboo - Integration into JIRA 69
Screen shot 40: Atlassian Crucible - View on review of code 70
Screen shot 41: Atlassian Crucible - Add new comment (select code & type) 70
Screen shot 42: Atlassian FishEye - Changelog dashboard 71
Screen shot 43: Mylyn - Overview 75
Screen shot 44: Mylyn - Task List 76
Screen shot 45: Mylyn - Create a new query 77
Screen shot 46: Mylyn - Change sets 77
Screen shot 47: Mylyn - Task editor - Planning tab 78
Screen shot 48: Mylyn - Task editor - Context tab 78
Screen shot 49: Mylyn - Task editor - Task detail tab 79
Screen shot 50: Mylyn - Focused Package Explorer 79
Screen shot 51: Mylyn - Task Search 80
Screen shot 52: Mylyn - Reporting bugs from Error Log 80
Screen shot 53: Cube°n – Overview 82
Screen shot 54: Cube°n – Task Repositories & Queries 83
Screen shot 55: Cube°n - Create a new query 83
Screen shot 56: Cube°n - Task Query Result View 84
Screen shot 57: Cube°n - Task Explorer 84
Screen shot 58: Cube°n - Task Explorer in Task Context View mode 85
Screen shot 59: Cube°n - Task Editor 85
Screen shot 60: CodeBeamer ALM plug-in for NetBeans 86
Screen shot 61: CodeBeamer ALM plug-in - Project Explorer 88
Screen shot 62: CodeBeamer ALM plug-in - Results View 88
Screen shot 63: CodeBeamer ALM plug-in - Editor 89
Screen shot 64: Atlassian IntelliJ Connector 90
Screen shot 65: Atlassian IntelliJ Connector – Servers configuration 91
Screen shot 66: Atlassian IntelliJ Connector – Workspace divider with tabs 91
Screen shot 67: Atlassian IntelliJ Connector – JIRA tab 92
Screen shot 68: Atlassian IntelliJ Connector – Bamboo tab 92
Screen shot 69: Atlassian IntelliJ Connector – Crucible tab 93
Screen shot 70: Atlassian IntelliJ Connector – FishEye support 94
Screen shot 71: Atlassian Eclipse Connector 95

1 Introduction

"Every program contains at least one bug, and can be reduced in length by at least one instruction."
-- Old programmer's bible Chapter 211 Verse 20-21

Change is a constant feature of software development. All projects have their main objective to change something. A system or its parts are upgraded, fixed, or replaced, presumably to provide better or greater functionality, ease of use, reduction of operating expenses, etc.; however, change is not universally benign and must be controlled in its introduction to a project. If not properly handled, change can slip the schedule, affect the quality, and even kill the project. As a project draws closer to its completion, the impacts of change are more severe. Clearly, a mechanism is needed to control change.

A part of the overall change management approach is called **Configuration Management**. Configuration Management is the process of controlling and documenting change to a developing system. As the size of an effort increases, so does the necessity of implementing effective Configuration Management. It allows large teams to work together in a stable environment, while still providing the flexibility required for creative work.

The key part of Configuration Management is **Requirements Management**. Requirements Management involves establishing and maintaining agreement between customer and developer on both technical and non-technical requirements. This agreement forms the basis for estimating, planning, performing, and tracking project activities throughout the project and for maintaining and enhancing developed software. Key activities include:

- planning the requirements phase
- establishing the requirements process
- controlling requirements changes
- minimizing the addition of new requirements (scope creep)
- tracking progress
- resolving issues with customers and developers
- holding requirements reviews

[MIT-LUD] [MIT-PRF] [MIT-WIK2]

This work is concerned with software tools used in Requirements Management. They are called in many ways: **Issue Tracking Systems**, Trouble Tracking Systems, Bug Tracking Systems, Requirements Tracking Systems, etc.; however, their purpose remains the same: collecting requirements, their management, and tracking their progress.

In the beginning of this work, the author provides an introduction into the problematic of the above-mentioned tracing. There, the lifecycle of the requirement and roles involved in this lifecycle will be explained. One of the goals of this part should also be to explain the importance of issue tracking. The end of the chapter takes a look at development and management methods like CMMI and ISO standards (which can also be used as metrics of development quality) and how they relate to this theme.

In the second and third parts of this work, the most popular tools today for issue / bug tracking will be examined and then the reader will get acquainted with the possibility of their integration into development environments.

In the last chapter, the development of plug-in for NetBeans IDE is presented, which would make it easier for developers to use the advantages of JIRA tracking system.

2 Modern issue tracking

2.1 Issue/Requirement tracking systems

As slightly mentioned in the beginning citation, there is an old saying that any program is never truly finished. There is always something that needs to be added, redone, or fixed. A document containing a call for an adjustment of a system is called a **Change request**.

2.1.1 What should issue tracking systems do?

In such situations, when there are constantly new requirements coming, tools are necessary, which would allow somebody to fully and easily:

- share the information across the team;
- have an instant overview of the state of the software;
- expertly decide about releasing;
- set and update the importance of individual fixes and adjustments; and
- have a recorded history of changes.

[MIT-BLA] [MIT-HAU1]

This can be compared to old Czech TV series called "Návštěvníci" (Visitors), where they used a computer called "the Central world brain," but calling such a tool a brain is not pointless because the main function of such a tool is only "to remember":

- what should be fixed or created;
- what the bug symptoms and appearances are, what actually doesn't work;
- how it should work the right way;
- **who** reported the request, who confirmed, analyzed, implemented the solution, and verified it;
- **when** the request was reported, when it was fixed and when verified;
- what led to the decision to choose one way of fixing instead of another;
- **what** changes in code were made; and
- **how long** it took to handle the request.

[MIT-AND] [MIT-HAU1] [MIT-KAN]

2.1.2 Way of collecting change requests

In the real world, there are various ways how to gather change requests:

- crash reporting tools (BreakPad, Bug Buddy, Dr. Konqi, Windows Error Reporting, …);
- **issue tracking systems** (Bugzilla, JIRA, Rational ClearQuest, Trac, …);
- contact forms (web forms, integrated);
- discussion boards;
- e-mails; and
- etc.

[MIT-BLA]

As one can see, the so-called **issue tracking systems** are especially marked because those are tools primarily designed to do this job, to fulfill the above-mentioned requirements. All others are just substitutes, tools primarily designed for different purposes.

2.1.3 Benefits of using issue tracking systems

Issue tracking systems, if used in the right way, can bring following benefits:

- Improve the quality of software;
- Increase satisfaction of users and customers;
- Ensure requests accountability;
- Improve communication in the team and also to customers
- Increase of productivity of the team; and
- Reduce expenses.

[MIT-BLA] [MIT-HAU1] [MIT-KAN]

In one of following chapters, the reader will find out more about software development management methods and standards like CMMI and ISO 20000 (which can be also used as a metric of quality of development) – implementing these measures also brings many other benefits.

2.2 Change requests

2.2.1 Composition of a Change request

A change request is a document containing a call for an adjustment of a system; it is of great importance in the change management process. A change request is declarative, i.e. it states what needs to be accomplished, but leaves out how the change should be carried out. Important elements of a change request are:

- **Reporter's Name** or ID. If there are questions, one needs to know who originated the request.
- **Type of change request** states what kind of request it is (bug, enhancement, new requirement, …)

- A **brief description** of what the problem is. For example, "Fatal error when printing landscape." is a good description; short and to the point.
- **Build or Version number** of the code being built. Is this the shipping version or a build done in-house for testing and development? Some bugs may only occur in the shipping version; if this is the case, the version number is a crucial piece of information.
- The **component** must be specified. This facilitates assigning the bug to a developer assigned to that part of the product.
- List **details, including how to duplicate the bug** and any other relevant data or clues about the bug. Start with how the computer and software is setup. List each and every step (don't leave any out) to produce the bug. Sometimes, a minor detail can make all the difference in duplicating or not duplicating a bug. For example, using the keyboard versus using the mouse may produce very different results when duplicating a bug.
- **Severity** of the request tells the reader of the bug how bad the problem is. Or in other words, say what the results of the bug are. Here's a common list for judging the severity of bugs:

    1. Blocker – blocks development and/or testing work, production could not run;
    2. Critical – crashes, loss of data, severe memory leak;
    3. Major – major loss of function;
    4. Minor – minor loss of function or other problem where easy workaround is present; and
    5. Trivial – cosmetic problem like misspelled words or misaligned text.

    Of course, usually a blocker is handled by development team immediately.

- To who is the bug **assigned?**; who is going to be responsible for the bug and do the work on the bug? Some companies make it so, that there is a set person whose only responsibility is to reassign tickets to the proper person.

[MIT-BIR] [MIT-BLA] [MIT-HAU2] [MIT-WIK3]

2.2.2 Change request types

Change requests typically originate from one of five sources:

- problem reports that identify bugs, which must be fixed (forms the most common source);
- system enhancement requests from users;
- events in the development of other systems;
- changes in underlying structure and/or standards (e.g. in software development, this could be a new operating system); and
- demands from senior management.

[MIT-BLA]

Based on the above-mentioned description, one can categorize change requests into the following categories:

- **Issue**
  Issue can be described as "non-standard" or "not-awaited" behavior of the system. It can be a bug, but needn't be. This type is usually chosen by people who are not sure how to categorize their problem, and later is usually changed to some other type (bug/enhancement) or closed.

- **Bug**
  A software bug (or just "bug") is an error, flaw, mistake, failure, fault, or "undocumented feature" in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result). Most bugs arise from mistakes and errors made by people in either a program's source code or its design, and a few are caused by compilers producing incorrect code.

- **Enhancement**
  This category contains requests for improvements or enhancements to an existing feature or task.

- **Requirement**
  Humans usually classify things as requirements when they include new functionality, new features to a system, either determined by ourselves or requested by a customer.

- **Task**
  This is a common category for both functional and non-functional tasks that need to be done. For example, in this category we can put requests for hardware upgrades or other environmental updates, etc.

[MIT-ATL] [MIT-BLA] [MIT-HAU2]

Of course, taxonomy of change requests can vary depending on company conventions.

2.2.3 Change request workflow

The life of a change request starts as it is **opened** by somebody. Usually, the submitter of a request does not know (and does not need to know) who the appropriate person for handling the request is, so it is assigned to a lead domain expert. The lead domain expert is the person responsible for assigning requests to the appropriate person in the development team.

After receiving a new request ticket, the first step is **verification**. A bug verifier duplicates the bug by following the steps listed in the "Details" section of the bug. If the bug is reproduced and has all the proper information, the ticket can be re-assigned to a person who will be responsible for fixing the bug. If the bug is not written clearly, is

missing some steps, or cannot be reproduced, it will be sent back to the bug reporter for additional information.

Problems can be also marked as "Fixed Duplicates," meaning that the problem is specified as the duplicate of the already fixed one.

The developer accepts the request ticket, which indicates that he/she has seen it and is aware that it is his/her responsibility to resolve it. The developer works on the request and based on the conclusions, he/she assigns a status to it, indicating what the next step should be. Actually there are two ways:

1. The problem described is an issue which **will not be fixed**. The following reasons are possible:

    - The request is "Postponed," meaning that because of technical reasons, time constraints, or other factors it will not be incorporated into the code until the next version of the product.

    - Problem is marked as "Known problem," indicating that the problem or a similar request is already known, and either a workaround exists for handling it, or it is approximately scheduled some time when the proper fix of the problem can be expected

    - Issue is "Rejected" by the developer and should be forced to accept (through Council) or closed:

        - Issue is "Rejected for other reasons" – for example, for the lack of information, change of conditions, impossibility to replicate the issue, etc.

        - The developer has examined the issue, the product requirements, and the design documents and determined that the bug is not a bug; it is "Working as Designed". What the product or code is doing is intentional as per the design.

        - Issue is "Deferred" – meaning that the cost of fixing the bug is too great, given the benefits that it would produce.

2. Problem is **scheduled** by the developer and will be fixed, what can result in following fix states:

    - Fixed status indicates that a change was made to the code and will be available in the next build. Testers search the database on a regular basis, looking for all "Fixed status" tickets.

    - Duplicate status, again, indicates that the problem is found as a duplicate of an existing issue.

- Resolved means that the problem has been taken care of, but no code has been changed. For example, bugs can be resolved by getting new device drivers or third party software.

After being **fixed,** the bug is reassigned back to the submitter, a representative, or tester, who retests the bug by duplicating the original circumstances. If the bug is fixed and everything now works properly, another test with slightly different circumstances can be performed to confirm the fix. If the bug passes the tests, it gets a **Verified** status. If the bug does not pass the test, the request is re-opened and is sent back to the developer.

After being tested, the ticket is **closed**. Here the lifecycle of the change request ends; however, a regression of the problem may occur in the future may occur, and a request can be **re-opened** and the cycle starts again.

In the diagram below, you can also see the so-called **Council**. That means the problem is discussed by the representatives of the developers and the testers to make a decision on whether the bug should be accepted or closed.

[MIT-BIR] [MIT-BLA] [MIT-HAU1]

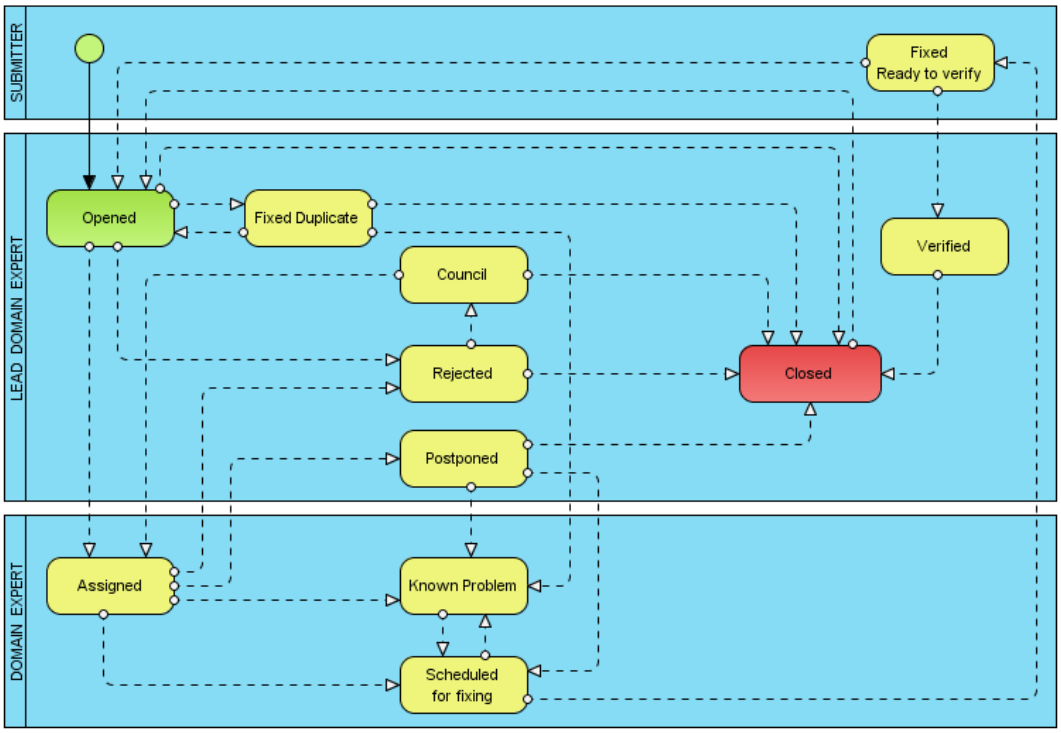


Figure 1: Request state transition

2.2.4 Issue reporting best practices

As soon as you run into a problem in the software, create a new issue request (ticket). In a well-written report, you:

- **Explain how to reproduce the problem;**
- Analyze the error so you can describe it in a minimum number of steps;
- Include all the steps;
- Make the report easy to understand;
- Keep your tone neutral and non-antagonistic;
- Keep it simple: one bug per report; and
- If a sample test file is essential to reproducing a problem, reference it and attach it.

[MIT-AND] [MIT-HAU2]

As was already stated, there are many ways of how a request can end "in the trash" instead of being addressed. To raise the chance of an issue being fixed, there are couple rules which should be followed.

Rule #1 – Write good summary

This one-line description of the problem is the most important part of the report.

- The project manager will use it in when reviewing the list of bugs that have not been fixed.
- Executives will read it when reviewing the list of bugs that will not be fixed. They might only spend additional time on bugs with "interesting" summaries.
- The ideal summary gives the reader enough information to help decide whether to ask for more information. It should include:
    - A brief description that is specific enough that the reader can visualize the failure;
    - A brief indication of the limits or dependencies of the bug (how narrow or broad are the circumstances involved in this bug?); and
    - Some other indication of the severity (not a rating, but helping the reader envision the consequences of the bug).

Rule #2 – Explain how to reproduce the problem

- First, describe the problem. What is the bug? Do not rely on the summary to do this.
- Next, go through the steps that you used to recreate this bug.
    - Start from a known place (e.g. boot the program);
    - Then describe each step until you hit the bug;
    - NUMBER THE STEPS. Take it one step at a time.
- If anything interesting happens along the way, describe it. (One must give people directions to a bug. Especially in long reports, people need landmarks.)
- Describe the erroneous behavior and, if necessary, explain what should have happened. (Why is this a bug? Be clear.)
- List the environmental variables (configuration, etc.) that are not covered elsewhere in the bug tracking form.

- If the reader may have trouble reproducing the bug (special circumstances are required), be clear about these circumstances.
- Keep the description focused:
    - The first part of the description should be the shortest step-by-step statement of how to get to the problem.
- Add "Notes" after the description, if there are any. Typical notes include:
    - Comment that the bug will not show up if step X is executed between step Y and step Z.
    - Explain the reasoning for running this test.
    - Explain why this is an interesting bug.
    - Describe other variations of the bug.

Rule #3 – If two failures are visible, write two reports

Combining failures on one report creates problems:

- The summary description is typically vague. Words like "fails" or "doesn't work" are used instead of describing the failure more vividly. This weakens the impact of the summary.
- The detailed report is typically lengthened. Bug reports should not read something like the following: "Do this until that happens, in which case do not do this until the first thing is completed and then the test case of the second part must also be complete, and sometimes you may see this, but if not then that…."
- Even if the detailed report is rationally organized, it is longer (there are two failures and two sets of conditions, even if they are related) and; therefore, more intimidating.
- Often, one bug gets fixed, but not the other.
- When reporting related problems on separate reports, it is a courtesy to cross-reference them.

Rule #4 – Eliminate unnecessary steps

Sometimes, it is not immediately obvious which steps can be dropped from a long sequence of steps in a bug. Look for critical steps. Sometimes, the first symptoms of an error are subtle.

A list now exists of all the steps that were taken to show the error. Now, the aim is to shorten the list. As each step is executed, any hints of errors must be examined. The following factors should be examined:

- Error messages (i.e. A message appeared 10 minutes ago. The program did not fully recover from the error and the problem evident now is caused by the poor recovery.)
- Delays or unexpectedly fast responses.
- Display oddities, such as flashes, repainted screens, a cursor that jumps back and forth, multiple cursors, misaligned text, slightly distorted graphics, doubled

characters, omitted characters, or display droppings (pixels that are still colored even though the character or graphic that contained them was erased or moved).
- Sometimes, the first indicator that the system is working differently is that it sounds a little different than normal.
- An in-use light or other indicator shows that a device is in use when nothing is being sent to it (or a light that is off when it should not be).
- Debug messages: the debug monitor should be turned on (if it exists on the system) and should be monitored if or when a message is sent to it.

If, what seems like a critical step has been encountered, everything else from the bug report should be eliminated. One must now go directly from that step to the last one (or few) that shows the bug. If this does not work, individual steps or small groups of steps can be removed.

Rule #5 – Variations after the main report

The failure may look different under slightly different circumstances. For example:

- The timing changes if additional two sub-tasks are performed before the final reproduction step is hit.
- The failure will not show up at all or is much less serious if something else is put at a specific place on the screen
- The printer prints different characters (instead of the characters described) if the file is made a few bytes longer

This is all useful information for the programmer and it should be included. But to make the report clear:

- It should start with a simple, step-by-step description of the shortest series of steps needed to produce the failure.
- The failure must be identified. (Descriptions of how it looks like or what impact it will have.)
- A section should be added that states "ADDITIONAL CONDITIONS" that describes, one by one, the additional variations and the effect on the observed failure.

[MIT-KAN] [MIT-RAD]

Overcoming objections

Programmers, as humans, operate under time constraints and competing priorities. For example, outside of the 12-hour workday, some programmers are faced with fatigue and apathy, instead of fixing bugs.

Some things will often motivate programmers to fix the bug:

- The situation with the bug is very poor.
- The complexity piques the programmer's curiosity.
- It affects many people.
- Arriving at the fix is trivial.
- It has embarrassed the company, or a similar bug embarrassed a competitor.
- One of the bug`s cousins embarrassed the company or a competitor.
- Management wishes for the bug to be fixed.
- The programmer is susceptible to flattery from the person with the requests, trusts the person, owes a favor, or accepts bribes.

Programmers avoid spending time on a bug due to the following reasons:

- The programmer cannot replicate the defect.
- Strange and complex sets of steps are required to induce the failure.
- Not enough information is known and it will take a lot of work to solve them.
- The programmer does not understand the report.
- The solution is unrealistic (e.g. "corner case").
- It will take a lot of work to fix the defect.
- A fix will introduce too much risk into the code.
- There will be no perceived customer impact.
- The fix is unimportant (nobody cares about such a minor error or unused feature).
- It is not a bug, it is a feature.
- Management does not care about bugs of this type.
- The programmer does not like or trust the person with the requests, or the customer who is complaining about the bug.

[MIT-KAN] [MIT-HAU2]

This chapter has been written primarily about bugs and their reporting; however, same conditions hold also for other types of requests. When writing request tickets, these conditions should be considered.

2.3 Quality metrics & Issue tracking

For the customer, it is quite difficult to recognize how effective and how well a company works just by observation. To make it easier for people to recognize how mature a company is, some metrics were developed, which allude to the quality of production:

- CMMI model – rating the maturity of a company
- ISO 9000 standards – rating of product quality

Every company tries to optimize its inner processes to be more effective, work better, and minimize resource waste. By obtaining a certification of fulfilling some ISO

standard or some CMMI level, a company proves it is able to work at some specific level. This can be the factor that makes it easier for the customer to decide to hire a company over another.

Trying to reach those certifications can be, first, a good marketing tactic, secondly, it can save money for the company by optimizing its processes, and thirdly, it can be used as a benchmark to assess the adequacy of its quality programs.

This chapter is dedicated to a short introduction of the two above-mentioned metrics and their influence on issue tracking.

2.3.1 Capability Maturity Model Integration (CMMI)

Capability Maturity Model Integration (CMMI) is a set of products used to improve business processes. It was developed by the Software Engineering Institute (SEI) at Carnegie-Mellon University in 1991. It consists of CMMI models, methods for evaluation, and educational materials. CMMI allows the firm to set objectives and priorities for improving processes.

[MIT-WIK1]

The CMMI model contains a set of "best practices," which covers the life cycle from design, through development, delivery, and maintenance. It is designed primarily for organizations engaged in the production and implementation of software; however, it is such a general model, that it can be applied to a wide variety of processes in different enterprises.

The "best practices" say what to do, but do not say how to do it and who shall do it. For example, in the requirements management area, it is recommended to monitor changes in requirements, but the specific methods of monitoring and determining the responsibilities are not explicitly recommended in the model.

CMMI helps organizations to set goals for improving the processes and their priorities. It improves processes, and advises how to ensure stable and capable and mature processes.

[MIT-IBA] [MIT-PRO] [MIT-WIK]

Process area, goals and practices

Individual practices of similar characteristics in CMMI models are concentrated into so-called Process Areas, e.g. requirements management, project planning. A Process Area has 1 to 4 goals and each goal is comprised of practices.
Those goals and practices which are specific to a single process area are called specific goals and practices. An additional set of goals and practices applies across all of the process areas; this set is called generic goals and practices.

[MIT-IBA]

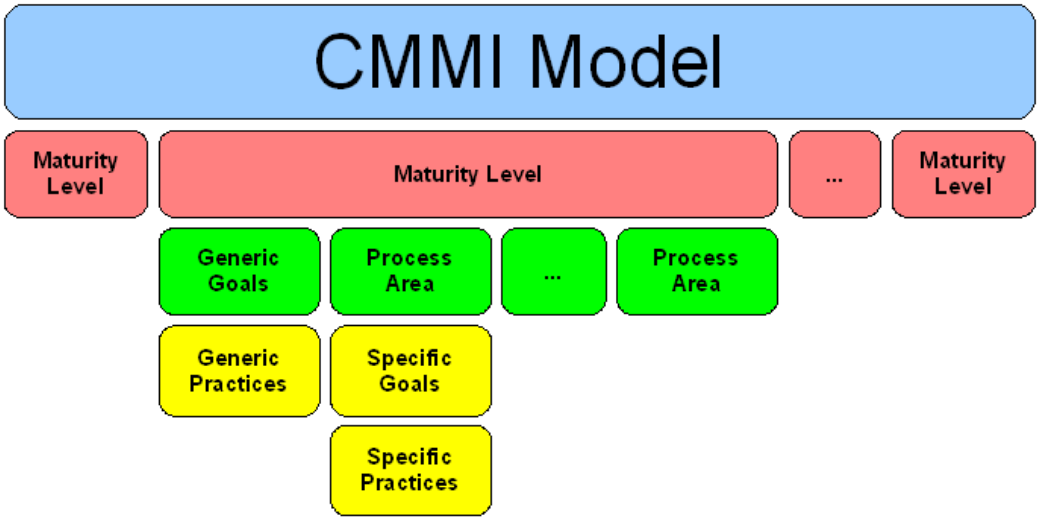


Figure 2: CMMI schema

[MIT-IBA]

Staged and Continuous representation of CMMI

CMMI exists in two representations: continuous and staged. Representations allow the organization to monitor the various objectives while improving the processes. Representations are different in structure, but their content is same.

Staged representation – 5 levels of maturity of companies

Staged representation offers a proven set of steps aimed at improving the processes in the organization. Each step is the basis for further improvement. Staged representation uses **5 maturity levels** of companies. Each maturity level contains an exact amount of Process Areas. **To achieve the level, it is necessary to handle all its Process Areas.** It is not possible to skip the levels, because the handling processes at a lower level is a precondition for achieving a higher level. The advantage of representation is that it allows an overall comparison between organizations based on the level of maturity.

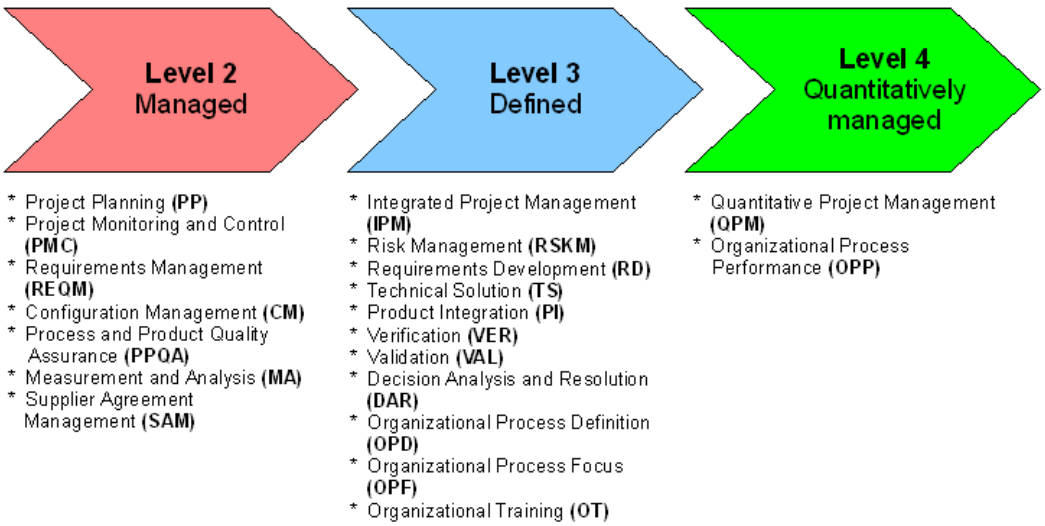


Figure 3: Staged representation of CMMI and Process Areas

[MIT-IBA]

As mentioned above, the CMMI model recognizes **5 degrees of maturity of a company**:

1. **Initial:** Organizations on the first degree of maturity do not depend on the processes. Problems are solved "ad-hoc", there are almost no long term plans. These organizations usually operate through "heroic performance" of individuals. Answers to questions concerning the duration of some task, etc. are merely estimates. There is no systematic procedure and no data which could provide accurate answers to these questions.

2. **Managed:** On the second level of maturity, the organization tries to define its core and the most used processes. The organization is, therefore, focused only on description of specific processes, how much and what kind of processes is decided arbitrarily by the management.

3. **Defined:** Organizations on the third level of maturity already have most of the processes defined. They have not created models of their business processes, but they understand how operation and support of these processes work . From the process architecture, one can determine how the organization works. In the case of a problem, the process can be easily and quickly, which could cause the problem, and also the solution to the problem can be quite easily proposed.

4. **Quantitatively managed:** Organizations on the fourth level of maturity have finished the phase of defining processes. These organizations have managers who collect data on processes performance characteristics and on the satisfaction of customers. They use this data to make decisions on how to optimize processes.

5. **Optimizing:** Organizations on the fifth degree of maturity have processes built in a way that reflects their essence. Processes are defined and managed. In addition, there is a system of their constant progressive improvement wherever it is possible.

[MIT-IBA] [MIT-WIB] [MIT-XIN]

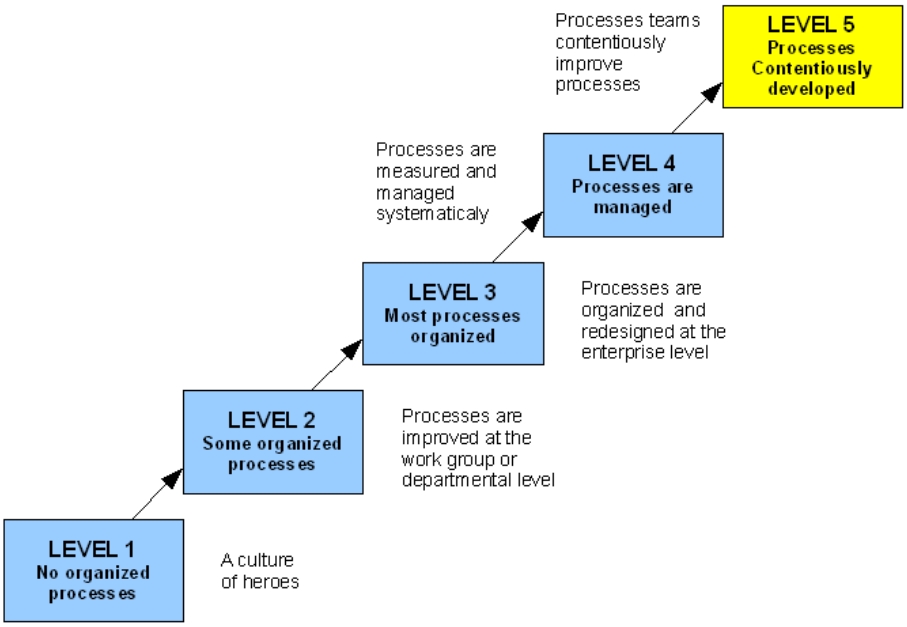


Figure 4: CMMI staged maturity levels

[MIT-XIN]

Continuous representation - 6 levels of capability in process areas in companies

Continuous representation allows the organization to adapt the processes improvement process in such a way, so it best suits the company's business objectives and so it allows one to focus on risk mitigation in problem areas. Continuous representation allows the rating of each process individually and establishes **6 capability levels of processes** for the rating:

0. **Incomplete process**: A process that is either not performed or partially performed. One or more of the specific goals of the process area are not satisfied, and no generic goals exist for this level, since there is no reason to institutionalize a partially performed process.

1. **Performed process**: A capability level 1 process is a process that satisfies the specific goals of the process area. It supports and enables the work needed to produce work products.

   Characteristics:
   - chaotic process;
   - unpredictable cost, schedule, and quality;

2. **Managed process**: A capability level 2 process is a performed (capability level 1) process that has the basic infrastructure in place to support the process. It is planned and executed in accordance with policy; employs skilled people who have adequate resources to produce controlled outputs; involves relevant stakeholders; is monitored, controlled, and reviewed; and is evaluated for adherence to its process description. The process discipline reflected by capability level 2 helps to ensure that existing practices are retained during times of stress.

Characteristics:

- intuitive;
- cost and quality are highly variable;
- the plan is under conscious control;
- informal methods and procedures;

To achieve capability level 2, the following must be met:

- controlled collection of requirements;
- planning of software project;
- subcontracts for the software are controlled;
- software quality assurance;
- software configuration management;

3. **Defined process**: A managed (capability level 2) process that is tailored from the organization's set of standard processes according to the organization's tailoring guidelines, and contributes work products, measures, and other process improvement information to the organizational process assets. It is about synchronization of processes in the organization according to predefined conventions. Processes can be managed between them in accordance with established policies.

   Characteristics:

   - focused on quality;
   - reliable cost expectations and time plans;
   - improving, but still unpredictable profit of quality assurance system;

   Key elements to achieve this level:

   - organization process definition;
   - organization process improvement;
   - training program;
   - integrated software management;
   - application of engineering methods in software product;
   - coordination between working groups;
   - detailed examination and testing;

4. **Quantitatively managed process:** Such an advanced process satisfies the conditions defined in the preceding stage, and in addition it uses quantitative analytical techniques for its control. To measure the quality of the process, there are defined measurable objectives, which are also used for management of process performance.

   Characteristics:

   - quantitative;
   - statistically well-controlled quality of the product;

Key elements to achieve this level:

- measurement and quantitative management of the production process;
- quality management;

5. **Optimizing process**: The highest level of maturity involves measurably controlled processes, which are also changed and developed, so as to permit the performance of current and planned business objectives of the organization.

   Characteristics:

   - focus is on continually improving the process performance through both incremental and innovative technological changes or improvements;

   Key elements to achieve this level:

   - prevention of errors;
   - technology innovation;
   - driven changes of production processes;

This representation allows a comparison between the organizations, only on the basis of Process Areas. Process Areas in the continuous representation are divided into four categories: **Project Planning, Support, Engineering,** and **Process Management.**

[MIT-IBA] [MIT-WIB] [MIT-XIN]

Area	Maturity level	Process Area	Abbreviation
Project Management	2	Project Planning	PP
	2	Project Monitoring and Control	PMC
	3	Integrated Project Management	IPM
	3	Risk Management	RSKM
	4	Quantitative Project Management	QPM
Engineering	2	Requirements Management	REQM
	3	Requirements Development	RD
	3	Technical Solution	TS
	3	Product Integration	PI
	3	Verification	VER
	3	Validation	VAL
Support	2	Configuration Management	CM
	2	Process and Product Quality Assurance	PPQM
	2	Measurement and Analysis	MA
	3	Decision Analysis and Resolution	DAR
Process Management	3	Organizational Process Definition	OPD
	3	Organizational Process Focus	OPF
	3	Organizational Training	OT
	4	Organizational Process Performance	OPP

Table 1: CMMI Process Areas categories

[MIT-XIN]

2.3.2 ISO standards (concerned with software development)

ISO 9000 standards – quality management systems

ISO 9000 is a family of standards for quality management systems. There are three major ISO 900x standards important for software development:

- **ISO 9000:2000 – Quality management systems – Fundamentals and vocabulary.**
  Covers the basics of what quality management systems are and also contains the core language of the ISO 9000 series of standards. A guidance document, not used for certification purposes, but an important reference document to understand terms and vocabulary related to quality management systems. In the year 2005, a revised ISO 9000:2005 standard was published; therefore, it is now advised to refer to the standard as ISO 9000:2005.

- **ISO 9001:2000 – Quality management systems**
  Requirements are intended for use in any organization which designs, develops, manufactures, installs and/or services any product, or provides any form of service. It provides a number of requirements which an organization must fulfill if it is to achieve customer satisfaction through consistent products and services, which meet customer expectations. It includes a requirement for the continual (i.e. planned) improvement of the Quality Management System, for which ISO 9004:2000 provides many hints.

- **ISO 9004:2000 – Quality management systems - Guidelines for performance improvements**
  ISO 9004 covers continual improvement. It gives advice on what could be done to enhance a mature system. This standard very specifically states that it is not intended as a guide to implementation.

[MIT-WIK5]

Because ISO standards are not readable documents, here is a citation of summary of content of ISO 9001:2000 from Wikipedia:

- *The quality policy is a formal statement from management, closely linked to the business and marketing plan and to customer needs. The quality policy is understood and followed at all levels and by all employees. Each employee needs measurable objectives to work towards.*
- *Decisions about the quality system are made based on recorded data and the system is regularly audited and evaluated for conformance and effectiveness.*
- *Records should show how and where raw materials and products were processed, to allow products and problems to be traced to the source.*
- *[One] need[s] a documented procedure to control quality documents in your company. Everyone must have access to up-to-date documents and be aware of how to use them.*

- *To maintain the quality system and produce conforming product, [one] need[s] to provide suitable infrastructure, resources, information, equipment, measuring and monitoring devices, and environmental conditions.*
- *[One] need[s] to map out all key processes in your company; control them by monitoring, measurement and analysis; and ensure that product quality objectives are met. If [one] can't monitor a process by measurement, then [one must] make sure the process is well enough defined that … adjustments [can be made] if the product does not meet user needs.*
- *For each product [the] company makes, [one] need[s] to establish quality objectives; plan processes; and document and measure results to use as a tool for improvement. For each process, [one must] determine what kind of procedural documentation is required (note: a "product" is hardware, software, services, processed materials, or a combination of these).*
- *[One] need[s] to determine key points where each process requires monitoring and measurement, and ensure that all monitoring and measuring devices are properly maintained and calibrated.*
- *[One] need[s] to have clear requirements for purchased product.*
- *[One] need[s] to determine customer requirements and create systems for communicating with customers about product information, inquiries, contracts, orders, feedback and complaints.*
- *When developing new products, [one] need[s] to plan the stages of development, with appropriate testing at each stage. [One] need[s] to test and document whether the product meets design requirements, regulatory requirements and user needs.*
- *[One] need[s] to regularly review performance through internal audits and meetings. [One must] determine whether the quality system is working and what improvements can be made. [One must] deal with past problems and potential problems. [One should] keep records of these activities and the resulting decisions, and monitor their effectiveness (note: … documented procedure for internal audits [are required]).*
- *[One] need[s] documented procedures for dealing with actual and potential nonconformance (problems involving suppliers or customers, or internal problems). [One should] make sure no one uses bad product, determine what to do with bad product, deal with the root cause of the problem and keep records to use as a tool to improve the system.*

[MIT-WIK5]

ISO/IEC 20000 standards - IT Service management

ISO 20000 is the international standard for IT Service management. It is based on, and is intended to supersede, the earlier British Standard, BS 15000. The standard actually is comprised of two parts:

- ISO 20000-1 is the 'Specification for Service Management', and this is what is used to certify organizations. It is comprised of ten sections:

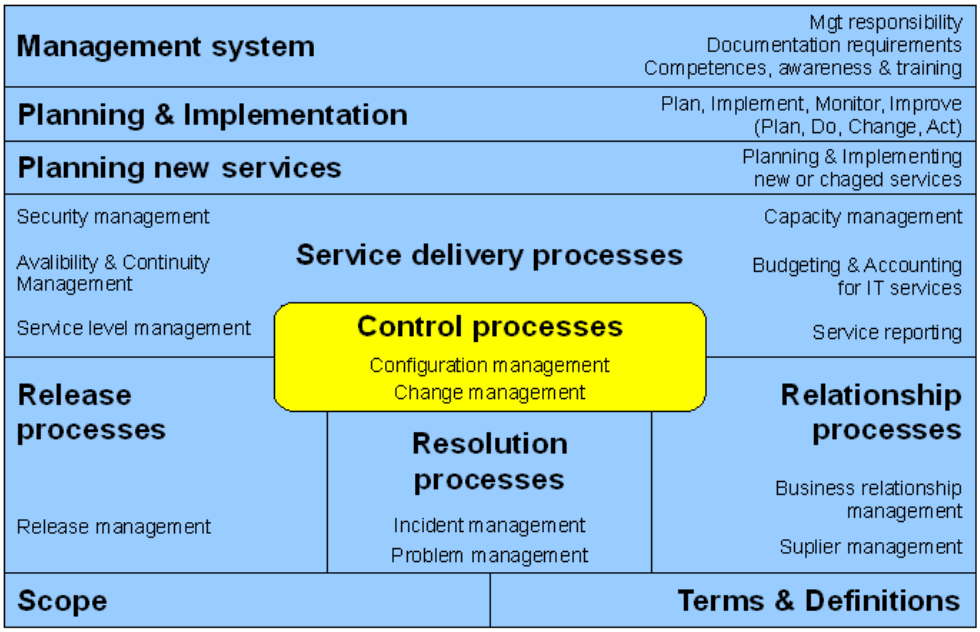


Figure 5: ISO 20000 sections schema

[MIT-LIV]

- ISO 20000-2 is the 'Code of practice for Service Management', and describes best practices, and the requirements of Part 1.

ISO 20000 is a common reference standard for all companies (regardless of sector, size and type of organization) that provide IT services for internal and/or external customers. On the basis of a common terminology for service providers (the IT Service Management Organization), customers, and suppliers, the integrated process approach is consistently regarded as a factor for success.

[MIT-LIV]

2.3.3 Issue tracking role within CMMI and ISO standards

Within all methods, there are areas, more or less, coupled with Issue tracking.

In CMMI, we can find the following Process Areas (up to and including level 4):

- Configuration Management
- Measurement and Analysis
- Project Monitoring and Control
- Project Planning
- Requirements Management
- Verification
- Validation

ISO 20000 introduces the following sections to handle:

- Change Management
- Configuration Management
- Incident Management
- Planning & Implementing new or changed services

- Problem Management
- Release Management

ISO 9001 speaks about the following items, which should be fulfilled:

- *Decisions about the quality system are made based on recorded data and the system is regularly audited and evaluated for conformance and effectiveness.*
- *To maintain the quality system and produce a conforming product, you need to provide suitable infrastructure, resources, information, equipment, measuring, and monitoring devices, and environmental conditions.*
- *One must determine customer requirements and create systems for communicating with customers about product information, inquiries, contracts, orders, feedback and complaints.*

The above-mentioned methods all mention so-called **Configuration Management**. Configuration Management is the process of controlling and documenting change to a developing system.

The key part of Configuration Management is **Requirements Management**. Requirements Management involves establishing and maintaining agreements between customer and developer on both technical and non-technical requirements. This agreement forms the basis for estimating, planning, performing, and tracking project activities throughout the project and for maintaining and enhancing developed software. Key activities include:

- planning the requirements phase;
- establishing the requirements process;
- controlling requirements changes;
- minimizing the addition of new requirements (scope creep);
- tracking progress;
- resolving issues with customers and developers; and
- holding requirements reviews.

Problem management, Incident management, and Change management are just special cases of Requirements Management.

Release management involves splitting projects into iterations. One must plan which requirements can be published to production, and which cannot. To be able to do this, one must determine if they are implemented or not. This is just one of many subtasks of **Project monitoring and control**. To be able to control and decide about the project, the actual state of all its parts must be known, but not only this – people also need to see various statistics – estimated and real work times on the features, etc., which can help them to make better estimates, and which enable them to evaluate the price of the work, or help with decisions about splitting work. People need tools, which collect the data above, which in turn, allows them to do their **measurements and analysis**.

[MIT-LUD] [MIT-PRF] [MIT-WIK2]

3 Tracking systems overview

To handle all requirements mentioned in the previous chapter, people created various Issue tracking systems. In this chapter, the most popular ones will be introduced. There are plenty of other tracking systems – or ports or clones of those mentioned below (for example, a Java port jTrac exists for the Python based Trac, etc.). The five mentioned below are something like exemplars for all others.

Because all below-mentioned systems have their goods, and they must be compared in some way, here are the specific that will be discussed. We will primarily expand on requirements mentioned in the previous chapter.

1. **Installation and initial configuration**
   Brief introduction into systems environment requirements and explanation how difficult is it to setup the tracking system for initial use.

2. **User interface and learning curve**
   Some system user interfaces are more user friendly than others. Shown (if possible by screenshots) are typical use cases "Creating new and looking for a ticket".

3. **Ticket system**
   It is expected that the system allows the setup of workflow and to add custom attributes. Here, the investigation will show how difficult it is to change default workflows, if it is possible to add or modify custom attributes, and what steps are needed to make this happen.

4. **Projects management**
   The possibility to handle more projects within one instance of tracking system will be discussed

5. **Analysis and monitoring**
   The kinds of statistics, graphs, and other monitoring resources that are provided by the tracking system will be shown.

6. **Source code management systems integration**
   Here, a short summary will be provided about the possibility to connect various versions of control systems (SVN, CVS, etc.) with the tracking system.

7. **Accessibility and extensibility**
   The reader will see the possibilities of extending basic functionality of the tracking system by plug-ins, and learn about its accessibility from other systems.

8. **Specialties**
   This part will summarize additional value against other systems, as is either advertised by developers of the tracking system, or as has been displayed throughout this investigation.

Special care will be taken about the JIRA issue tracking system. In the author's opinion, this tracking system is the winner of this "competition", and it has been decided to use it as much as possible within NetBeans IDE.

3.1 Bugzilla

Homepage:	http://www.bugzilla.org/
License:	Open source
Implemented in:	Perl
Demo site:	http://landfill.bugzilla.org/ (registration required)

Bugzilla is probably the most well-known of the open source issue-management tools. It is used on many open source projects such as Mozilla, Eclipse, and many Linux distributions, and is well-adapted to large, open projects.

Installation

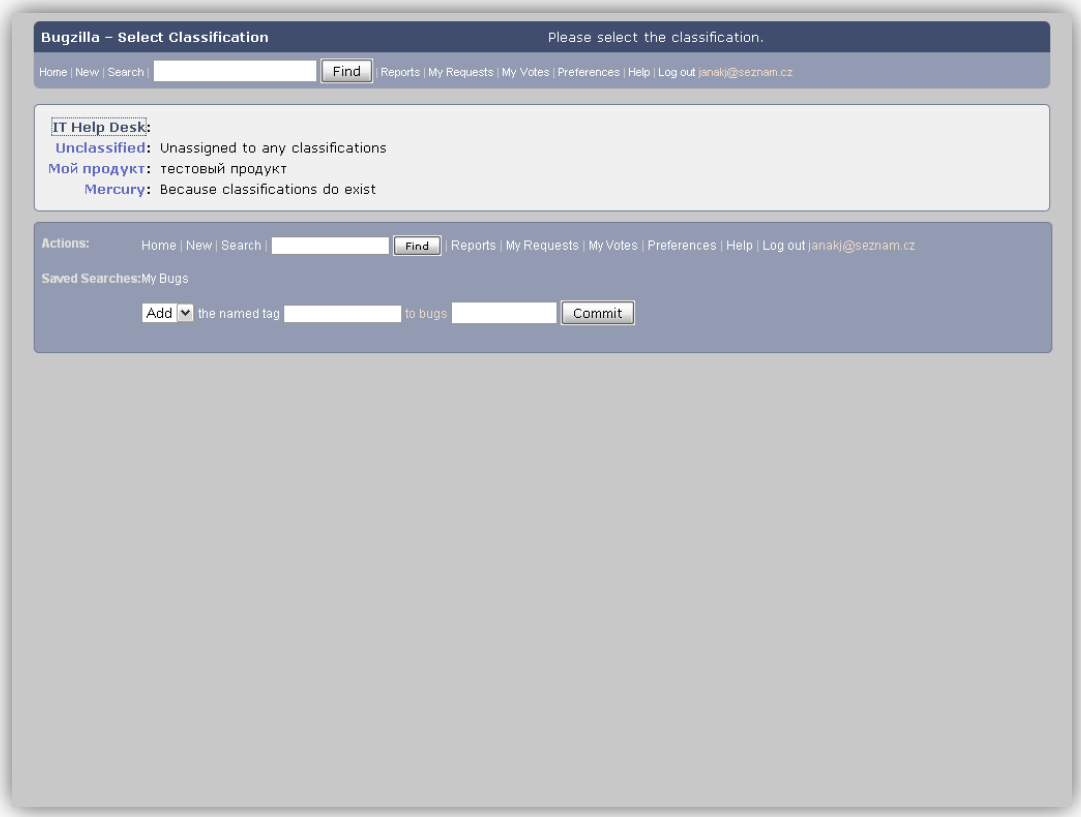
Bugzilla itself is distributed as a gZip package. Before installing Bugzilla itself, the user must manually prepare a running environment – install Perl, a database (MySQL or PostgreSQL), and a web server with Perl module support (the most common is Apache HTTPD server with a mod_perl module). Afterwards, the user unpacks the downloaded gZip package to a chosen directory and edits configuration files of the web server and configuration files of the application.

[TSO-GAR]

User interface

Easily said, Bugzilla UI is strictly functional. There is nothing very nice about it, it provides plenty of functions within a small space, and in the beginning, the user can feel quite uncomfortable and lost; however, after discovering it, the user will find out that it is not very complicated and working with it is straightforward.

Screen shot 1: Bugzilla use-case - Main Page



Screen shot 2: Bugzilla use-case - New bug report (select Classification)

Bugzilla – Enter Bug: Printers

Home | New | Search | Find | Reports | My Requests | My Votes | Preferences | Help | Log out janakj@seznam.cz

Before reporting a bug, please read the bug writing guidelines, please look at the list of most frequently reported bugs, and please search for the bug.

Product: Printers
Reporter: janakj@seznam.cz
Component: Voucher
Component Description: Voucher
Version: unspecified
Severity: normal
Platform: PC
OS: Windows XP
Target Milestone: —

We've made a guess at your operating system and platform. Please check them and make any corrections if necessary.

Initial State: NEW
Assign To: 123steel@gmail.com
QA Contact:
CC:
Default CC:

Flags: Requestee:
another-flag
another-flag2
blocker
regression
test

Estimated Hours: 0.0
Deadline: (YYYY-MM-DD)

Alias:
URL: http://
Custom Field:
Server Farm: East Coast, WestCoast
Color: —
blocking-fennec-1.0: -
Summary:
Description:

Initial Description is Private

Attachment: Add an attachment
Keywords: (optional)
Depends on:
Blocks:

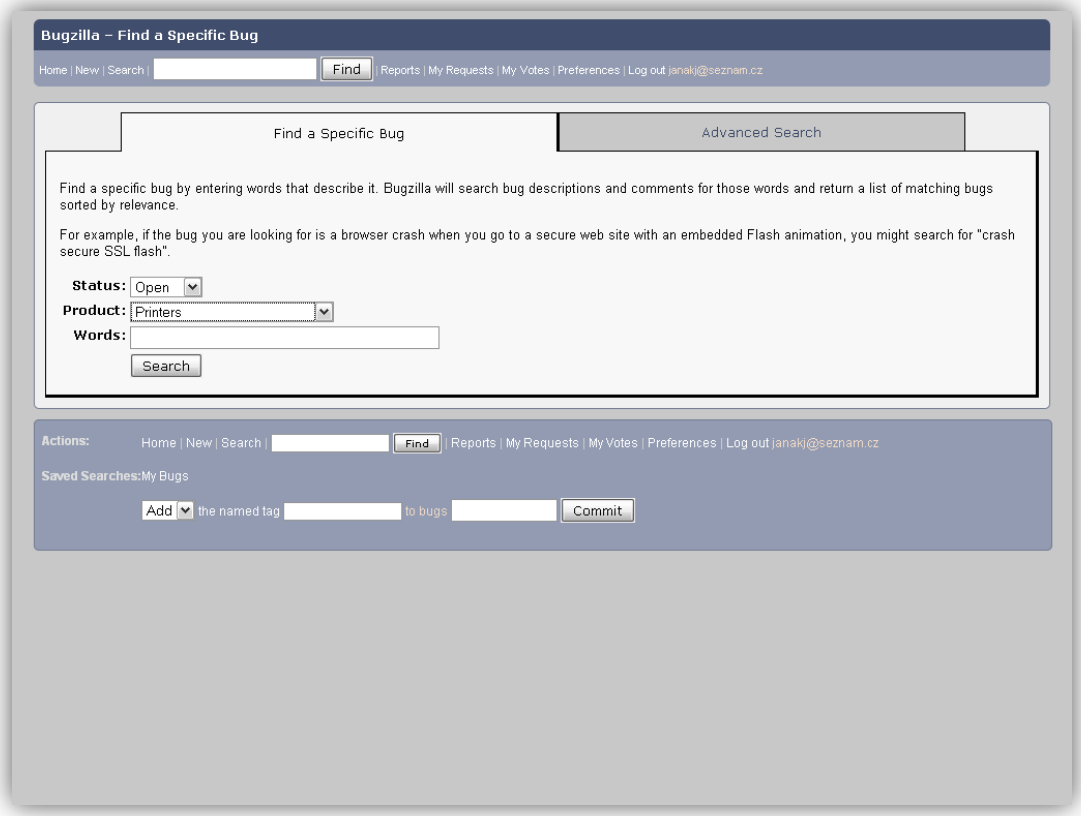
Commit | Remember values as bookmarkable template

Actions: Home | New | Search | Find | Reports | My Requests | My Votes | Preferences | Help | Log out janakj@seznam.cz

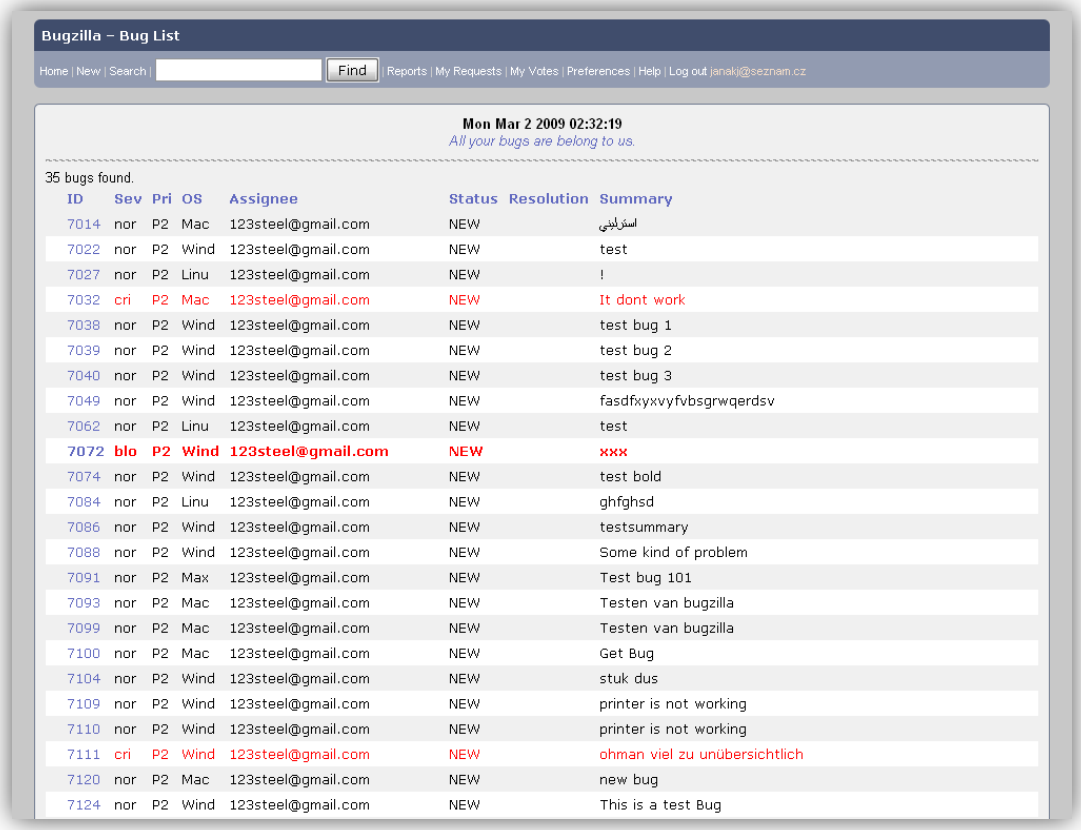
Saved Searches: My Bugs

Add the named tag to bugs Commit

Screen shot 3: Bugzilla use-case - New bug report (enter details)



Screen shot 4: Bugzilla use-case - Simple Search



ID	Sev	Pri	OS	Assignee	Status	Resolution	Summary
7014	nor	P2	Mac	123steel@gmail.com	NEW		استرلينى
7022	nor	P2	Wind	123steel@gmail.com	NEW		test
7027	nor	P2	Linu	123steel@gmail.com	NEW		!
7032	cri	P2	Mac	123steel@gmail.com	NEW		It dont work
7038	nor	P2	Wind	123steel@gmail.com	NEW		test bug 1
7039	nor	P2	Wind	123steel@gmail.com	NEW		test bug 2
7040	nor	P2	Wind	123steel@gmail.com	NEW		test bug 3
7049	nor	P2	Wind	123steel@gmail.com	NEW		fasdfxyxvyfvbsgrwqerdsv
7062	nor	P2	Linu	123steel@gmail.com	NEW		test
7072	blo	P2	Wind	123steel@gmail.com	NEW		xxx
7074	nor	P2	Wind	123steel@gmail.com	NEW		test bold
7084	nor	P2	Linu	123steel@gmail.com	NEW		ghfghsd
7086	nor	P2	Wind	123steel@gmail.com	NEW		testsummary
7088	nor	P2	Wind	123steel@gmail.com	NEW		Some kind of problem
7091	nor	P2	Max	123steel@gmail.com	NEW		Test bug 101
7093	nor	P2	Mac	123steel@gmail.com	NEW		Testen van bugzilla
7099	nor	P2	Mac	123steel@gmail.com	NEW		Testen van bugzilla
7100	nor	P2	Mac	123steel@gmail.com	NEW		Get Bug
7104	nor	P2	Wind	123steel@gmail.com	NEW		stuk dus
7109	nor	P2	Wind	123steel@gmail.com	NEW		printer is not working
7110	nor	P2	Wind	123steel@gmail.com	NEW		printer is not working
7111	cri	P2	Wind	123steel@gmail.com	NEW		ohman viel zu unübersichtlich
7120	nor	P2	Mac	123steel@gmail.com	NEW		new bug
7124	nor	P2	Wind	123steel@gmail.com	NEW		This is a test Bug

Screen shot 5: Bugzilla use-case - Bug List

Bugzilla – Bug 7022 test Last modified: 2009-01-06 00:40:15

Home | New | Search | Find | Reports | My Requests | My Votes | Preferences | Help | Log out janakj@seznam.cz

Bug List: (2 of 35) First Last Prev Next Show last search results

Bug 7022 - test (edit)

Status: NEW (edit)
Product: Printers
Component: Voucher
Version: unspecified
Platform: PC Windows XP
Importance: P2 normal
Target Milestone: ---
Assigned To: Jon (edit)
QA Contact:
Reset QA Contact to default
URL:
Whiteboard:
Keywords:
Depends on:
Blocks:
Show dependency tree / graph

Reported: 2009-01-06 00:40 by sl
Modified: 2009-01-06 00:40 (History)
CC List: Add me to CC list
0 users (edit)
Custom Field:
Server Farm: East Coast WestCoast
Color: Red
blocking-fennec-1.0: —
Flags:
sl_chua: another-flag +
sl_chua: another-flag2 -
sl_chua: blocker +
sl_chua: regression ?
sl_chua: test +
addl. another-flag
addl. another-flag2
addl. test

Orig. Est.	Current Est.	Hours Worked	Hours Left	%Complete	Gain	Deadline
0.0	0.0	0.0 + 0	0.0	0	0.0	(YYYY-MM-DD)

Summarize time (including time for bugs blocking this bug)

Attachments
Add an attachment (proposed patch, testcase, etc.)

Additional Comments: Private

Status: NEW Commit
Mark as Duplicate

Collapse All Comments - Expand All Comments

Description From sl 2009-01-06 00:40:15 (-) [reply] Private
test

Bug List: (2 of 35) First Last Prev Next Show last search results
Format For Printing - XML - Clone This Bug - Top of page

Actions: Home | New | Search | Find | Reports | My Requests | My Votes | Preferences | Help | Log out janakj@seznam.cz
Saved Searches: My Bugs
Add the named tag to bugs 7022 Commit

Screen shot 6: Bugzilla use-case - View/Edit bug report

Ticket system

Bugzilla supports a fairly complete, albeit hard-coded workflow model (see diagram below). Although this workflow model cannot be customized in Bugzilla, it usually proves sufficient for most organizations.

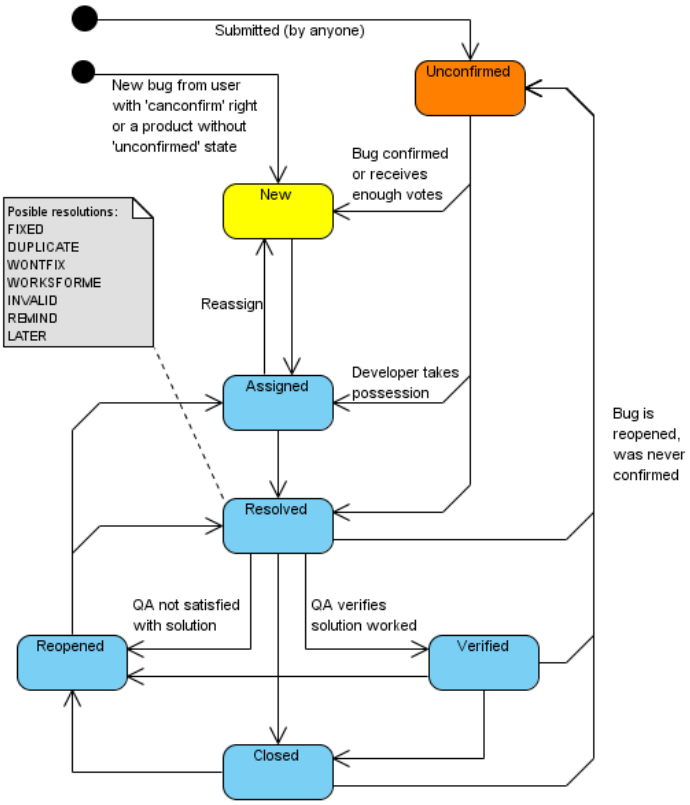


Figure 6: Bugzilla ticket workflow

Bugzilla also supports adding custom fields to a bug database, to capture and search data that is unique to the organization.

[TSO-MOZ1]

Projects management

Bugzilla allows administrators to create and modify so called "Classification trees" of products. This means there is no problem to create a tree where, for example, the first level would be names of projects, and second level would be names of products being edited. No special hacks are required.

Analysis and monitoring

Bugzilla has a very advanced reporting system, which is at first look quite hard to start to use – the user interface is not very friendly.

To see how the bug database currently looks like, a table can be created using any two fields as the X and Y axis, and using any search criteria to limit the bugs in question. For example, Product could be represented by the X axis, and Status by the Y axis, and then a report would show how many bugs were in each Status, in each Product.

The same table can be viewed as a line graph, bar graph, or pie chart. A "Z axis" can also be specified to generate multiple tables or graphs. These reports can be exported as CSV, so that they can be edited in a spreadsheet. Finally, to see how the Bugzilla installation has changed over time, Bugzilla also supports a charting system, which can create graphs that track changes in the system over time.

[TSO-SMA]

Source code management systems integration

Bugzilla integration with source code management systems is provided via plug-ins. CVS, Subversion, Bonsai, Teamtrack Performance and Tinderbox are officially supported.. Those and others can be found on the Add-ons resources page.

Accessibility and extensibility

Bugzilla can be accessed and modified by an XML-RPC web services interface. This makes it possible to write external tools that interact with Bugzilla easily.
There are also many plug-ins and extensions for Bugzilla. A list of them can be found at https://wiki.mozilla.org/Bugzilla:Addons.

[TSO-MOZ2]

Specialties

- Bugzilla UI is generated from HTML templates; therefore, it is possible to customize UI quite easily.
- There is wide variety of localizations available.
- Bugzilla notifies users of any new or updated bugs by e-mail.
- Bugzilla supports basic time tracking.
- Bugzilla also supports a system of votes, in which users can vote for issues or features they wish to see implemented.

[TSO-MOZ1] [TSO-SMA] [TSO-WIK1]

Conclusion

Bugzilla is a tried-and-true solution that supports large projects and user bases. Its workflow features are more than sufficient for most organizations. On the downside, Bugzilla is particularly complicated to install and maintain, and the user interface would not win any prizes for design or usability. Its reporting features will do the job, though they are not particularly user-friendly.

3.2 CodeBeamer

Homepage:	http://www.intland.com/products/codebeamer.html
License:	Proprietary (Free – 500 USD)
Implemented in:	Java
Demo site:	https://codebeamer.com/cb/user

CodeBeamer is not just a simple Issue tracking system, but it is a Collaboration Development Platform with integrated Application Lifecycle Management. Under this title we can imagine set of services such as Project related Document Management, Wiki, Forum, Online chat and of course Issue tracking integrated with version control (SVN).

It's available for free for students and evaluators; however for business use it is commercial. As an example of public use we can name for example JavaForge.com community, which host multiple Java open-source projects and is running above CodeBeamer.

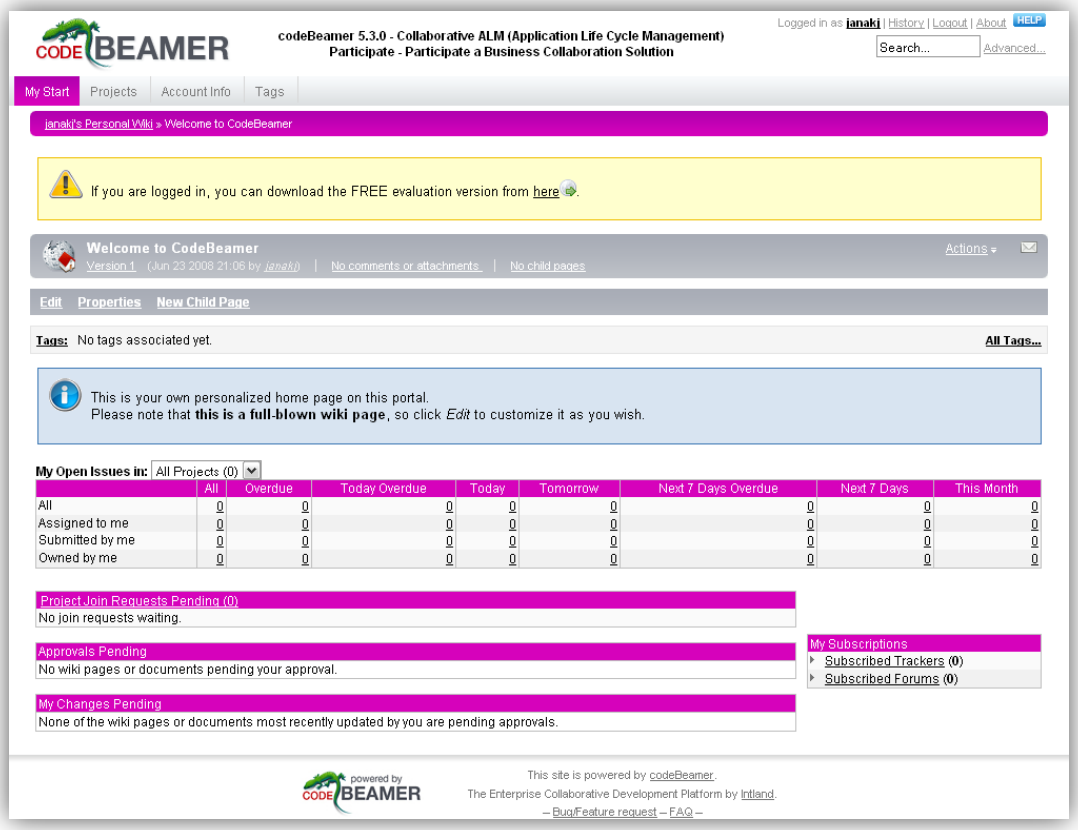
Installation

CodeBeamer is a Java EE application running on Apache Tomcat servlet container. It is delivered as a binary installation file either for Windows or for Linux. Tomcat installation files are included in the package; however, there is no problem to install CodeBeamer to your own instance of Tomcat. After finishing the installation, it is necessary to set a connection to the database (and some other configuration) in property file.

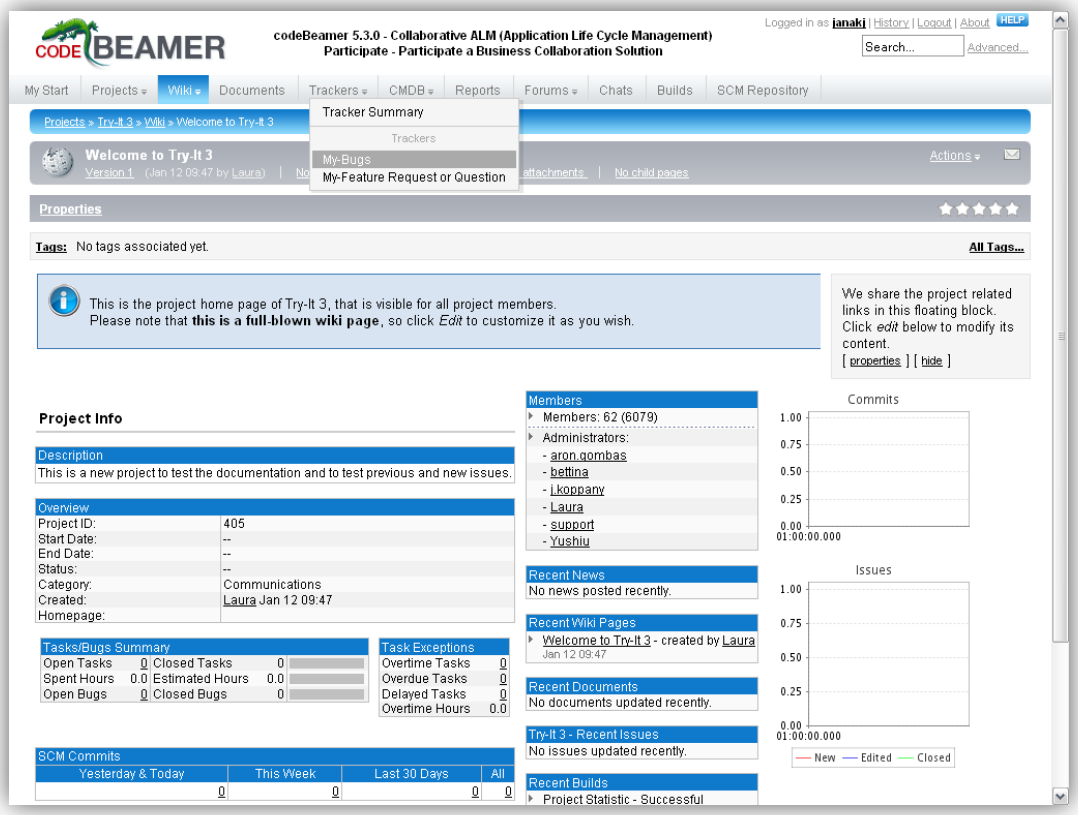
[TSO-INS2]

User interface

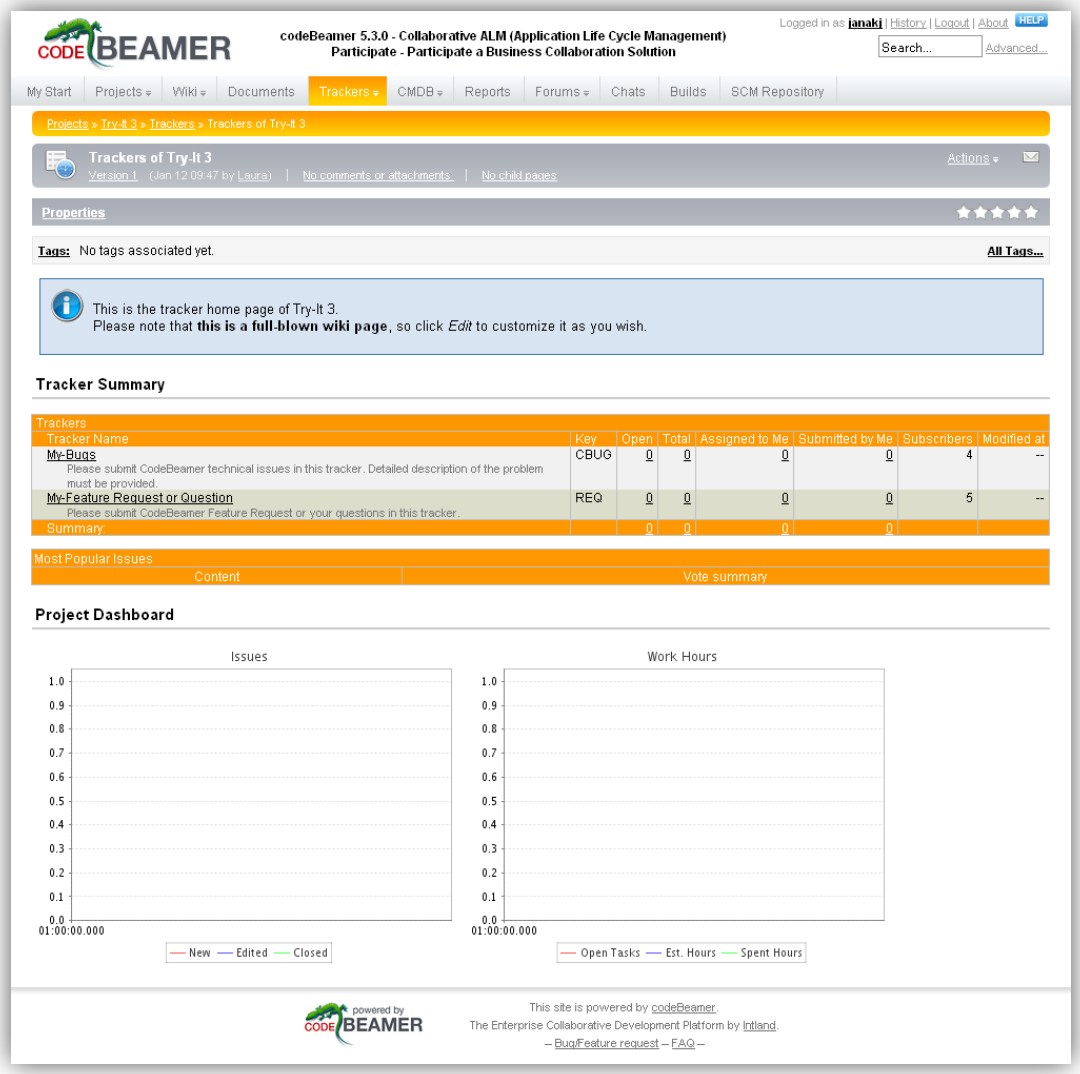
The user interface looks to have a better structure than Bugzilla UI; however, it still would not win the price for usability. As you can see below in the use-case scenario, it is necessary to go quite deep into the UI to add new issue requests. It would help to move some most used functions to some extra panels, or other similar reorganizations. While using CodeBeamer UI, there was no problem to find anything, only many things involved too many steps.



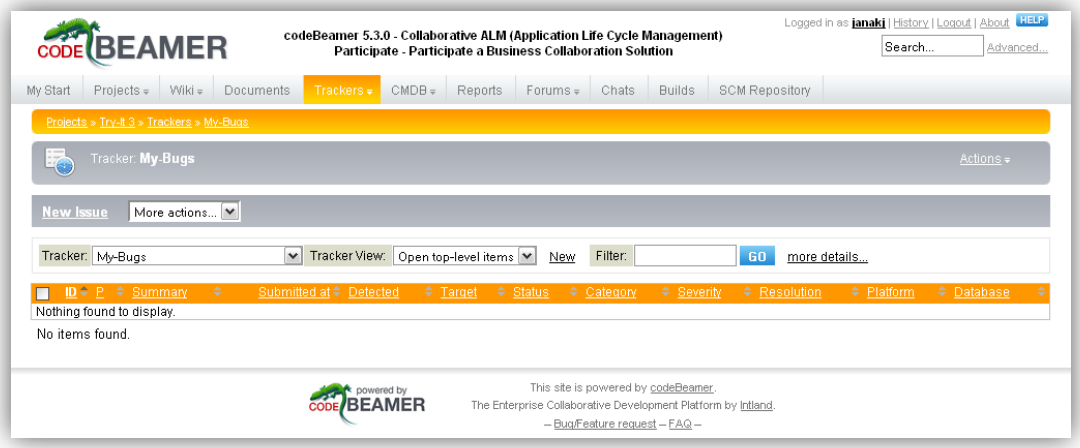
Screen shot 7: CodeBeamer use-case - My Start (personal Main page)



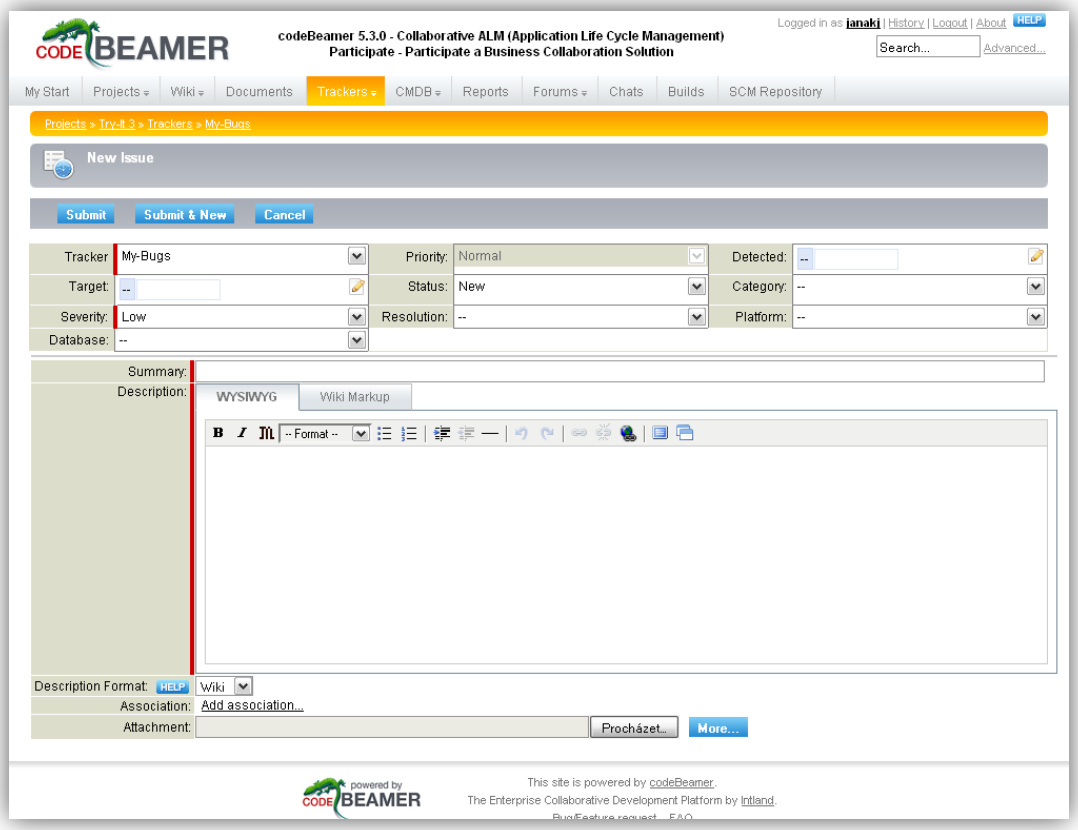
Screen shot 8: CodeBeamer use-case - Project dashboard



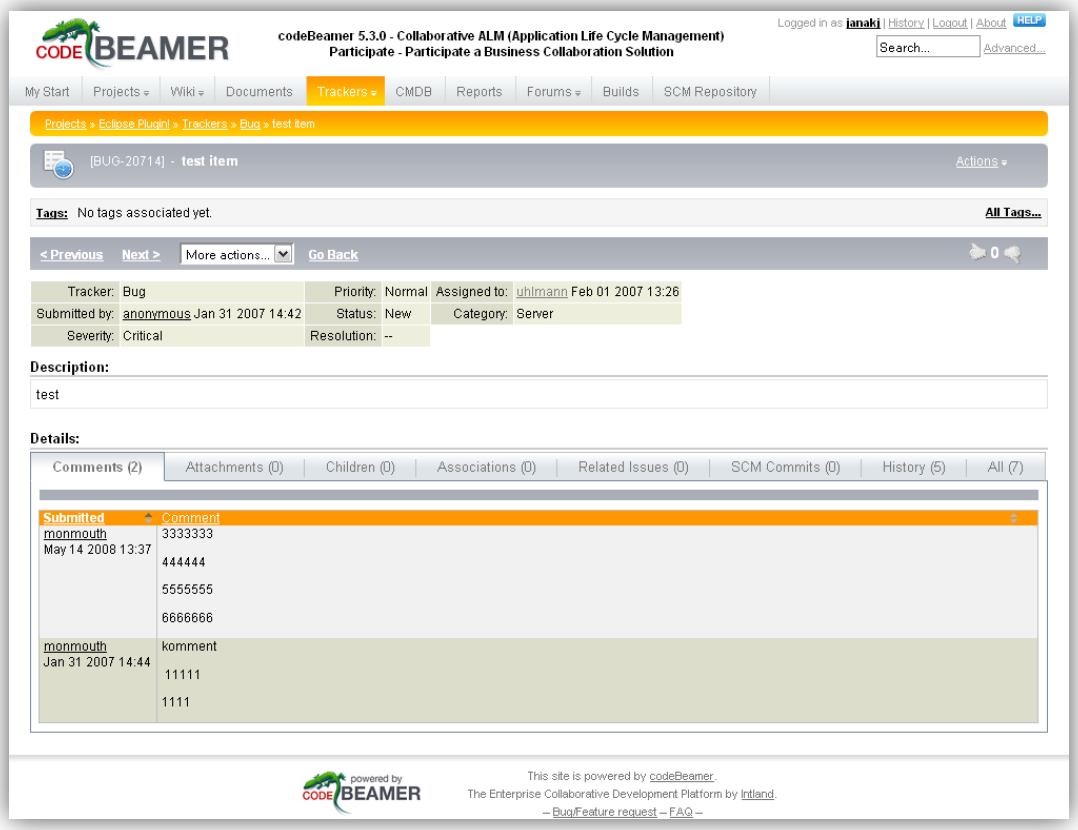
Screen shot 9: CodeBeamer use-case - Project trackers dashboard (select tracking database)



Screen shot 10: CodeBeamer use-case - Bug list



Screen shot 11: CodeBeamer use-case - Add new issue details



Screen shot 12: CodeBeamer use-case - View issue

Ticket system

CodeBeamer provides a default workflow, which can be customized within the user interface. A guide can be found on the CodeBeamer Wiki page "Workflows" (http://cb.esast.com/cb/wiki/5713).

The workflow diagram below contains the following node types:

- Yellow node - denotes the start state. A new tracker item will appear in this status.
- Blue nodes - denote the regular states. It will move among these during most of the tracker item life cycle.
- Orange nodes - as there are only leaving transitions for these, one cannot move items to them, but can move items from these to the regular items. These nodes are primarily used for proper migration from legacy trackers.

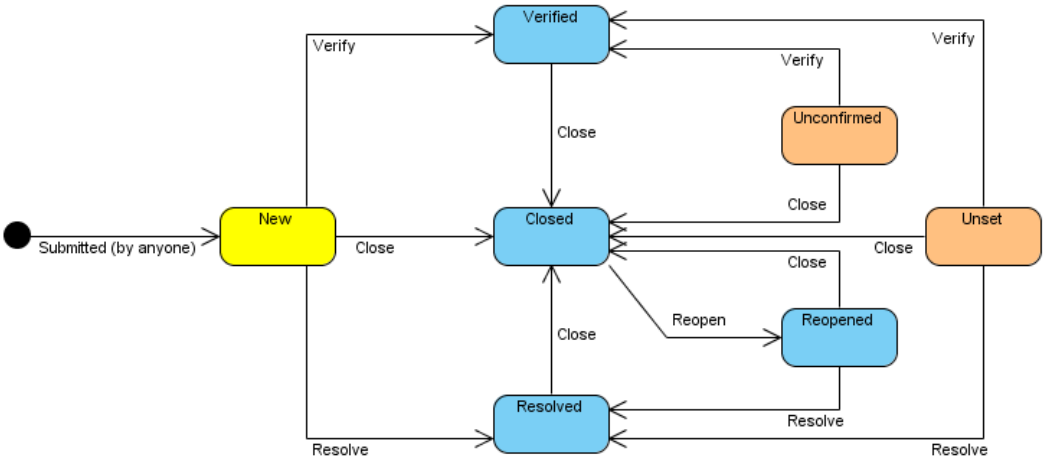


Figure 7: Default CodeBeamer ticket workflow

CodeBeamer also supports adding of custom fields to the tickets within the user interface.

[TSO-INS2]

Projects management

CodeBeamer, by default, supports the management of multiple projects. Projects can be also grouped in so called "Working sets," which enable their easier management. So, for example, projects can be sorted into working sets named after the customer for whom they are, and so on.

Each project can have its own settings, access rights, ticket workflow, etc. It works such that project settings are applied in first place, after this, working set settings are taken (if the project is in some working set), and finally, default CodeBeamer settings.

Analysis and monitoring

CodeBeamer has configurable reporting systems. By default it provides:

- Real Time Task and Bug Reporting with charts;
- A Gantt chart;
- Source code quality and code change reports with trends;
- Reports with custom reports, allowing increased flexibility and automatic generation of release notes and other reports; and
- Built-in CMMI reports.

Reports work on a list of tracker items from one or more trackers belonging to one or more projects. Simple reports contain one table for each tracker. Merged reports present the tracker items in a single unified tabular layout.

After running a report, the results can be exported into Excel, PDF, CSV, XML and Wiki formats.

Source code management systems integration

CodeBeamer has built-in interfaces for CVS, Subversion, PVCS, SourceSafe, CM Synergy, and ClearCase SCM systems and can easily be extended to support other systems.

Accessibility and extensibility

CodeBeamer has Web Services API based on the Caucho Hessiann binary web service (http://www.caucho.com/hessian/).

Specialties

As mentioned in the beginning, CodeBeamer is not just a simple issue tracking system, but it is a Collaboration Development Platform with integrated Application Lifecycle Management. This introduces the following features:

- A built-in Document Management System;
- An internal enterprise Wiki system, which integrates all components such as tracker, document manager, SCM system together into a single application, where all artifacts can be linked and referenced with Wiki markups;
- Support for periodical builds with continuous integration;
- Built-in discussion forums;
- An advanced security platform;
- E-Mail integration with inboxes; and
- Support for providing quality assurance statistics and metrics.

[TSO-COS] [TSO-INS1] [TSO-INS2]

Conclusion

CodeBeamer is just that what it calls itself to be: Collaboration Development Platform. It provides a variety of tools to support both management and development, and in addition, provides also its integration plug-ins into most common IDEs (this will be mentioned in next chapter). It is also easily configurable and maintainable. The UI is seen as a small minus, where the most common tasks are not topped and remain "hidden" in their categories – then can be found when one searches, but they are not evident.

3.3 IBM Rational ClearQuest

Homepage:	http://www.ibm.com/software/awdtools/clearquest/
License:	Proprietary (1720 – 4810 USD)
Implemented in:	Perl / Visual Basic
Demo site:	http://www.ibm.com/developerworks/downloads/r/rcq/tryonline.html (registration required)

Rational Software is one of the IBM software brands, which includes solutions and tools designed to facilitate application development. Its portfolio covers the entire software development cycle, from requirements gathering, design and modeling through to coding and testing to the project management, change management, and quality assurance. Part of IBM Rational is the **Rational Unified Process methodology (RUP),** which became a de-facto reference model for the management of large software projects.

Rational ClearQuest is a flexible tool for tracking requirements and changes across the software development process. It offers a complete and fully customizable environment for integration and monitoring of any changes during software development. The RUP concept presupposes using Unified Change Management tools at all stages of the life cycle of the project; therefore, there are wide possibilities of integration of ClearQuest with other Rational tools.

[MIT-WIK2]

The difference between this and other issue tracking tools is that application is primary designed as a fat client, which is directly connected to the databases in which schemas and user data are stored. The Microsoft Windows platform is an available native client. If an alternative platform is favoured, the Eclipse platform based client, Eclipse plug-in, or configuring web-based access can be executed.

Installation

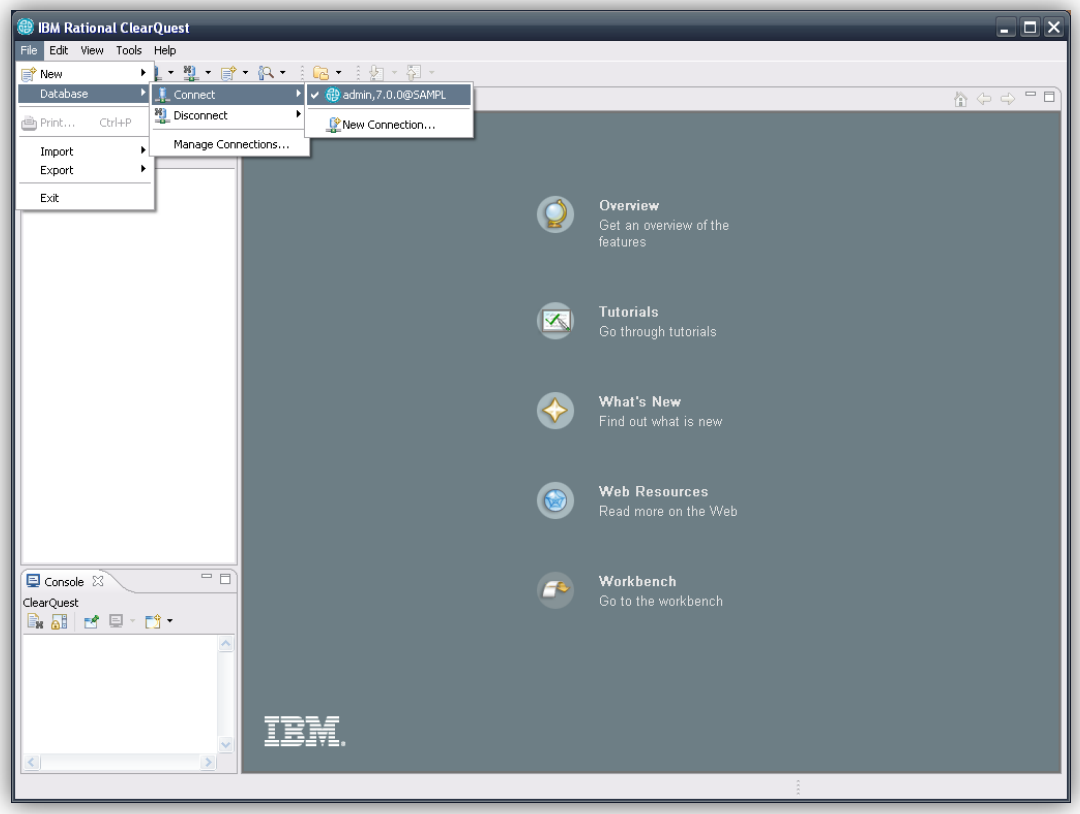
ClearQuest is delivered as a Java-based binary installation file, which supports either GUI-based installation (more friendly for desktops) or text-based installation (more friendly for remote servers). No special preparations are necessary; the installation file includes all necessary parts.

[TSO-IBM1]

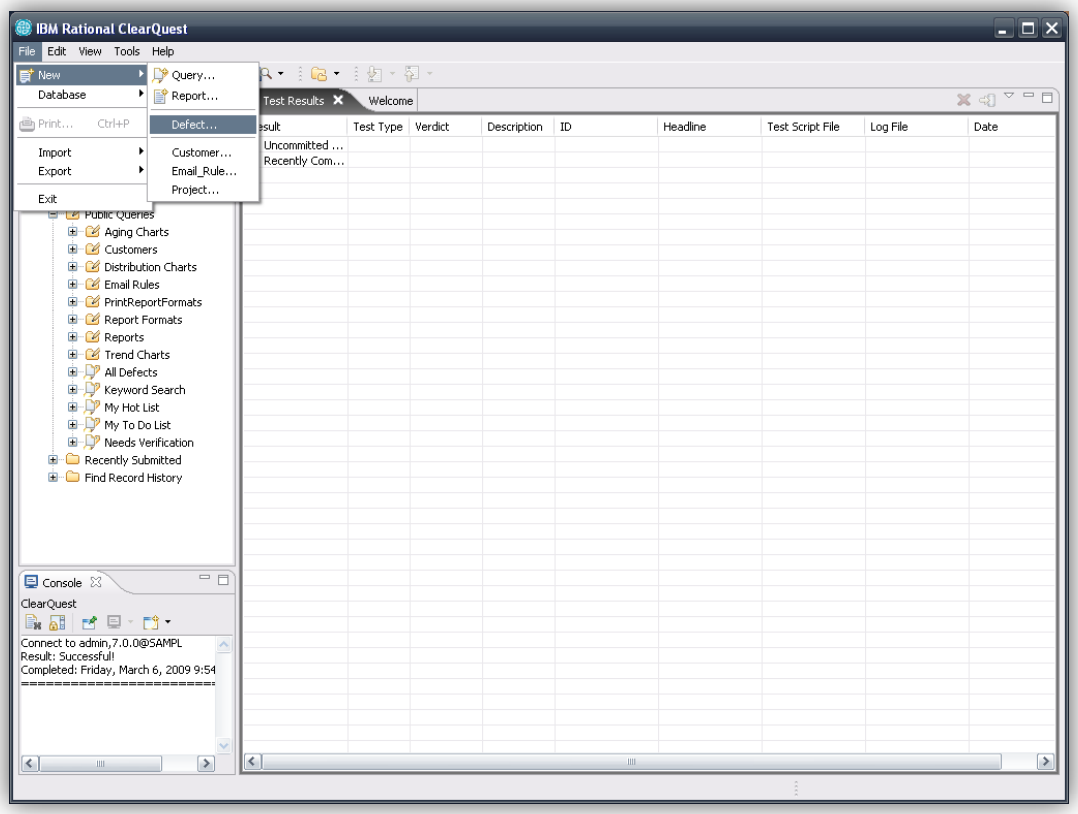
User interface

ClearQuest is primary designed as a desktop client which connects to databases. This can be a little bit confusing for people used to web-based Issue tracking systems. Using it is like using an ordinal application (i.e. a text editor, etc.) For example, to create a new ticket, one must choose New -> Defect, the right mouse button is used to get a list of actions available with the things seen, and so on; however, even after the author did not feel comfortable within the system. Even on web forums, ClearQuest has been criticized for its large resource footprint, sluggish performance, and sub-par usability and design.

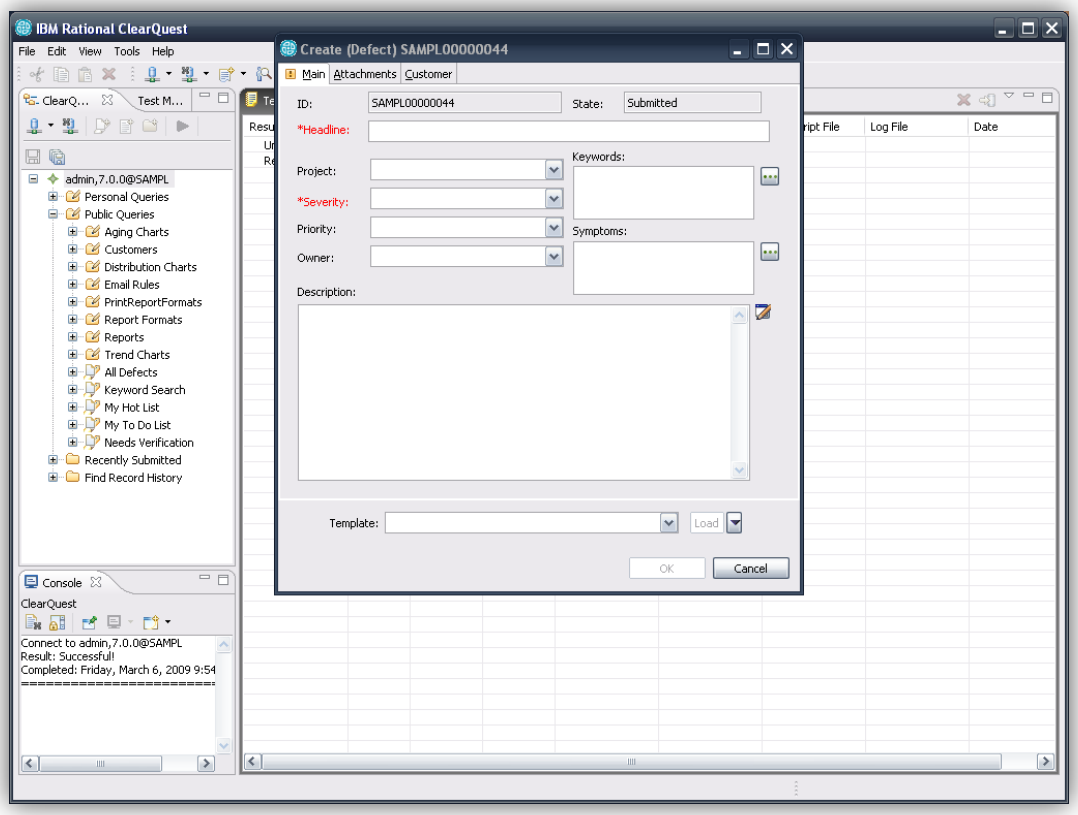
As mentioned above, except for the Eclipse platform-based client, there is also a native Windows client and web-based client. All three clients are quite similar in appearance, so a user should not have any problems with changing from one to another. The web version offers all the functionality of the full version except for the ability to create queries.



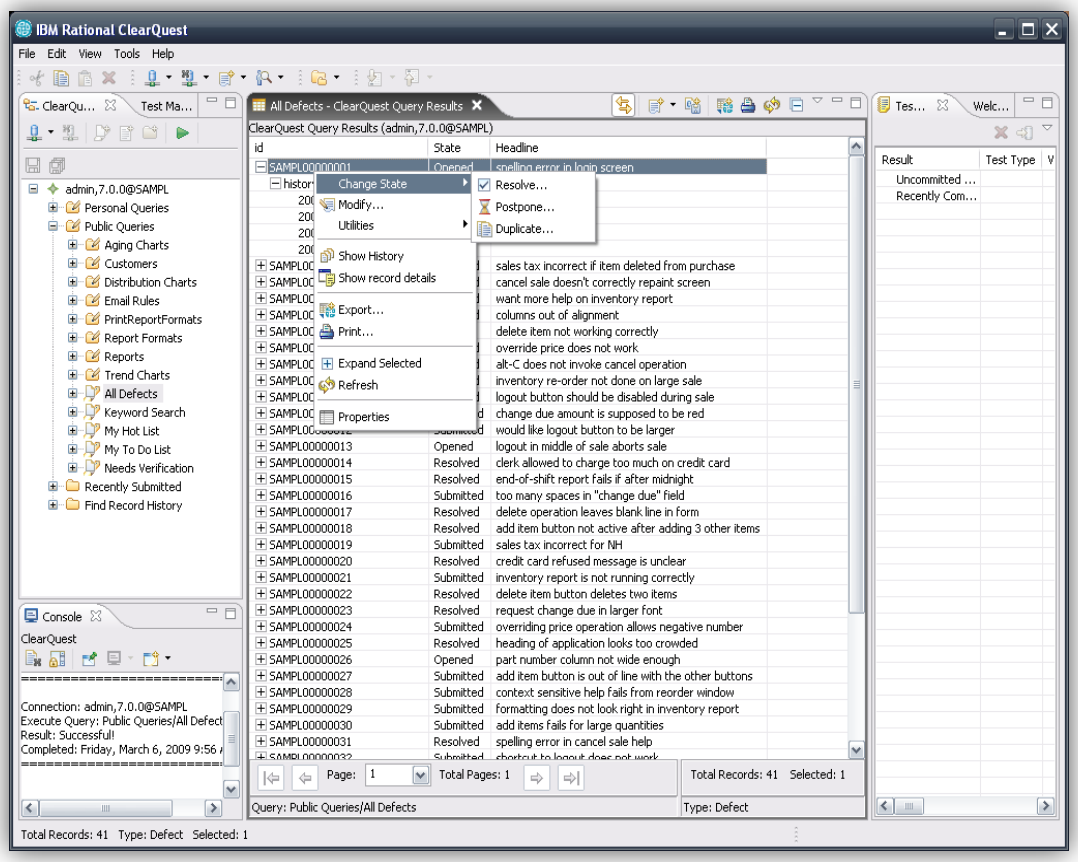
Screen shot 13: ClearQuest use-case - Login to selected requests database



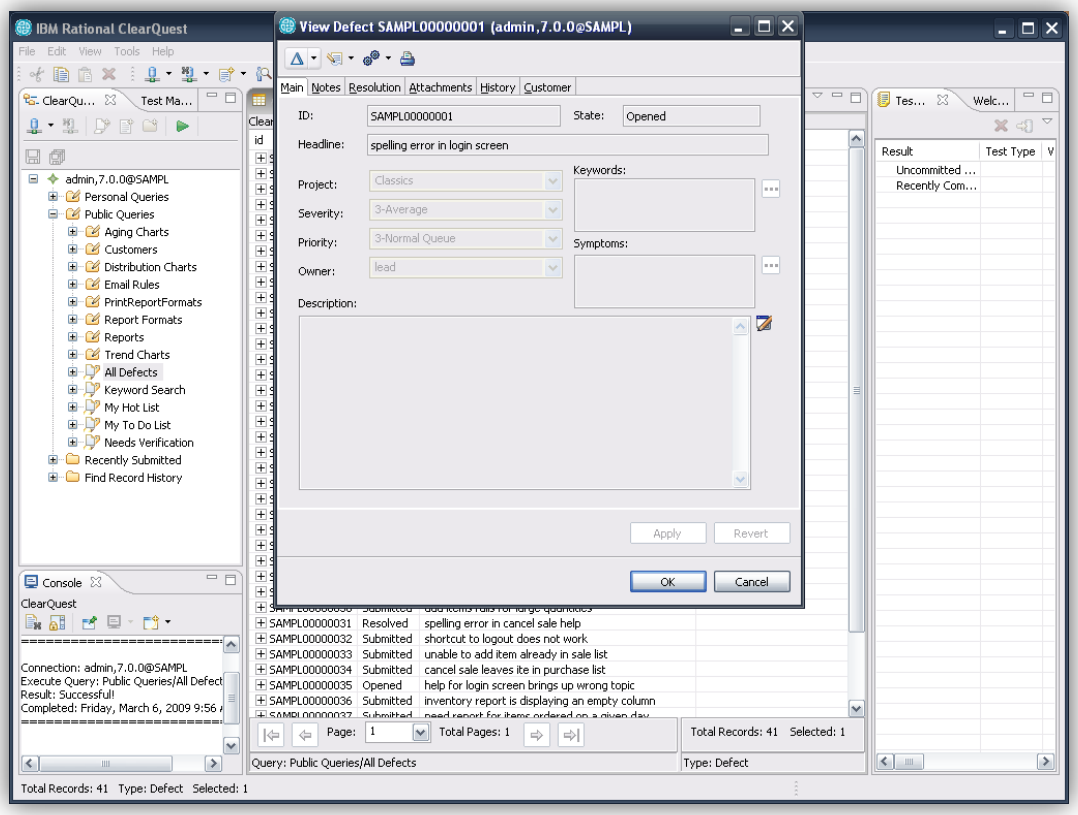
Screen shot 14: ClearQuest use-case - Create new defect ticket



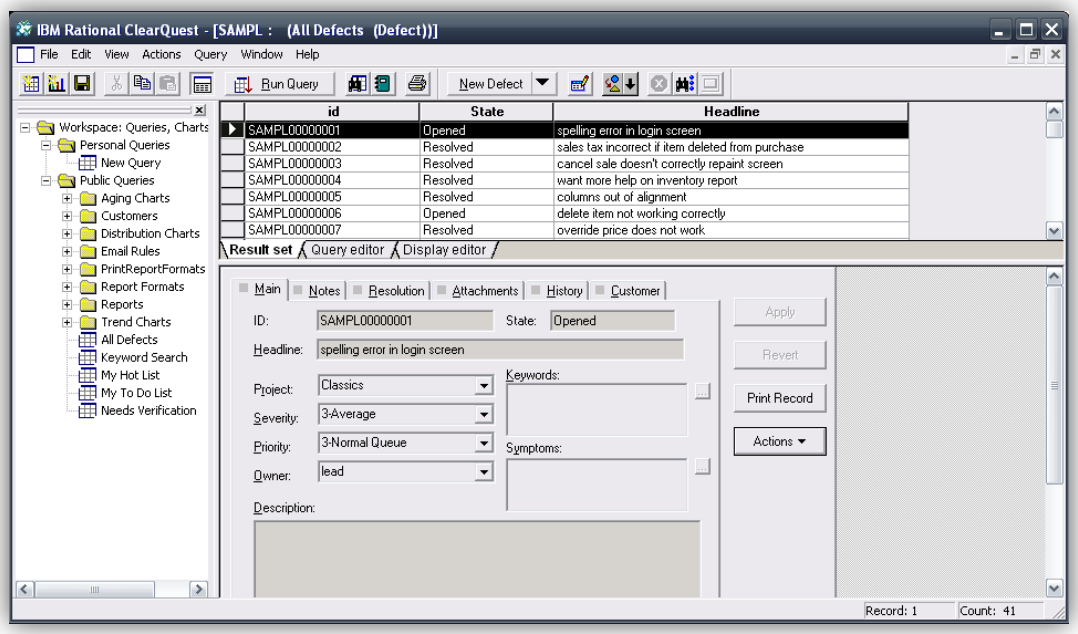
Screen shot 15: ClearQuest use-case - Add new defects details



Screen shot 16: ClearQuest use-case - List of defects (and enabled actions)



Screen shot 17: ClearQuest use-case - View defects details



Screen shot 18: ClearQuest client for MS Windows

Ticket system

By default, the ticket workflow keeps the recommendation by RUP, as you can see below.

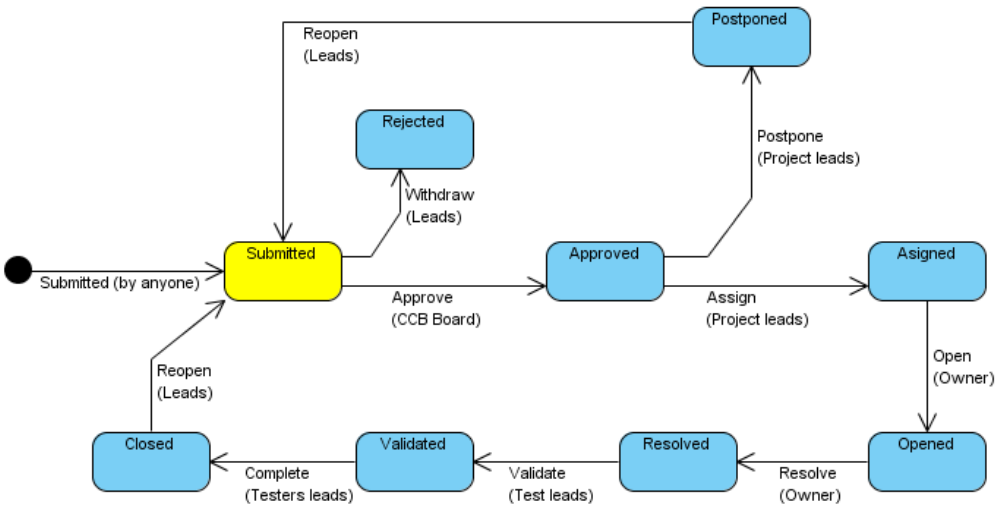
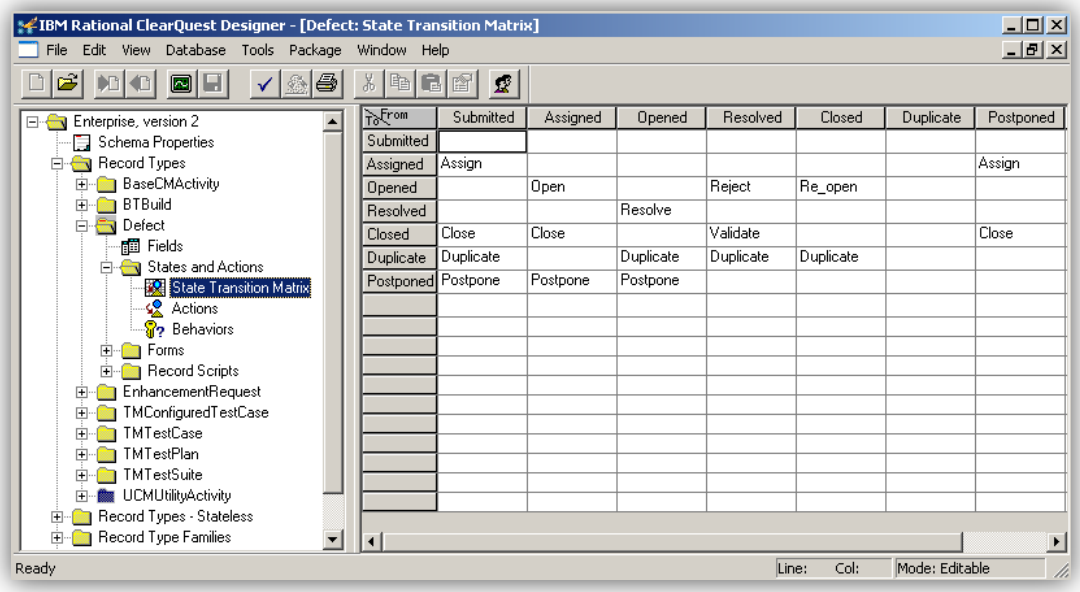


Figure 8: Default IBM Rational ClearQuest ticket workflow

[TSO-IBM1]

However, it is really easy to modify it using the ClearQuest Designer application.



Screen shot 19: ClearQuest Designer - modifying ticket workflow

ClearQuest also supports Perl scripting, which allows more sophisticated configuration and automation of activities within the system.

Projects management

ClearQuest is designed as a multi-project tracking software. For each project, one can simply create a new database with its own configuration, ticket workflow, access rights, etc.

Analysis and monitoring

Analysis and monitoring skills of ClearQuest are top quality. Perhaps thanks to the RUP methodic behind it, ClearQuest offers plenty of report and graph templates, all of them easily customizable. There are "UI wizards" (step-by-step guides), which enable even a beginner user to create his or her own custom queries and reports; however, as mentioned, the most frequently used templates are there by default.

Source code management systems integration

By default, ClearQuest supports close integration with Rational ClearCase. ClearCase is a software tool for revision control of source code and other software development assets developed by the Rational division of IBM.

There is also possible integration with other source code management systems; however, installation of 3rd party plug-ins is necessary.

Accessibility and extensibility

IBM provides Rational Team API - a unified, client-side Java API for ClearCase and ClearQuest (http://www.alphaworks.ibm.com/tech/teamapi/).

"In order to use this API, customers must have installed Version 7.0.0 or 7.0.1 of Rational ClearQuest client or ClearCase Web server. That software installs the required Rational Team API JAR files that are used by the customer's Java classes to access the respective subprovider's repositories. The Rational Team API JAR files contain implementations of the Rational Team API interfaces for each product. For ClearQuest, the Java classes call into existing Java Native Interfaces (JNIs). For ClearCase, the Java classes make HTTP calls to the ClearCase Web Server. These libraries, in turn, access each product's repository."

IBM Ration Team API webpage
[TSO-IBM2]

ClearQuest UI is based on the Eclipse RPC platform what makes it easily extensible. ClearQuest itself supports Perl scripting, which allows for more sophisticated configuration and automation of activities within the system.

Specialties

- Tight integration with other Rational family products;
- Automatic notification of events and changes via e-mail;
- Define and run queries and generate reports and analysis as distribution and time graphs;
- Response to events and changes by hooks;
- Creation and execution of test plans and scenarios and result reports via the integrated TestManager (CQTM);
- Supports integration with project management tools MS Project a Rational Portfolio Manager; and
- Supports integration with Microsoft Visual Studio .NET.

[TSO-IBM1] [TSO-WIK2]

Conclusion

As it is, Rational ClearQuest is just another Issue tracking tool with excellent reporting and monitoring possibilities; however, if it is put into the Rational tools-based development environment, it becomes a very powerful issue tracking tool with strong integration to other Rational tools.

3.4 Trac

Homepage:	http://trac.edgewall.org/
License:	Open source
Implemented in:	Python
Demo site:	http://www.hosted-projects.com/trac/TracDemo/Demo

Trac is maybe best known and most used open source (and free) issue tracking system, next to Bugzilla. It offers a very intuitive and fine looking user interface, close junction with version control systems (mostly Subversion), revision control, and wiki for content. Due to these factors, it is widely spread in the open source community and used in many small teams and companies. There are many sites (for example Assembla.com) which offer Trac & Subversion hosting for free.

Installation

Trac is distributed as a zipped package. Before installation of Trac itself, the user has to manually prepare the running environment and install:

- Python;
- Genshi (XML templating language for Python);
- SetupTools (collection of enhancements to more easily build and distribute Python packages);
- Database (Sqlite, MySQL or PostgreSQL);
- A web server with Python module support (most common is Apache HTTPD server with mod_python module) ; and
- A subversion (or other supported version control system).

After this, the user unpacks the downloaded package to the chosen directory and runs a Python-based Trac installer.

After installation is finished, the user configures Trac using a command line utility called "trac-admin". All changes in configuration need to be made by this utility, although for performing the most common admin tasks there is a possibility to install and use the plug-in "webadmin."

[TSO-EDG]

User interface

The Trac user interface is an excellent example of usability and simplicity. You can reach any target within three clicks of the mouse. There are not too many buttons and options, so even beginners can quickly understand how to use it.

The welcome page is a standard wiki page and can be used for project documentation. The whole system is based on wiki, so a user quickly understands that t is quite easy to hyperlink information between an issue database, revision control, and wiki content.

Use Demo/Demo to log in as the admin user

trac
Integrated SCM & Project Management

Search

Login | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

Start Page | Index by Title | Index by Date | Last Change

= Welcome to the Trac Sample Project! = i

Trac is a **minimalistic** approach to **web-based** management of **software projects**. Its goal is to simplify effective tracking and handling of software issues, enhancements and overall progress.

All aspects of Trac have been designed with the single goal to **help developers write great software** while **staying out of the way** and imposing as little as possible on a team's established process and culture.

As all Wiki pages, this page is editable, this means that you can modify the contents of this page simply by using your web-browser. Simply click on the "Edit this page" link at the bottom of the page. WikiFormatting will give you a detailed description of available Wiki formatting commands.

"trac-admin *yourenvdir* initenv" created a new Trac environment, containing a default set of wiki pages and some sample data. This newly created environment also contains documentation to help you get started with your project.

You can use trac-admin to configure Trac to better fit your project, especially in regard to *components*, *versions* and *milestones*.

TracGuide is a good place to start.

Enjoy!
The Trac Team

Starting Points

- TracGuide -- Built-in Documentation
- The Trac project -- Trac Open Source Project
- Trac FAQ -- Frequently Asked Questions
- TracSupport -- Trac Support
- Syntax Highlighting

For a complete list of local wiki pages, see TitleIndex.

Edit this page | Attach file | Delete this version | Delete page

Download in other formats:
Plain Text

trac | Powered by Trac 0.10.5 By Edgewall Software. | Visit the Trac open source project at http://trac.edgewall.org/

Screen shot 20: Trac use-case - Project welcome page (Wiki)

Use Demo/Demo to log in as the admin user

trac
Integrated SCM & Project Management

Search

logged in as Demo | Logout | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

Create New Ticket

Short summary:

Type: defect

Full description (you may use WikiFormatting here):

Ticket Properties

Priority: major
Component: asd-def
Severity: sev-00
Assign to:
Milestone: EVT stage
Version:
Keywords:
Cc:

I have files to attach to this ticket

Preview | Submit ticket

Screen shot 21: Trac use-case - Add new ticket

Use Demo/Demo to log in as the admin user

trac
Integrated SCM & Project Management

Search

logged in as Demo | Logout | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

Available Reports | Custom Query

Available Reports

This is a list of reports available.

Report	Title
{1}	Active Tickets
{2}	Active Tickets by Version
{3}	Active Tickets by Milestone
{4}	Assigned, Active Tickets by Owner
{5}	Assigned, Active Tickets by Owner (Full Description)
{6}	All Tickets By Milestone (Including closed)
{7}	My Tickets
{8}	Active Tickets, Mine first

Create new report

Note: See TracReports for help on using and creating reports.

trac POWERED | Powered by Trac 0.10.5 By Edgewall Software. | Visit the Trac open source project at http://trac.edgewall.org/

Screen shot 22: Trac use-case - View tickets (select searching criteria)

Use Demo/Demo to log in as the admin user

trac
Integrated SCM & Project Management

Search

logged in as Demo | Logout | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

Available Reports | Custom Query

{7} My Tickets (3 matches)

This report demonstrates the use of the automatically set USER dynamic variable, replaced with the username of the logged in user when executed.

Edit report | Copy report | Delete report

Owned

Ticket	Summary	Component	Version	Milestone	Type	Priority	Created
#18	Test	component1	None	milestone1	task	major	03/02/09
#19	Test	component1	None	milestone1	task	major	03/02/09
#20	Test	component1	None	milestone1	task	major	03/02/09

Note: See TracReports for help on using and creating reports.

Download in other formats:
XML RSS Feed | Comma-delimited Text | Tab-delimited Text | SQL Query

trac POWERED | Powered by Trac 0.10.5 By Edgewall Software. | Visit the Trac open source project at http://trac.edgewall.org/

Screen shot 23: Trac use-case - List of tickets

Use Demo/Demo to log in as the admin user

trac
Integrated SCM & Project Management

Search

logged in as Demo | Logout | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

Ticket #18 (new task)

Test Opened 5 hours ago

Reported by:	demo	Assigned to:	Demo
Priority:	major	Milestone:	milestone1
Component:	component1	Version:	
Severity:		Keywords:	
Cc:			

Attachments

Attach File

Add/Change #18 (Test)

Comment (you may use WikiFormatting here):

Change Properties

Summary: Test

Type: task

Description:

Reporter: demo

Priority: major

Milestone: milestone1

Component: component1

Version:

Severity: sev-00

Keywords:

Cc:

Action

- leave as new
- accept ticket
- resolve as: fixed
- reassign to: Demo

Preview | Submit changes

Download in other formats:
Comma-delimited Text | Tab-delimited Text | RSS Feed

trac POWERED | Powered by Trac 0.10.5 By Edgewall Software. | Visit the Trac open source project at http://trac.edgewall.org/

Screen shot 24: Trac use-case - View/Edit ticket details

Ticket system

When a new environment is created, a default workflow is configured in trac.ini. This workflow is the basic workflow (see diagram below).
This workflow model can be changed by overwriting trac.ini with other workflow model files prepared in advance or customized by a direct edit of this file.

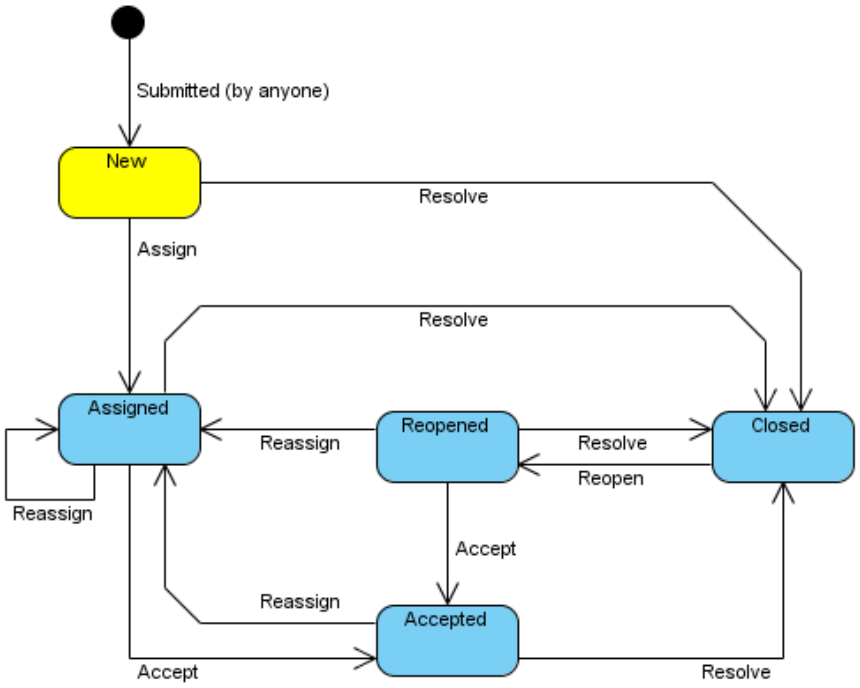


Figure 9: Default Trac ticket workflow

[TSO-EDG]

Trac also supports adding custom fields to the tickets by editing configuration file trac.ini; however, any tickets created before a custom field has been defined will not have a value for that field and it is necessary to manually update the database.

Projects management

Trac does not support management of multiple projects; therefore, having its own instance for each project is the only real working solution. This way, however, the possibility to monitor all projects at once, to manage tasks common for multiple projects, and others is lost.

Analysis and monitoring

In basic installation, Trac only supports basic monitoring functions. You can only split requests between releases and see how many requests have already been accomplished; however, there are plug-ins which, based on added custom fields, support additional monitoring.

Source code management systems integration

Trac natively supports version control systems Subversion, Git, Mercurial, Bazaar, and Darcs. For other systems, there are plug-ins available or can be made.
More details at http://trac.edgewall.org/wiki/VersioningSystemBackend.

Accessibility

Trac itself does not have any RPC support; however, there is an XML-RPC plug-in for Trac version 0.10 and higher, but there are two problems:

1. The first problem is that this plug-in is not included in the basic installation package and its installation requires some time.
2. The second problem is that the plug-in is still under development and its API still seems to be changing.

If one cannot count with presence of plug-in and cannot depend on API, then it is difficult to develop any remote Trac calling applications. It is difficult, but possible. For example, the Mylyn plug-in for Eclipse solves this problem by parsing HTML code of Trac UI and calling its HTTP POST/GET methods.

Extensibility

Due to its popularity in the open source community, a wide range of plug-ins and extensions exists for integrating Trac with 3rd party applications, extending functionality, or just simplifying common tasks. There are two big resources for such extensions:

- http://trac-hacks.org/ - extensions made by the Trac user community
- http://trac.edgewall.org/wiki/PluginList - officially supported extensions

Conclusion

Trac does not try to be a complex development and project management tool. It concentrates on one thing: issue tracking, and does the job very well. In basic installation, it does not provide much other functionality; however, there are plenty of extensions, which make Trac a more complex tool. All things considered, Trac still remains a tool designed for use in single project environments.

[TSO-AND] [TSO-EDG] [TSO-KOL] [TSO-SMA] [TSO-STA]

3.5 JIRA

Homepage:	http://www.atlassian.com/software/jira/
License:	Proprietary (Free – 4800 USD)
Implemented in:	Java
Demo site:	http://jira.atlassian.com/browse/DEMO

JIRA is a bug/issue tracking and project management system developed by Atlassian Software Systems. It is well-known for its complexity, but at the same time, keeping a clear and easy-to-use user interface. It is also famous for its community, which makes plenty of extensions and plug-ins, and also, for a wide variety of learning resources and documentation. The last big advantage is its easy integration with enterprise wiki Confluence and other Atlassian tools.

JIRA's code is open source. JIRA is good example of having proprietary software with open source code checked and extended by community. Atlassian provides JIRA for free to open source projects and organizations that are non-profit, non-government, non-academic, non-commercial, non-political, and secular. For other organizations, Atlassian charges between $1200 and $4800 US, depending on the version, with a 50% discount for an academic license. Starting with JIRA version 3.13, a free personal license is also available for non-commercial use. This license does not include Atlassian support and it is limited to three full users.

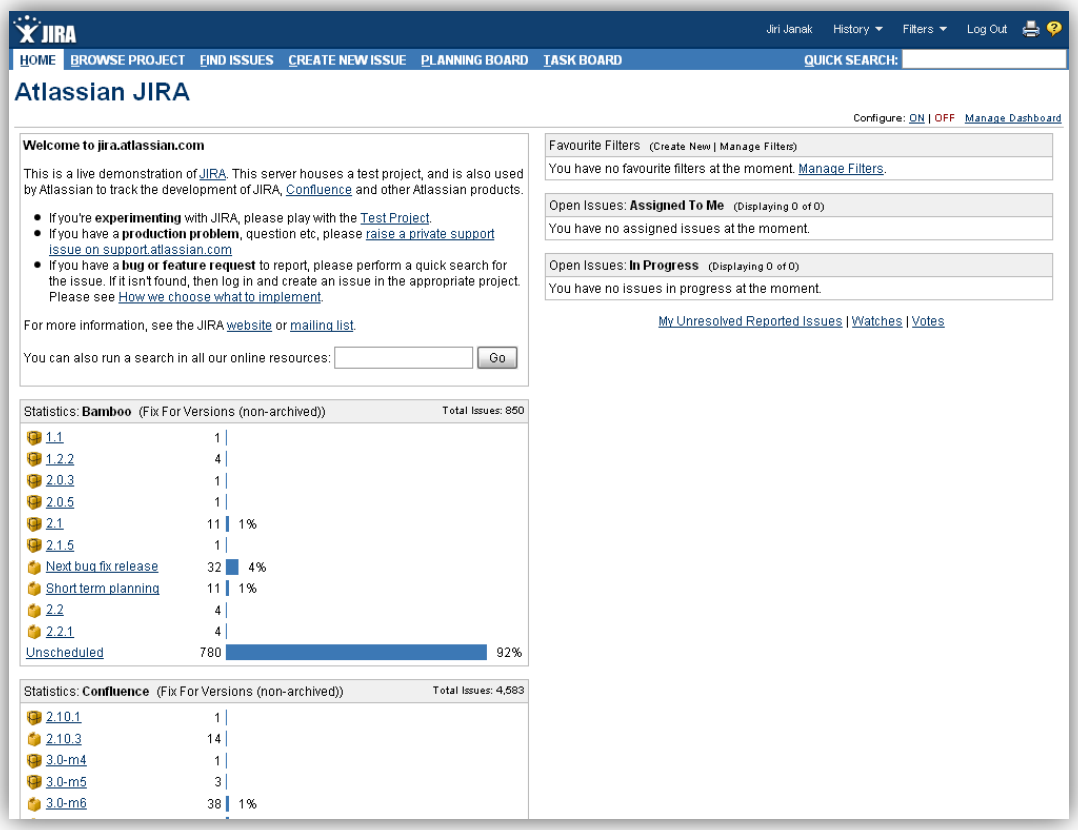
[TSO-ATL1]

Installation

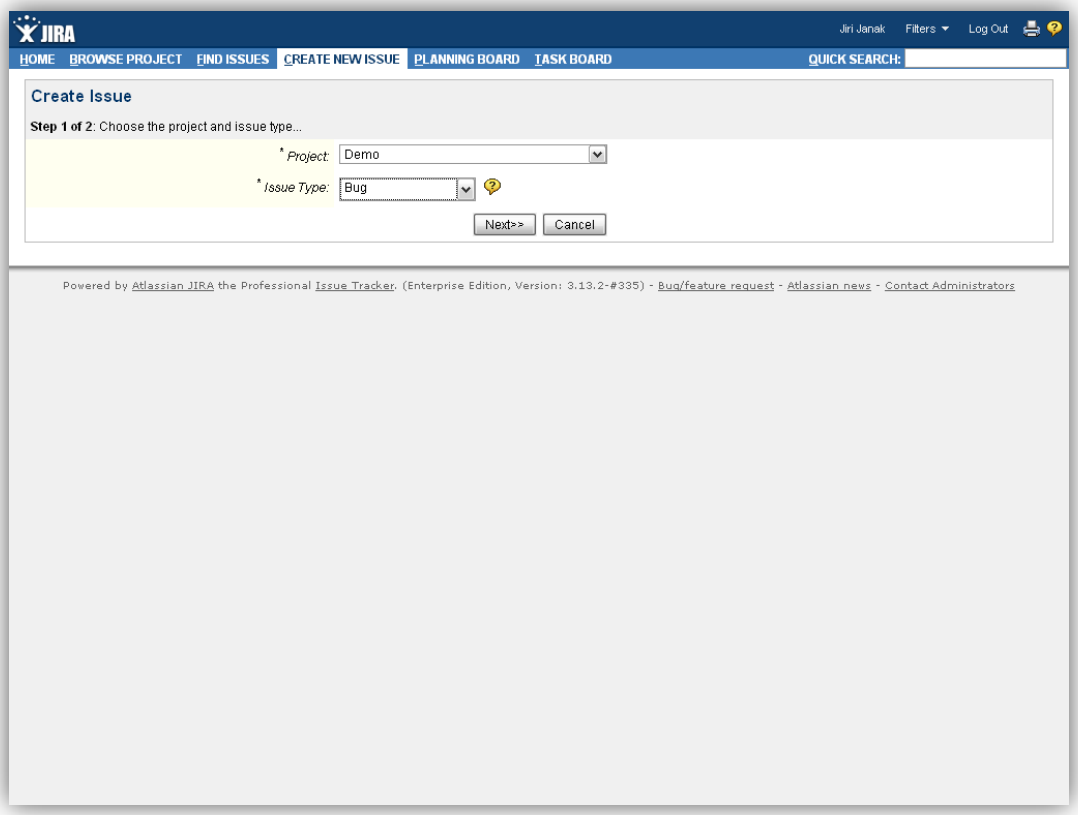
JIRA is delivered as a binary installation file, which supports either GUI-based installation (more friendly for the desktop) or text-based installation (more friendly for remote servers). No special preparations are necessary; the installation file includes all necessary parts.
There is also a possibility to not install JIRA to one's own environment, but have it hosted by Atlassian.

User interface

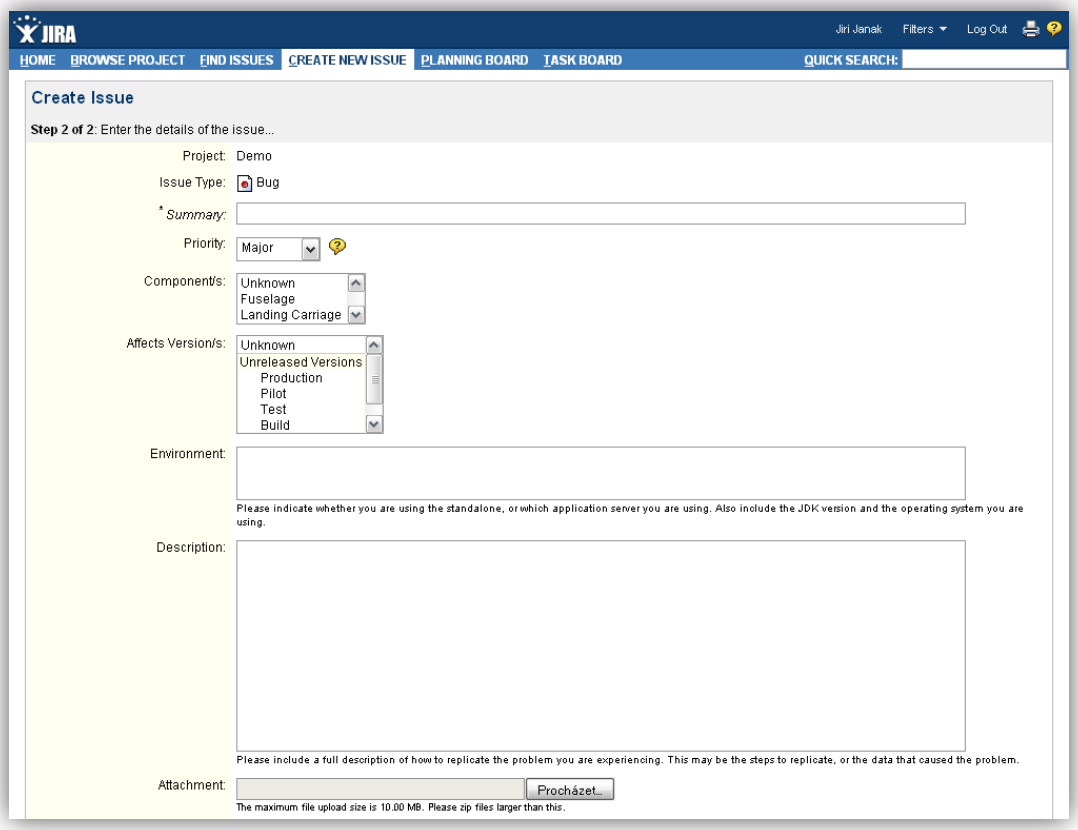
JIRA's user interface is one of its biggest benefits. When JIRAs complexity is observed, it stays very clean and easily understandable due to the UI, but it can be easily customized. Every user can define a unique UI configuration as is best for the individual.
Some tasks, by default installation, require too many steps. For this dilemma, there are plenty of plug-ins which enable a user to perform those task more easily. For example, by default, if a user wants to move ticket from one project release to another, the action must be done inside the ticket. This is not very comfortable if the user must handle 50 tickets. The GreenHopper plug-in (which we will mention later) exists exactly for this reason. Not JIRA itself, but these extensions and customizations make JIRA a top notch product.



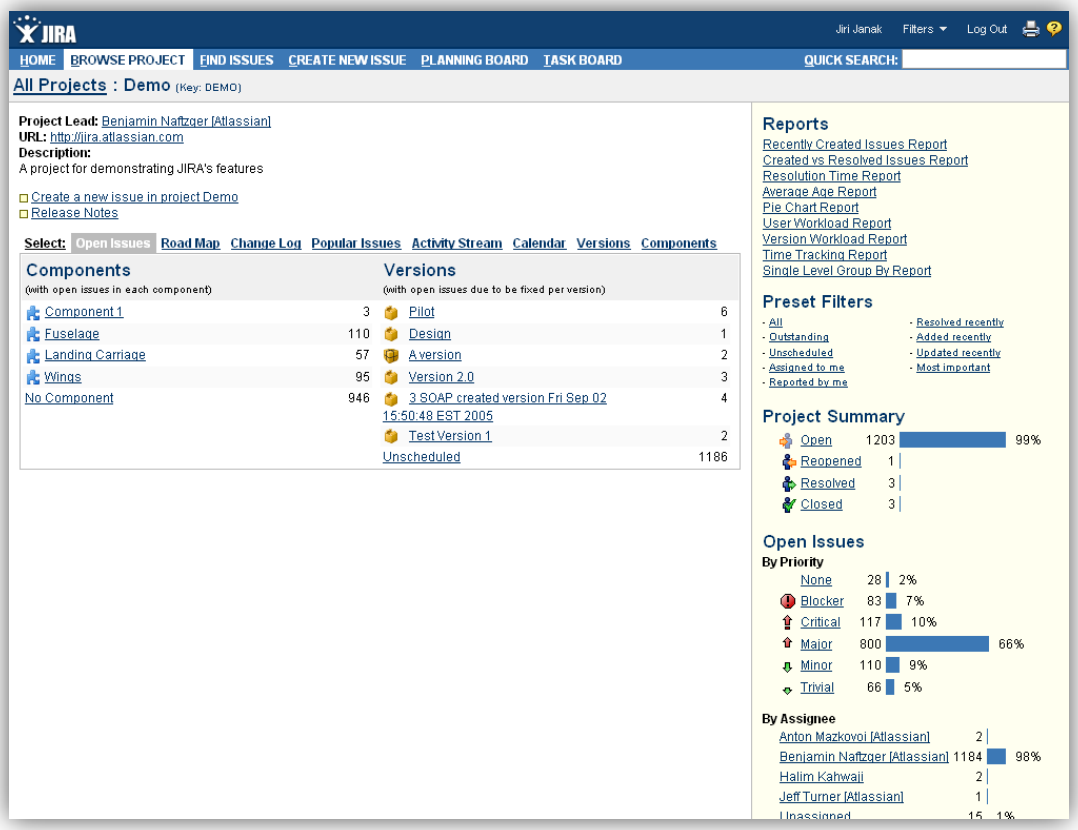
Screen shot 25: JIRA use-case – Dashboard



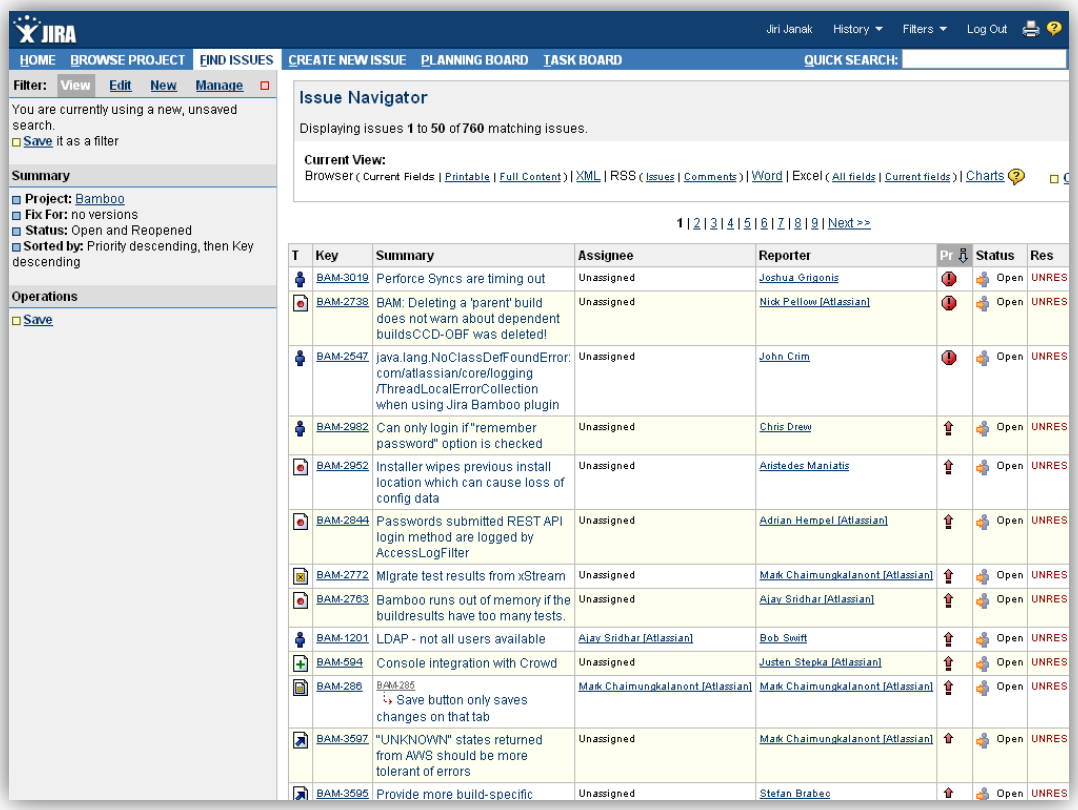
Screen shot 26: JIRA use-case – Create new issue (selecting project and issue type)



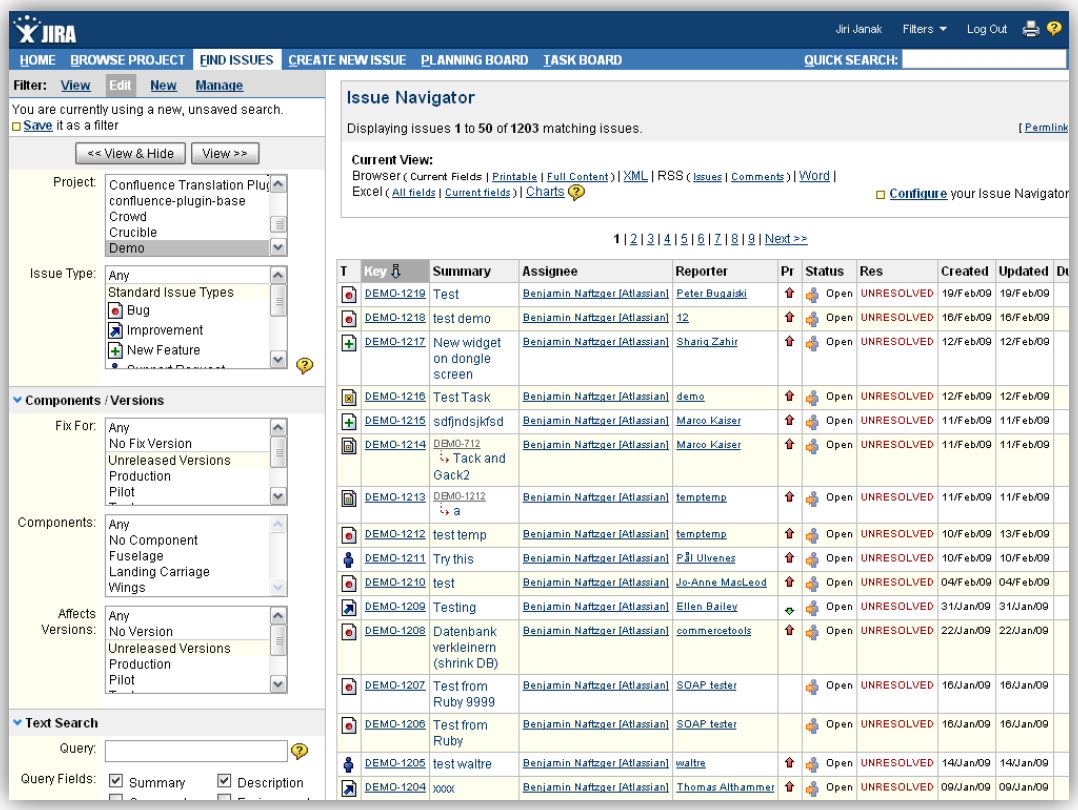
Screen shot 27: JIRA use-case – Create new issue (adding details)



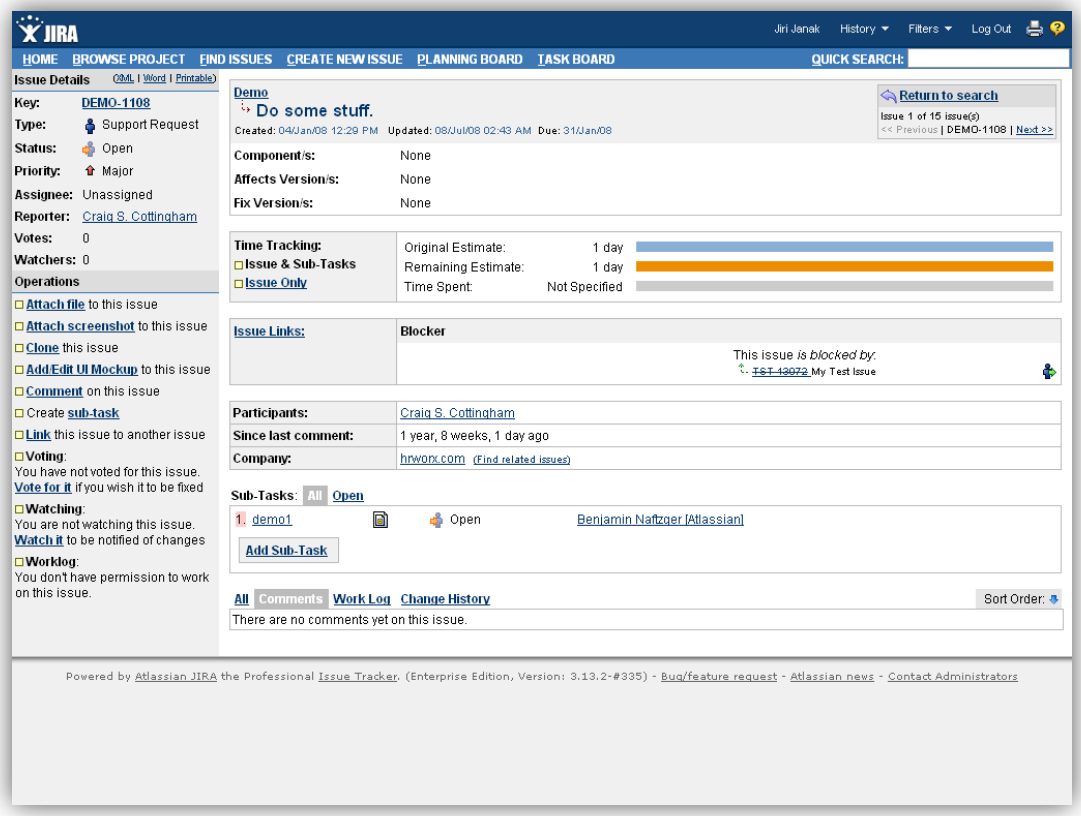
Screen shot 28: JIRA use-case – Projects Dashboard



Screen shot 29: JIRA use-case – Issue navigator (list of issues)



Screen shot 30: JIRA use-case – Creating a new search filter



Screen shot 31: JIRA use-case – View ticket (enabled actions on the left)

Ticket system

JIRA provides a default workflow – a generic set of steps that start up immediately (see diagram below). It is easy to configure JIRA's workflow within the UI. JIRA's flexible workflow architecture lets a user define individual workflows for departments, projects, and even issue types. Each workflow can have as many or few steps as required.

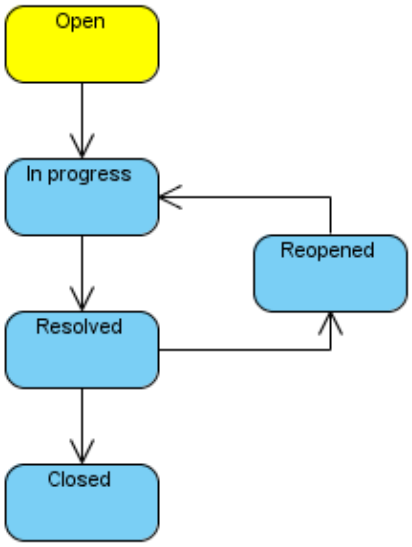


Figure 10: Default JIRA ticket workflow

JIRA allows the user to configure ticket custom fields on a system-wide, project-specific and/or issue type basis. JIRA ships with over 20 types of custom fields and a user can also develop individual types of custom fields within the UI.

JIRA also allows you to create and manage one's own issue ticket types. A user can really customize JIRA for all kinds of environments, not only for software development, but also for example, for "life management".

[TSO-ATL1]

Projects management

JIRA is designed as multi-project request tracking and project management system. Projects can be sorted to the categories, etc. as shown on image below. Each project can have individual settings (workflow, filters, metrics, reports, issue types, etc.)

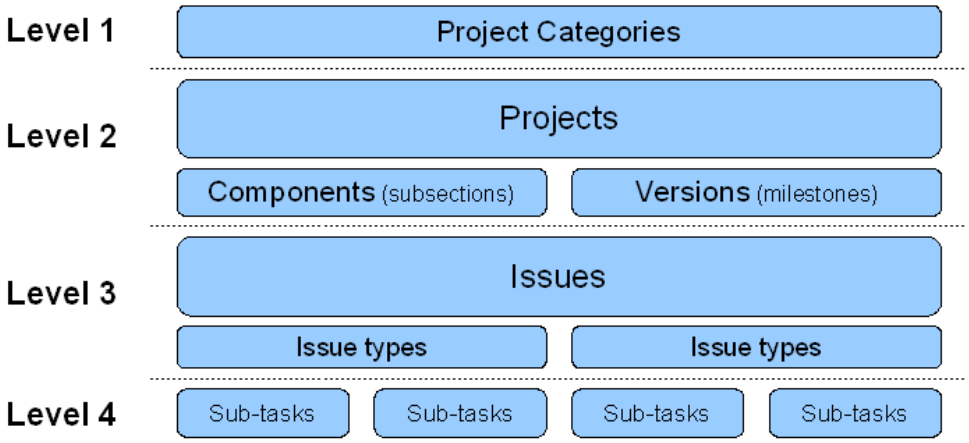


Figure 11: JIRA concepts

[TSO-ATL2] [TSO-LAW]

Analysis and monitoring

JIRA supports monitoring and reporting based on filters.

A filter is something like a customized search. A user defines what conditions have tickets to fulfill and names this. Filters can also be set throughout all projects present in JIRA, not only for each project individually.

A filter then can be used not only for ticket management, but also as a source for monitoring and reporting tools, etc. The user can, for example, by a few clicks create a filter, which displays requests "assigned to it for current release", set this to be displayed as a pie chart with sorting based on the status of the request; furthermore, this chart can be displayed as a portlet on the dashboard.

The dashboard is usually the starting point. It is highly customizable by placing so-called portlets on it. They are pluggable user interface components. Portlets are not classical portal server portlets based on the Java Portlet Specification (JSR168, JSR286)

– they are for JIRA only. There are plenty of out-of-box portlets for monitoring and reporting purposes present; however, as mentioned, users can define their own or configure the out-of-box portlets based on filters.

Source code management systems integration

JIRA supports connection to a variety of source code management systems through plug-ins. Of course, the most common ones like Subversion connector are present out-of-box.
Atlassian also provides product FishEye, which helps development teams keep tabs on what is going on in the source code repository within the web interface. This is also possible to fully integrate into JIRA.

Accessibility and extensibility

JIRA 3.0 and above ships with the RPC plug-in, which enables remote access through XML-RPC and SOAP. It is also possible to use the Java API interface and extend it to JIRA.

There is quite a large community working on extensions of plug-ins and extensions of JIRA. Their work can be found on the JIRA webpage (http://www.atlassian.com/software/jira/plugins/).

[ANC-ATL]

Specialties

- Possibility to configure the UI individually for each user, even anonymous ones;
    - this allows for JIRA to be used even for Help-desk / Support / Customer service;
- Provides fine-grained enterprise level security;
    - there can be set access rights not only for individual projects, but also for requests, even for single comments of tickets;
- Requests can be created automatically from e-mails (useful again for Help-desk, etc.);
- Requests can be divided into subtasks;
- Default support for time tracking (logging of work, claiming, time estimations); and
- Integration with other Atlassian tools, like enterprise wiki Confluence, etc.

Conclusion

JIRA is a top product in request tracking and project monitoring. It has very friendly politics of distribution and thanks to the support of the open source community; it makes plug-ins and extensions to make JIRA even better. It is easy to learn and highly customizable.

3.5.1 GreenHopper plug-in for JIRA

Homepage: http://www.greenpeppersoftware.com/en/products/greenhopper-jira/
License: Proprietary (350 – 1150 USD)
Demo site: http://www.greenpeppersoftware.com/en/products/greenhopper-jira/demo/

While JIRA models the project information very well and has excellent reporting features, it is not a very attractive environment to work in for project managers. The interface is geared towards searching and listing issues, but editing and managing the project is, by default, quite unfriendly. This is where the GreenHopper plug-in is introduced. The main goals of this plug-in are to provide JIRA users with the following:

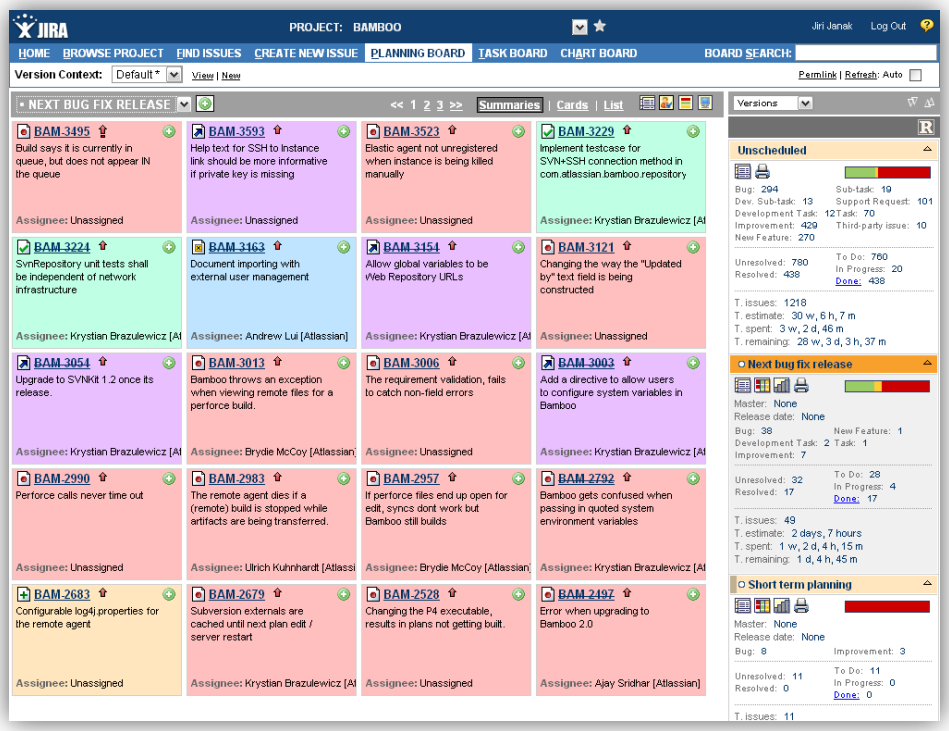
- an interactive and simple interface to manage their projects, and
- tools to increase the visibility and traceability of ongoing versions.

It was designed to make the interface more useable for both developers and project managers. Using a whiteboard and index card metaphor, it gives an instantly recognizable and very editable front end. Actually, it gives only three additional views:

- Planning Board – for a global view of all versions and all components;
- Task Board
- Chart Board – charts and diagrams to help a manager analyze the progress of a project

Here are the views in greater detail:

Planning Board



Screen shot 32: JIRA GreenHopper plug-in - Planning Board

Planning Board gives the user a global view of all versions and all components. It is divided into two parts:

- The main pane is called the "Issue navigator." The user sees a list of issues within the selected content and by using the drag-and-drop method, he or she can easily order and plan.

- The right pane contains droppable boxes and offers a choice of different management views. There are three ways to manage issues:

    1. Overview (all unreleased fix versions and unscheduled versions merged into one view) – this is very useful for ranking.
    2. Versions – by unreleased fix versions.
    3. Components – by components.

As mentioned, the right pane is droppable, so managing issues between versions and components have actually become a fun task in JIRA with GreenHopper. By simply dragging and dropping, issues can be associated to the appropriate version or component.

Each version or component box also contains the following information and allows the following operations:

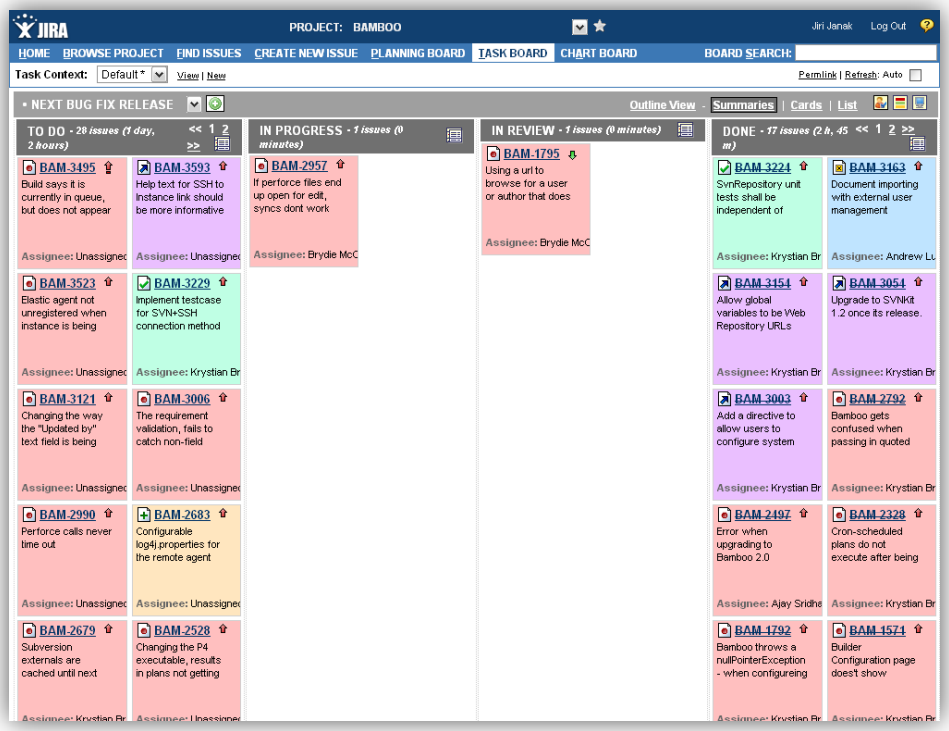
- **Progress bar,** which shows work progress on the selected version or component. Clicking on a color state will bring the issue navigator, pre-filled with the issues in that state.
- **"Master" attribute** in each version or component box. Through this setting one can organize versions/components in a " hierarchical" way. This is very useful and will help the user to have a global view of the versions and a more granular view of the sub-versions.
- **Statistics and counts** – release date, count of issues by type, count of unresolved and resolved issues, etc.
- **"Release version" button,** which will enable the user to release versions directly from the version boxes.
  If some issues are unresolved, GreenHopper offers two options:
    - Swap the unresolved issues to the selected version; and
    - Ignore these issues and release anyway.

So, in summary, Planning Board provides the user with:

- a global view of all unreleased versions;
- a global view of all components;
- a card and list look and feel;
- multi-level planning (hierarchical);
- issue ordering using drag-and-drop;
- issue planning using drag-and-drop;

- multi-selection in the planning board;
- coherent handling of sub-tasks;
- issue filtering and sorting; and
- quick access to the issue navigator.

Task Board



Screen shot 33: JIRA GreenHopper plug-in - Task Board

Task Board allows users to easily track the requests within versions. The screen is divided into multiple columns, which correspond to a JIRA transition. Users can add, remove, and rename these columns to make the board handier. Every user can personalize their own Task Board by hiding or showing specific columns.

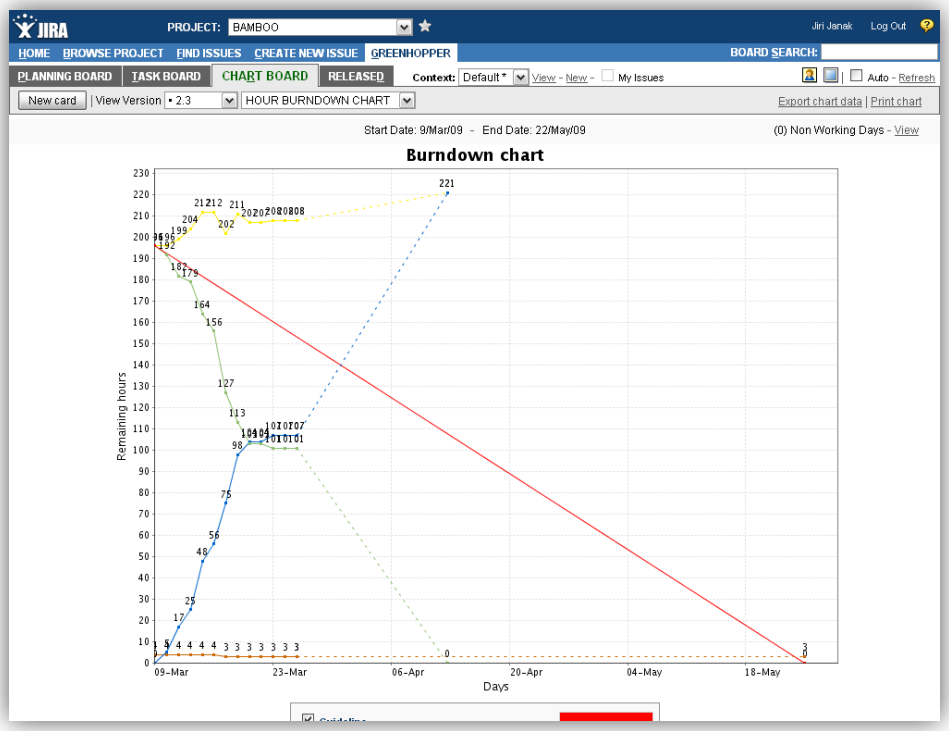
By simple drag-and-drop of issues between the swim lanes, a user can easily change the status of an issue. Dropping a card into a column will automatically display the list of all possible transitions. Of course, all notification events, post-functions, etc. will be triggered. If the transition of the issues requires some mandatory entries, GreenHopper will ask the user to enter this required data through a modal screen within the task board.

In summary, Task Board provides:

- an intuitive tracking of versions;
- an outlined task board;
- a personal board;
- configurable swim lanes;

- an update of the issue status using drag-and-drop;
- quick logging of a user's work;
- quick access to the issue navigator; and
- issue filtering and sorting.

Chart Board



Screen shot 34: JIRA GreenHopper plug-in - Chart Board

Chart Board offers plenty of charts and diagrams to help managers analyze the progress of a project. More about the available charts and their configuration can be found in the documentation available at http://www.greenpeppersoftware.com/confluence/display/GH/CHART+BOARD.

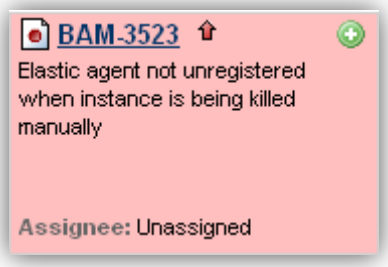
In summary, Chart Board provides:

- dynamic work and the requirement of no settings;
- a display of the burndown curve;
- a display of the team effort curve;
- a display of the estimation accuracy curve;
- a burndown chart based on a custom field;
- a burnup chart based on a custom field;
- a value chart based on a custom field;
- issue filtering; and
- a configurable start date and end date.

Issue views

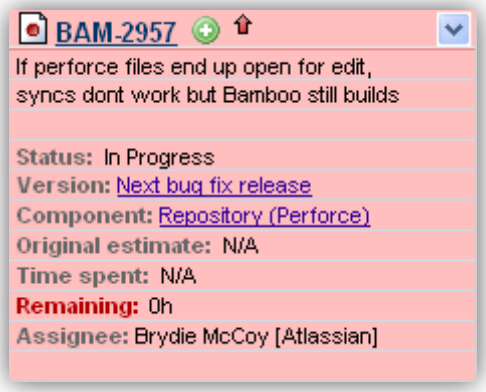
Issues can be displayed in three levels of details:

1. **Summary view**
   This view is designed to show only basic information about the ticket. By default, it shows the ticket number, short description, and assignee; however, this can be customized by user.



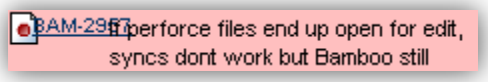
Screen shot 35: JIRA GreenHopper plug-in - Summary view of the ticket

2. **Card view**
   This view enables a detailed view of the ticket and also easy editing of its information. A user can customize which fields he or she wants to see; however, by default, this view takes up much place on the screen.



Screen shot 36: JIRA GreenHopper plug-in - Card view of the ticket

3. **List view**
   The most compact view. Shows only the ticket number and short description. Mostly used for basic overview and fast sorting of tickets.

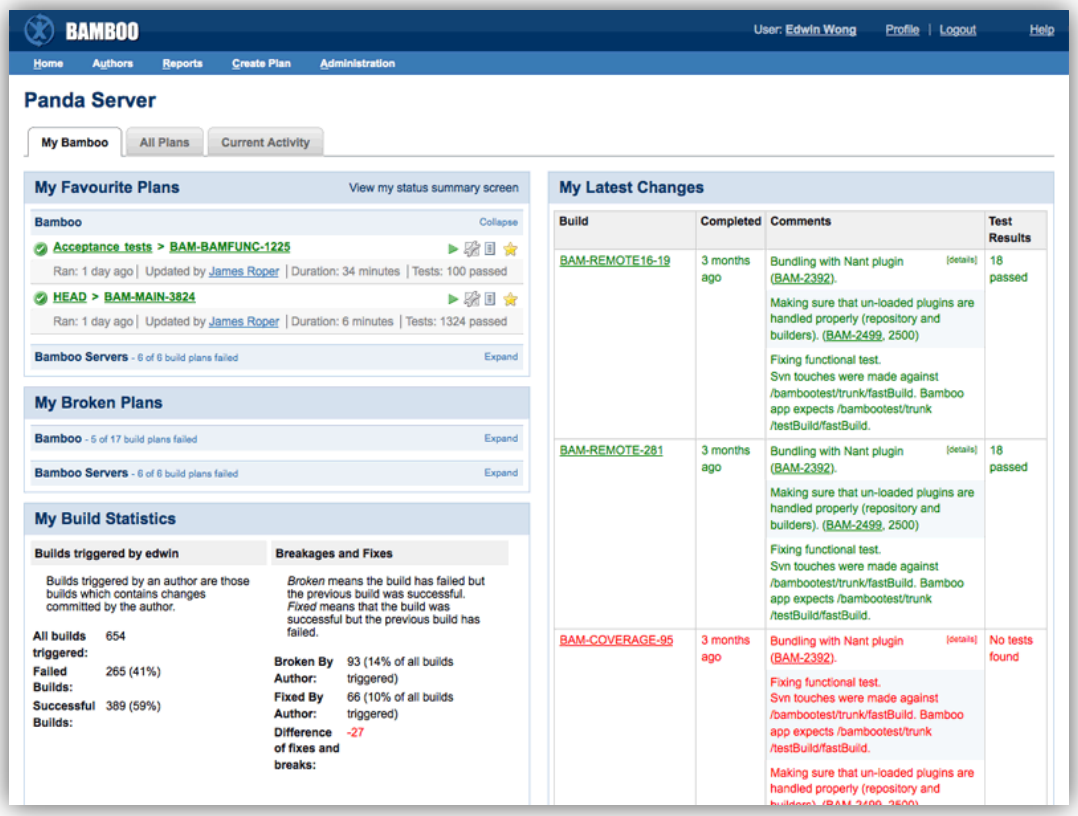


Screen shot 37: JIRA GreenHopper plug-in - List view of the ticket

[TSO-ATL2] [TSO-GRE1] [TSO-GRE2] [TSO-GRE3] [TSO-LAW]

3.5.2 Atlassian Bamboo

Atlassian Bamboo is the tool used for continuous integration. Bamboo automates the process of compiling and testing source code, saving time and instantly providing feedback about build problems for developers, allowing quick collaboration.



Screen shot 38: Atlassian Bamboo - "My activity" dashboard

Despite being a widely accepted best practice, many teams struggle to adopt a continuous integration process due to the perceived cost of setting up and maintaining such an environment. With Bamboo, setting up continuous integration process is simple. The installer auto-detects the development environment, enabling the user to start a build within minutes.

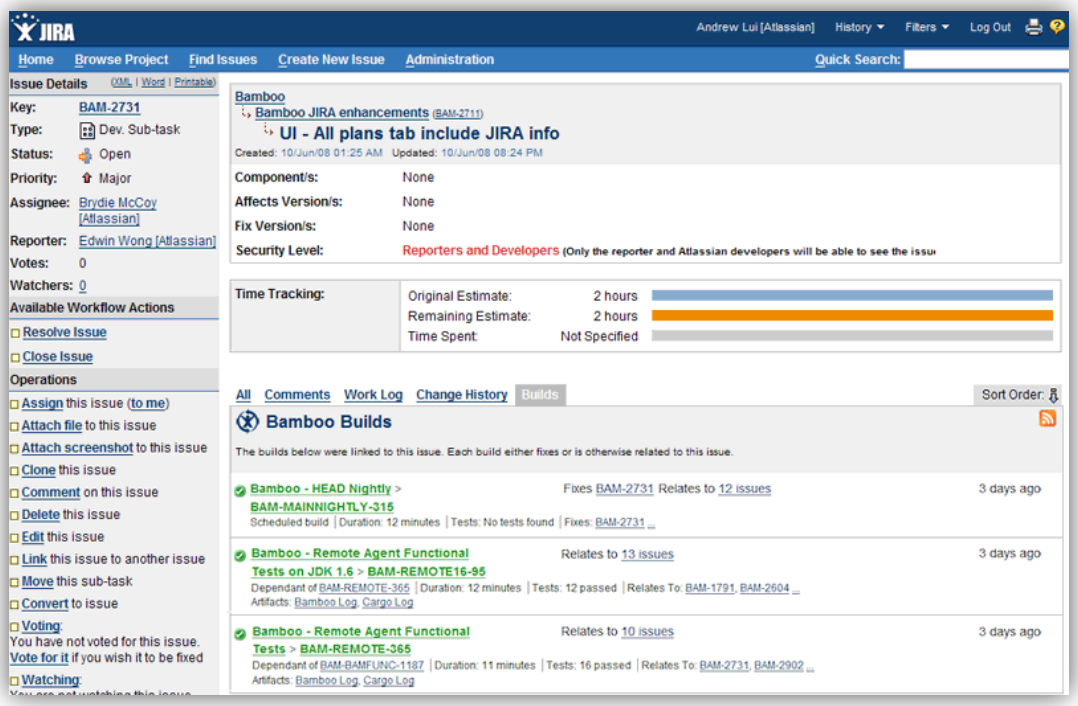
With Bamboo, one can specify multiple projects, each with their own set of unique build plans. A build plan defines elements such as:

- what tests to run;
- when builds should run;
- what outputs ('artifacts') are produced;
- which branch (if any) to check out; and
- who gets notified upon what build event.

Bamboo's notifications system and intuitive web interface provide developers with the necessary information and simple navigation to interact with each build, as well as manage the entire continuous integration process.

Bamboo also allows one to distribute builds across different machines. These distributed builds allow the user to harness the computing power of the entire network, maximizing build productivity and decreasing wait times, allowing the user to test on different platforms.
The user simply installs a new Bamboo remote agent on any additional build computer and the main Bamboo server will manage it. When a build is detected, the Bamboo server will automatically and intelligently distribute the builds to the agents. Remote agents are even able to restart themselves in the event of a crash with the Remote Agent Supervisor.

It is also possible out-of-the box to integrate Bamboo with JIRA. From within the issue details, one can manage builds and testing – as shown on the image below.



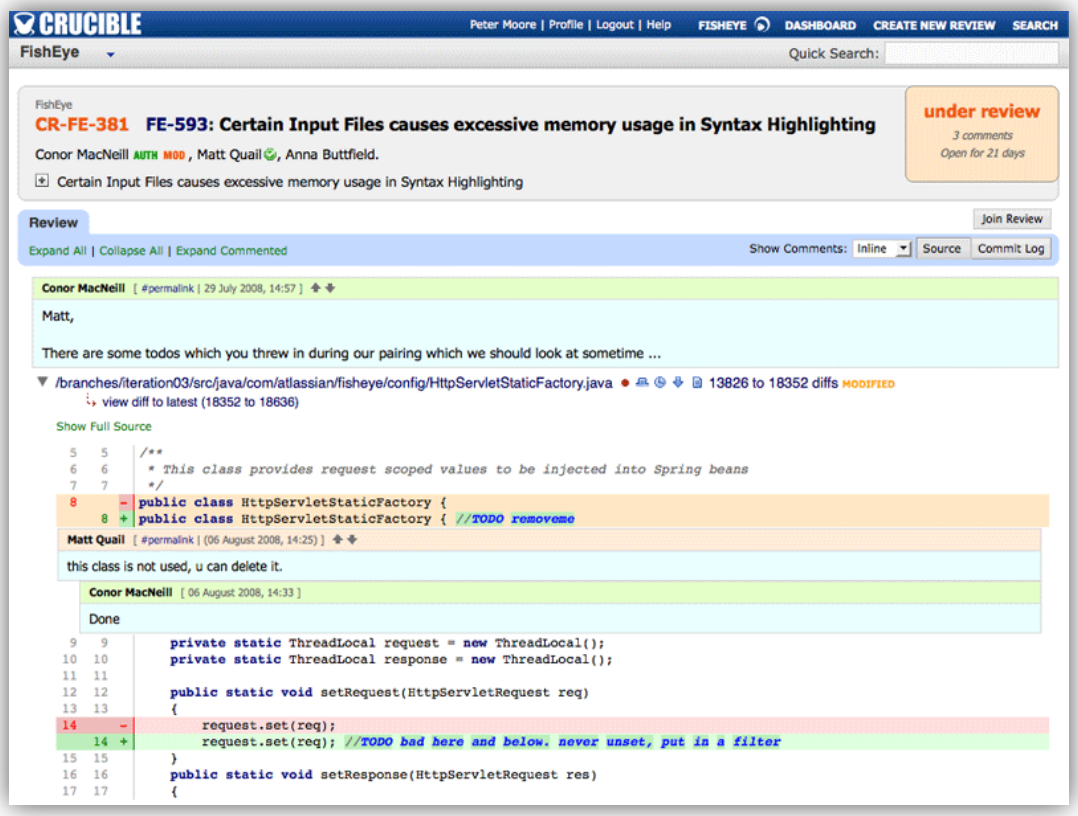
Screen shot 39: Atlassian Bamboo - Integration into JIRA

[TSO-ATL1]

3.5.3 Atlassian Crucible

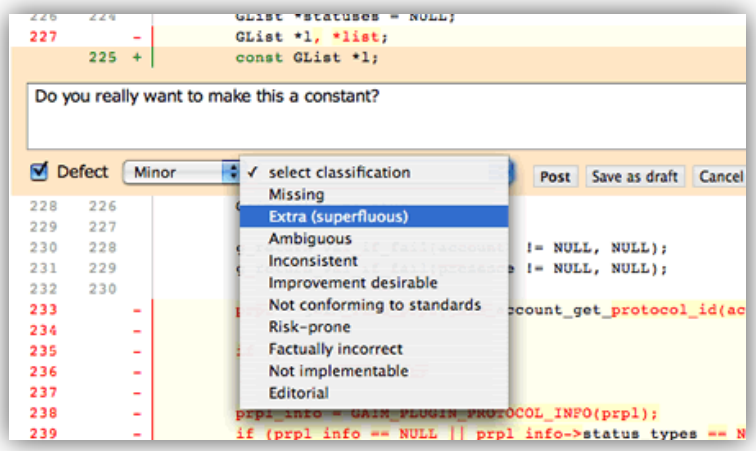
Crucible is a peer code review tool that allows teams to review, edit, comment, and record outcomes.

Effective code review improves code quality by identifying bugs earlier in the development cycle when they are less costly to fix. In the process, developers learn new development techniques from each other and raise their level of discipline, knowing their code will be reviewed by peers.



Screen shot 40: Atlassian Crucible - View on review of code

Making it easy for every developer to engage with the code review process is a key item to implementing the practice. Participation in reviews with Crucible is really simple. As soon as a potential problem is spotted in the code, the user simply selects the line(s) of code that he or she wants to comment on and starts typing.

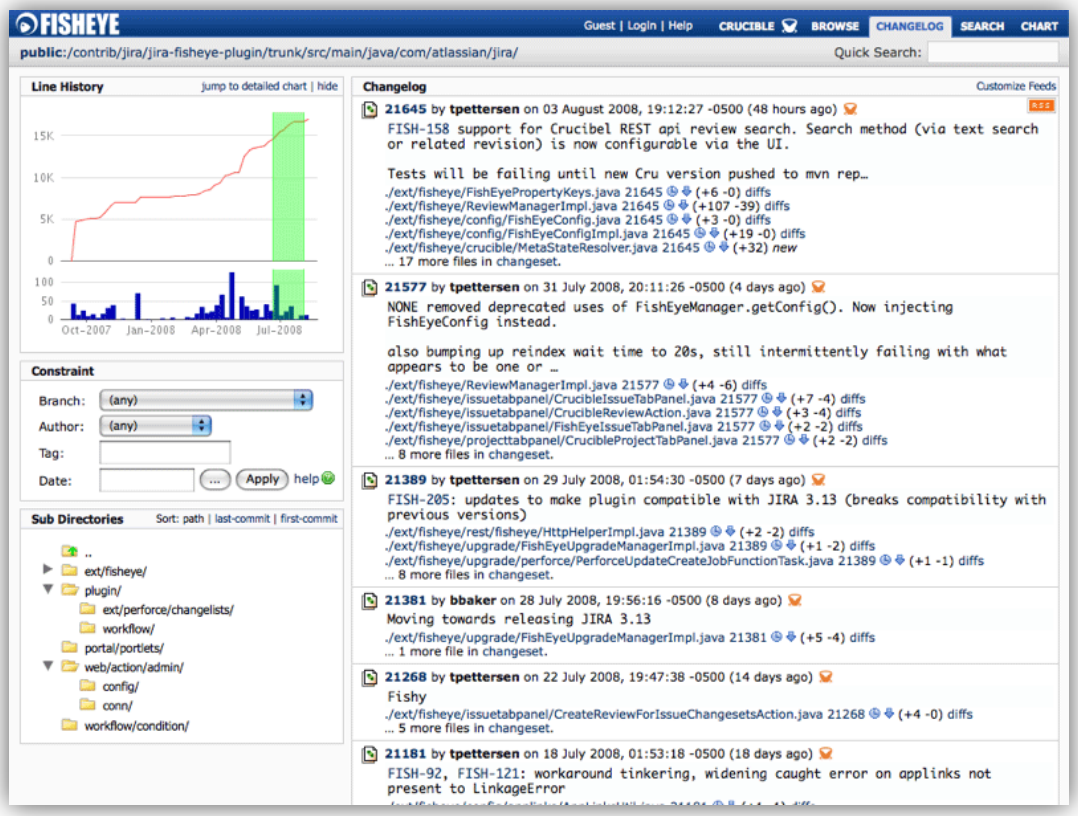


Screen shot 41: Atlassian Crucible - Add new comment (select code & type)

[TSO-ATL1]

3.5.4 Atlassian FishEye

FishEye is an advanced repository browser. It opens the source code repository and helps development teams keep tabs on what is occurring using a web interface.



Screen shot 42: Atlassian FishEye - Changelog dashboard

FishEye gives the source code repository (Subversion, CVS, or Perforce) a simple Web interface. Developers can use FishEye to browse directly to a file or directory, quickly traverse deep directory hierarchies, and see visualizations of a branch history for an individual file.

FishEye makes the repository instantly discoverable through search. Queries by author, date range, tag, branch, filename, comment, file content, etc. are possible. FishEye gives the user a chronological view of changesets (i.e. collections of files that were included in the same commit) per directory, per author, or per branch. It also allows a user to track these changesets, commits, and other activity in the repository in near real-time, all from the comfort of the favorite RSS reader, or via email alerts.

FishEye makes it simple to share quick links to source code by representing each line of code as a unique URL. FishEye can directly link to specific files, revisions, diffs, lines of code, directories, changesets, search results, and more.

It also gives the user a simple way to report on activity in the repository with meaningful charts and graphs. These charts can be used to report on the activity of teams, the status of projects, and the detail of what has changed. These reports can also be used in a user's own web pages and online documents.

3.6 Head to head comparison

	Bugzilla	CodeBeamer	ClearQuest	JIRA	Trac
Installation	☹ Necessity of preparation of environment, manual configuration of config files.	☺ Very easy by running step-by-step installer. No environment preparation needed.	☺ Very easy by running step-by-step installer. No environment preparation needed.	☺ Very easy by running step-by-step installer. No environment preparation needed.	☹ Necessity of preparation of environment. Installation by shell scripts and later edit of config files.
User interface	☹ Very plain, in the beginning not very user friendly, however very straightforward and functional.	😐 Complex UI. Not as intuitive in the beginning, but effective.	😐 Desktop application and web UI are available. Not a usability competition winner, however after some time user gets accustomed.	😐/☺ Basic UI is intuitive and simple to learn to use, but some tasks are not very comfortable, however UI is highly configurable and there a wide range of extensions exists.	☺ Very simple, intuitive and easy to learn UI.
Request/Ticket workflow	☹ Hard-coded request workflow model.	☺ Easy to customize within user interface.	☺ Easy to customize within ClearQuest Designer application	☺ Very easy to customize within user interface.	😐 Editable request workflow model (by edit of config file).
Request/Ticket custom fields	☺ Adding possible in UI.	☺ Adding possible in UI.	☺ Adding possible in ClearQuest Designer application	☺ Very easy to customize within user interface.	☹ Adding possible by edit of configuration file and manual update of database.
Multi-project environment	😐 Yes, by "Classification"	☺ Yes	☺ Yes, by having multiple issue databases.	☺ Yes	☹ Not supported. One instance of Trac per project.
Accessibility	☺ Built-in XML-RPC interface	☺ Built-in Caucho Hessiann binary web service interface	😐 Java API for ClearQuest available	☺ Built-in XML-RPC and SOAP interface; Java API available.	😐 XML-RPC interface plug-in available only for certain versions of Trac.

Additional value	+ Time tracking + System of votes which request to handle first	+ Complex development platform with build-in Wiki, DMS, Discussion board, etc. + CMMI reports	+ Tight integration with other Rational family products	+ Highly configurable & customizable + E-Mails can be source of issue tickets + Integration with other Atlassian family products + Variety of plug-ins which increase JIRA functionality	+ Built-in Wiki + Variety of plug-ins which increase Trac functionality

Table 2: Head-to-head comparison of most common issue tracking systems

4 Tracking systems IDE integration

As the reader could see in the previous chapter, most issue tracking systems have web-based user interfaces; however, for the developer, it is more comfortable to have everything within the development environment (IDE). That is why there were plug-ins and modules for those IDEs developed. Actually, there are two approaches on how to handle this problem:

1. **Universal plug-in with bridge for connecting the issue tracking system**
   This is the approach of the below-mentioned projects Mylyn and Cube°n. They provide common UIs within IDE for working with request tickets; however, they generally disable the use of specific features connected to issue tracking systems.

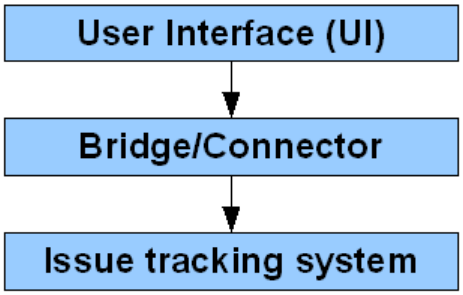


Figure 12: Universal plug-in scheme

As one can see, the most important part is the bridge, or connector, as it is handles all the communication with the issue tracking system. There are three types of bridges known so far:

1) *Web service-based bridge/connector*
   As most of the issue tracking systems provide a web service interface, this is most common approach. The connector "translates" UI commands to calling of web services and returns results for the UI.

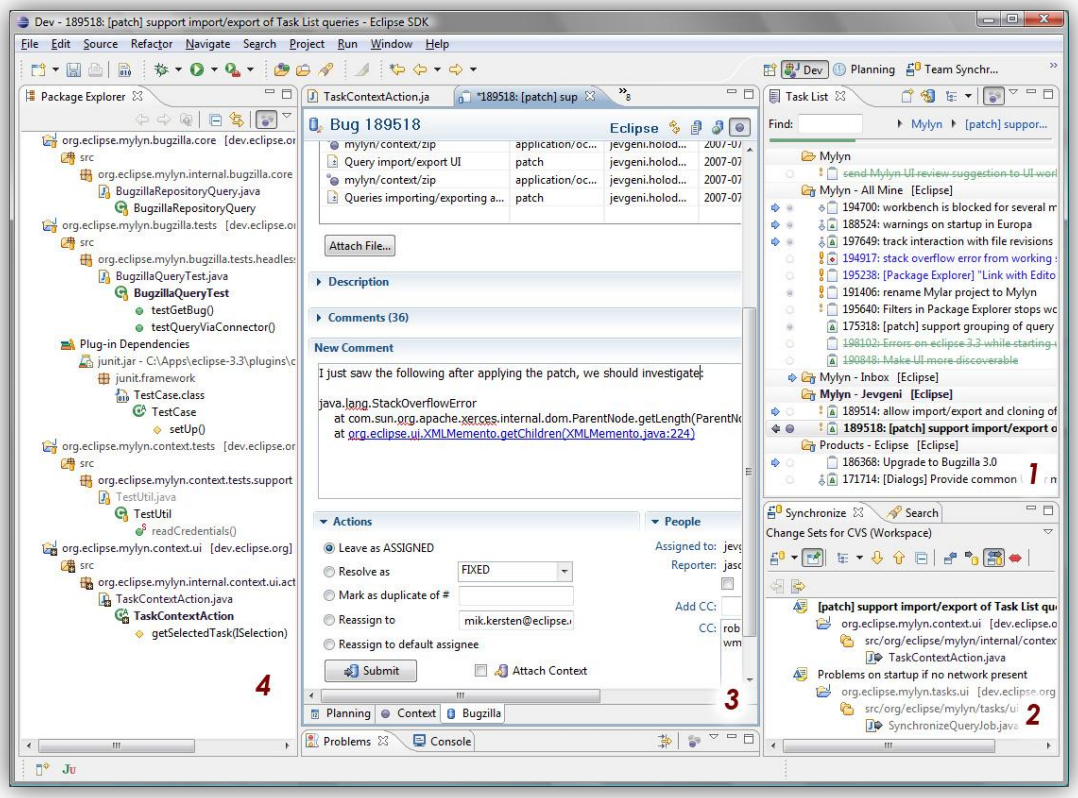
2) *API-based bridge/connector*
   There has to be a local application installed with a public application programming interface (API), which is called by the connector. The application handles communication with the issue tracking system itself.

3) *Bridge/connector based on parsing of issue tracking system UI*
   Not so common, but sometimes a used approach. The connector simulates the work of a user in the browser by calling HTTP requests and parsing HTML responses. This approach is used, for example, for working with Trac, as it cannot be guaranteed it will always have a working web service interface.

2. **Single issue tracking system plug-in**
   The plug-in is designed for usage with single particular issue tracking systems and tries to enable most of its features within IDE. Below, the CodeBeamer plug-in and Atlassian IDE Connector will be mentioned.

4.1 Mylyn

Homepage:	http://www.eclipse.org/mylyn/
Supported IDE:	Eclipse
Supported TS:	JIRA, Trac, ClearQuest, CodeBeamer, Bugzilla, …
License:	Open source
Development status:	Production/Stable



Screen shot 43: Mylyn - Overview

Mylyn is a task-focused UI for Eclipse. As it was one of first developed IDE integration plug-ins, it is also probably more advanced with the most features and most issue tracking system connectors.

It supports task management and monitors work activity to identify information relevant to the task at hand. Mylyn uses this task context to focus the Eclipse UI on the interesting information, hide the uninteresting, and automatically find what is related. This puts the needed information to get work done at the user`s fingertips and improves productivity by reducing searching, scrolling, and navigation. By making task

context explicit, Mylyn also facilitates multitasking, planning, reusing past efforts, and sharing expertise.

[TSI-KER] [TSI-ATL4]

Installation

Since Eclipse 3.2, Mylyn has become a part of the Eclipse installation; it only needs to be enabled within "Software Updates;" however, the user must remember to install repository connectors, which are not part of basic installation. This is also possible in "Software Updates."

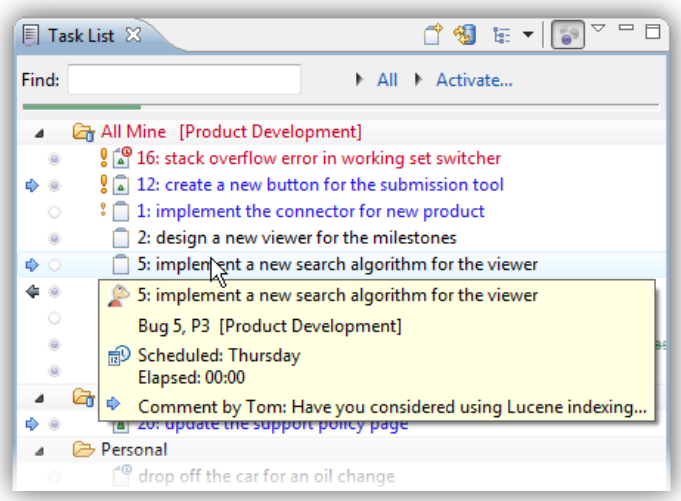
Mylyn can be installed as a standalone task list for managing personal tasks, with the focused UI support for the Eclipse SDK to reduce information overload, and with one or more task repository connectors (including one for JIRA).

[TSI-ECL1]

User Interface

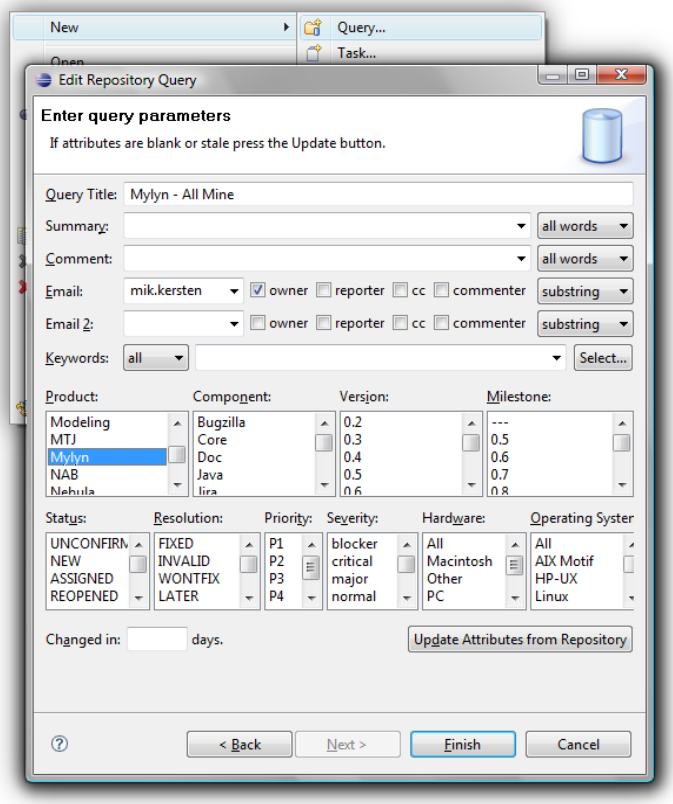
As shown on the image above (screen shot 46), the UI can be divided into four main components:

1. **Task List**
   Task List enables the user to view and manage tasks. It contains both "Local tasks" and shared "Repository tasks" that are stored in a task repository such as Bugzilla or JIRA. Local tasks are typically contained in categories, repository tasks are contained in special categories that represent queries.



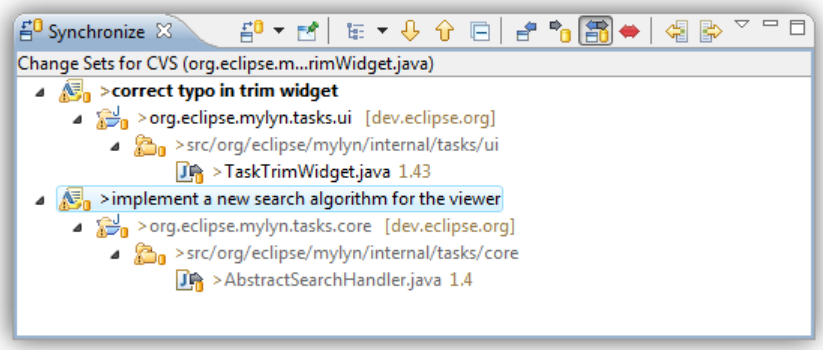
Screen shot 44: Mylyn - Task List

It is also from here where one can create new tasks (local or repository), and create new queries.



Screen shot 45: Mylyn - Create a new query

2. **A change set managed with task context**
   The task-focused interface provides several ways to improve work flow when working with a source code repository such as CVS or Subversion. The user can commit or update only the resources that are in the context of a particular task. This helps a user to work on several tasks concurrently and avoid polluting workspace with changes that are not relevant to the current task.



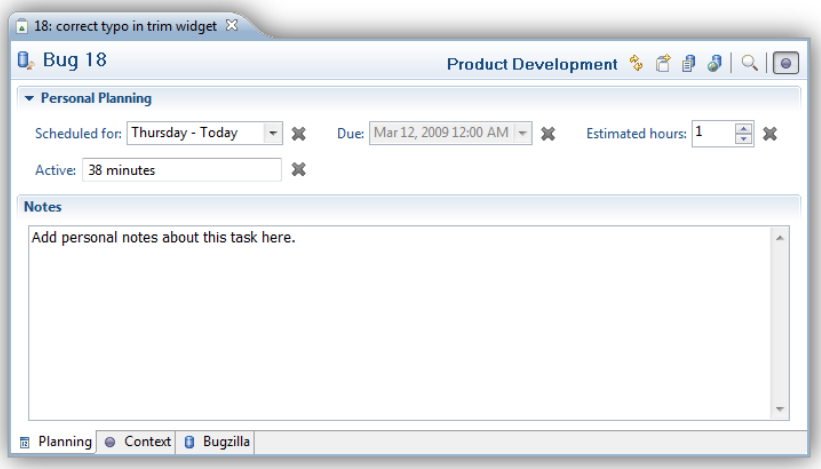
Screen shot 46: Mylyn - Change sets

3. **A rich task editor with offline support**
   The task editor allows the user to view and edit the tasks in the task list. One must double-click on a task in the task list to open the editor. The features of the task editor will vary depending on whether it is a local task or a shared repository task. For shared repository tasks, there are some differences,

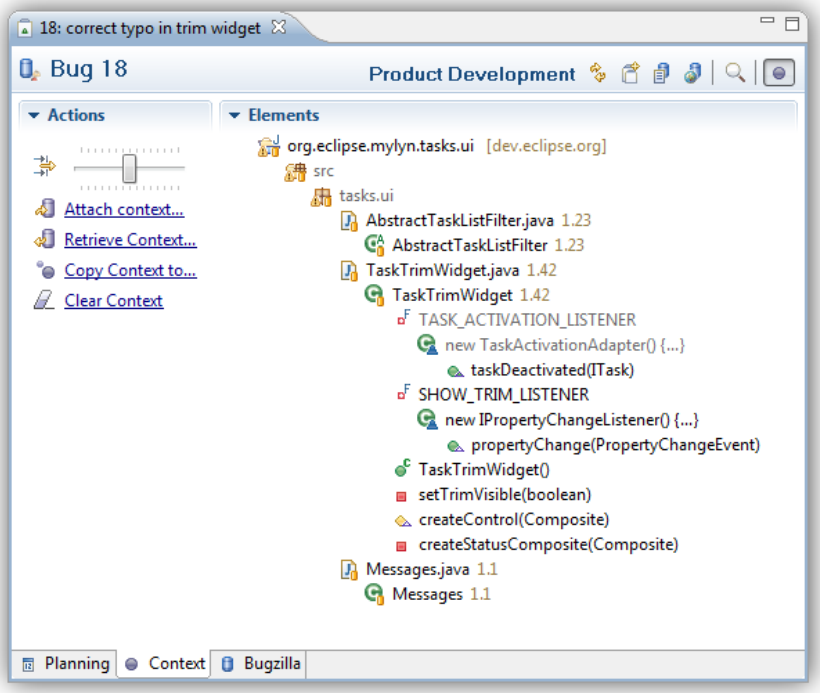
depending on the type of repository (and corresponding connector) that is being used.

Task editor consists of three tabs:

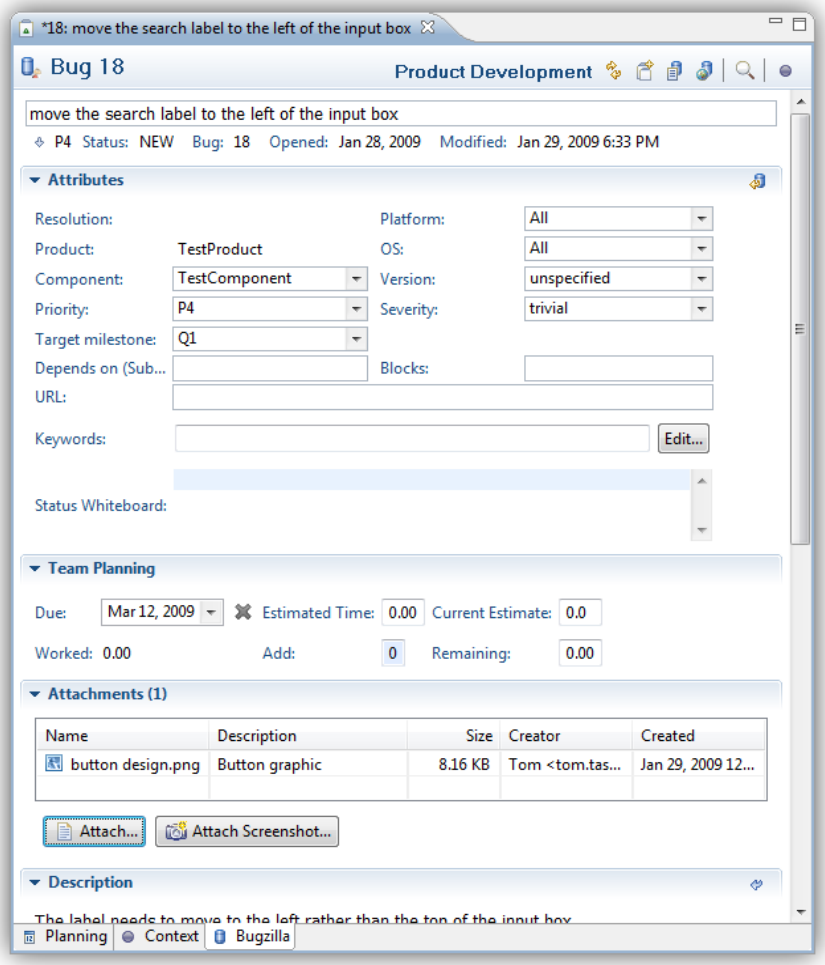
- Planning tab –shows local information about the task that is private to one`s workspace and allows the user to make private scheduling.
- Context tab – allows the user to manage the context of resources associated with the task.
- Task detail tab –shows information about the request ticket and allows the user to edit and manipulate it.



Screen shot 47: Mylyn - Task editor - Planning tab



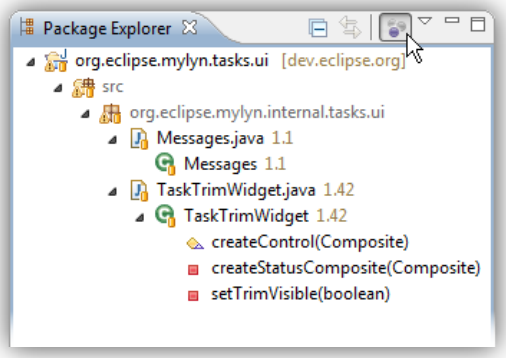
Screen shot 48: Mylyn - Task editor - Context tab



Screen shot 49: Mylyn - Task editor - Task detail tab

4. **Task-focused mode on Eclipse's Package Explorer**

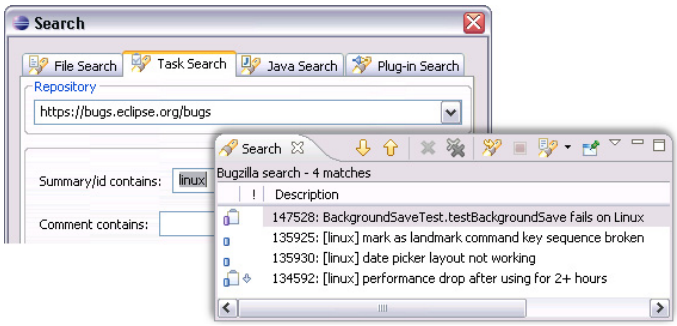
   Navigator views can be focused (e.g. Package Explorer, Project Explorer, Navigator) by toggling the "Focus on Active Task" button in the toolbar. When focused, the view will show only the resources that are "interesting" for the currently active task.



Screen shot 50: Mylyn - Focused Package Explorer

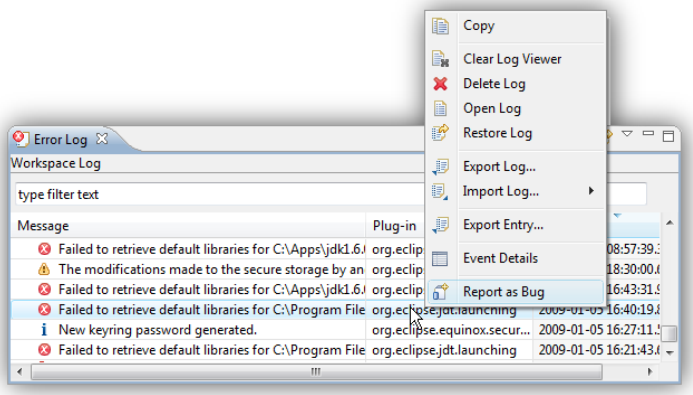
Other features

- **Task Search**
  A user can use Ctrl+H or the search menu (Search > Task Search), select the repository of interest, enter search criteria, and click Search. The search view opens, allowing one to open tasks that match the search, as shown in Figure 54. To refine a search, the user simply opens the search dialog again to restore the previous search criteria. Search results can be turned into a query from the pop-up menu in the Search view.



Screen shot 51: Mylyn - Task Search

- **Reporting bugs from the Error Log**
  Bugs can be created directly from events in the Error Log view. This will create a new repository task editor with the summary and description populated with the error and event's details.



Screen shot 52: Mylyn - Reporting bugs from Error Log

Repository connectors

As already mentioned, Mylyn recognizes two types of tasks: Local tasks and repository tasks. Local tasks are stored locally on the workstation and are not shared. Repository tasks are synchronized with an issue tracking systems repository connected by a repository connector.

The concepts are identical for other supported task repositories, but the level of integration and maturity of connectors varies. A fully integrated connector provides the following features:

- **Queries**
  A query is the mechanism for retrieving sets of tasks into Mylyn's Task List. Query editing and retrieval facilities are connector specific and provide either an Eclipse-based query designer, retrieval of queries from an account on a server, or both.

- **Rich editing**
  Tasks and queries can be edited with an integrated editor. This facility provides hyperlinking to tasks and other structured elements, as well as Eclipse and desktop integration, such as drag-and-drop. Connectors that provide rich editing can still provide facilities for opening tasks and queries with Eclipse's embedded browser.

- **Attachments**
  Files can be attached to and retrieved from the repository. This feature lets the user attach files from within the operating system or Eclipse workspace and it enables facilities such as context sharing.

- **Offline support**
  A user can work disconnected and access tasks and queries immediately without needing to wait on a server. This facility also provides change notifications, which let a user use the Task List as an inbox for tasks instead of relying on one`s e-mail client.

To let the user get the full integrated experience from Mylyn and stop relying on external or Web-based UIs, the connector must support all four of the facilities listed here.

[TSI-ECL2]

Conclusion

Mylyn was the first of its own and most of the other products copy its concepts and features. There are three main areas which Mylyn captures:

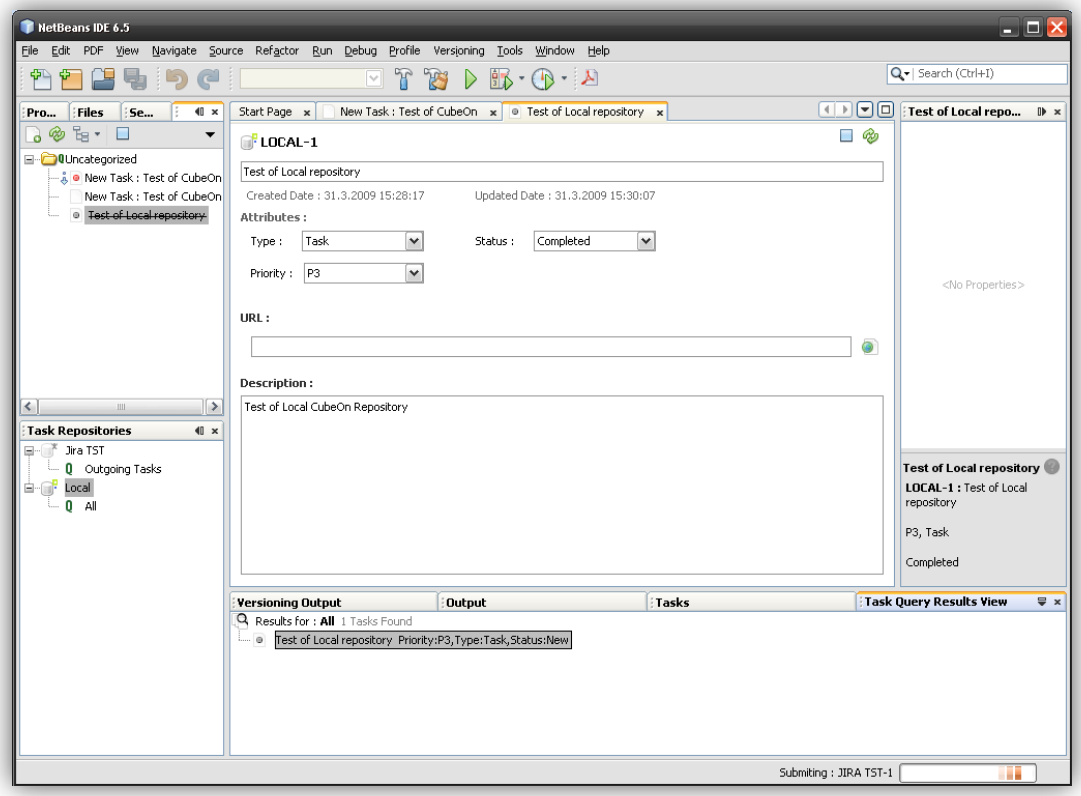
- Work with the bug tracking system, scheduling and working with tasks, storing the context for tasks, etc.
- Focus to display only relevant information in the IDE on the currently processed request (based on the context of the request).
- Create change sets to commit to the subversion (CVS) on the context basis.

As it was the first system, it also has the largest database of repository connectors, and its core is also used inside other plug-ins, as will be mentioned later.

[TSI-ECL1] [TSI-ECL3] [TSI-KER] [TSI-MAR]

4.2 Cube°n

Homepage:	http://code.google.com/p/cubeon/
Supported IDE:	NetBeans
Supported TS:	JIRA, Trac
License:	Open source
Development status:	Production/Stable



Screen shot 53: Cube°n – Overview

Cube°n is a task-focused UI for NetBeans IDE. It is a quite new project started as part of Google's "Summer of Code 2008". Same like Mylyn, it connects to various issue tracking systems using connectors; however, as it does not have so long a history and a smaller group of developers; it only has Trac and JIRA connectors working so far.

[TSI-GUN2]

Installation

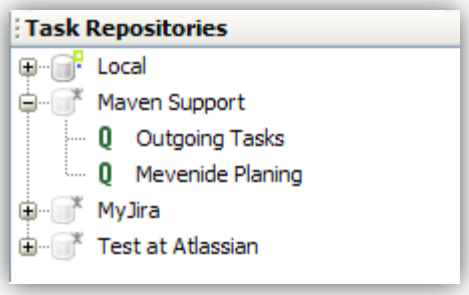
Cube°n can be installed easily just by checking the checkboxes within the "Tools – Plugins" window.

User Interface

Again, as shown in the image above (screen shot 56), the UI can be divided into couple components:

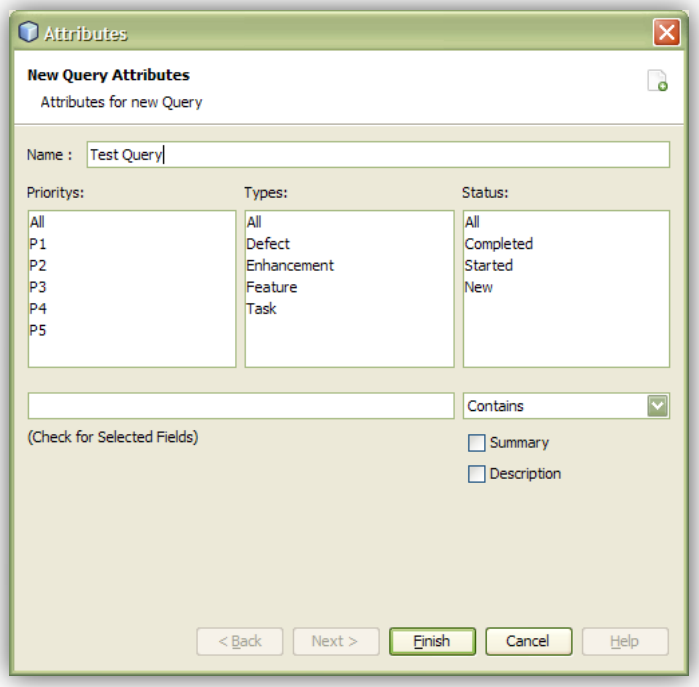
1. **Task Repositories**

   Task repositories are used for management of connections to repositories and also for creating and executing Task Queries.



Screen shot 54: Cube°n – Task Repositories & Queries

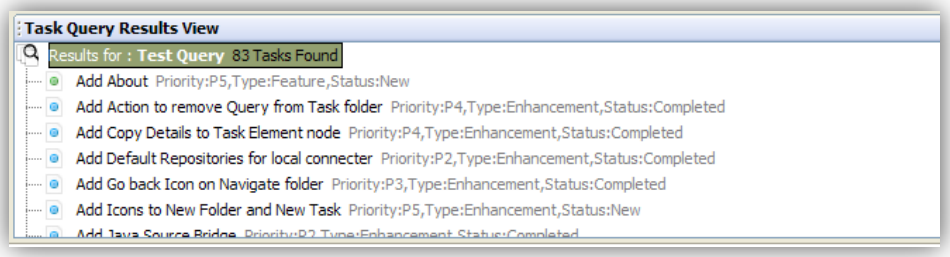
Task Queries can be either remote (e.g. in JIRA, one can use filters that have been defined) or can be local. It is also possible to combine both – a local query can be applied upon the results of a remote query.



Screen shot 55: Cube°n - Create a new query

2. **Task Query Result View**

   After selecting a query in the Task Repository window, the list of tasks is displayed.



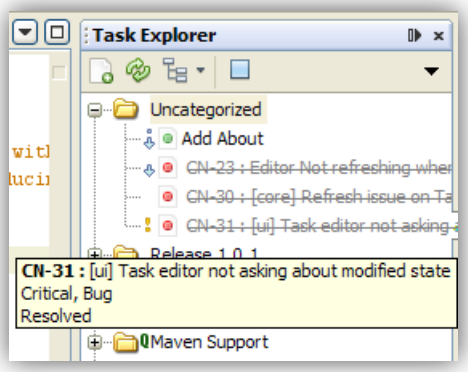
Screen shot 56: Cube°n - Task Query Result View

From here, the user can open and start editing the task.

3. Task Explorer

Task Explorer is another view for currently opened tasks. It provides the following features:

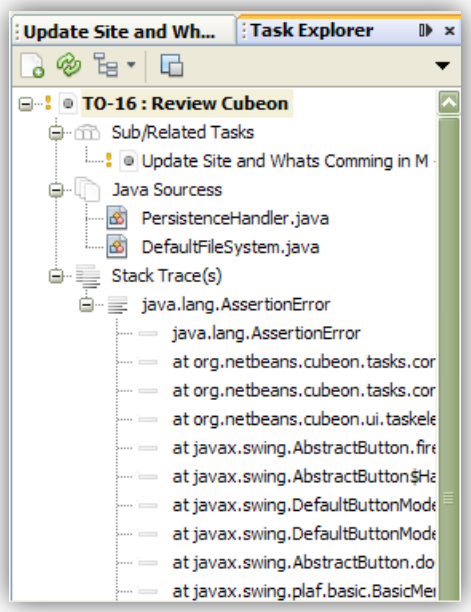
- tasks can be categorized into "Folders;"
- tasks can be filtered and sorted (by priority, status, etc.);
- synchronization with repository queries; and
- task icon budging (for example, priority).



Screen shot 57: Cube°n - Task Explorer

If Task is activated from Task Editor, it will automatically switch to Task Context View in Task Explorer. This is the view which would allow the user to:

- add any source to a task (if it is on the open projects class path);
- add and manage subtasks and related tasks; and
- add to task stack traces (automatically).



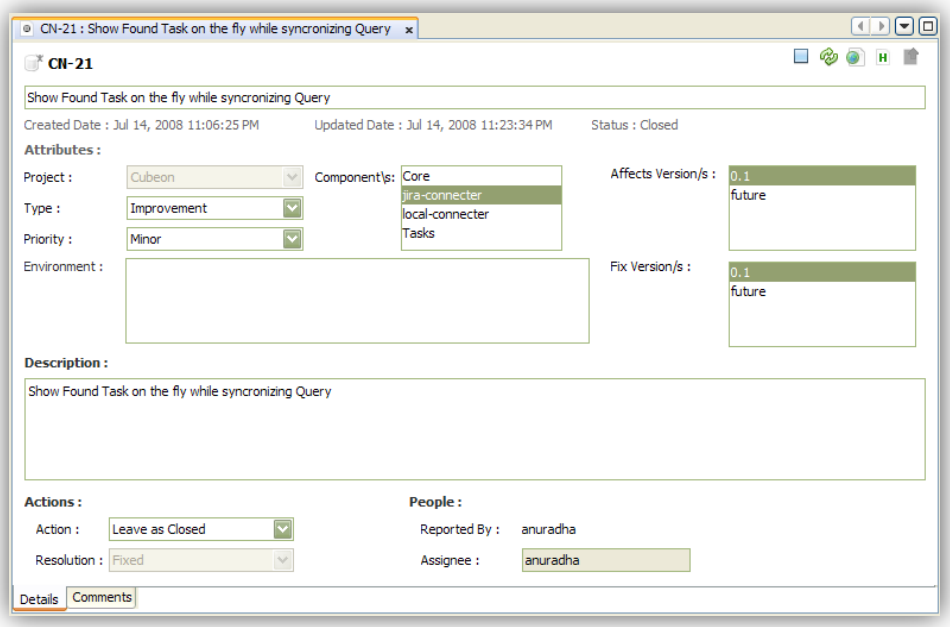
Screen shot 58: Cube°n - Task Explorer in Task Context View mode

It seems to be a clumsy way of enabling IDE to display only relevant information on the currently processed task, if compared with Mylyn.

4. Task Editor

Task editor consists of two tabs:

- Details tab –shows information about the request ticket and allows the user to edit and manipulate it.
- Comment tab –comments to the ticket are displayed and can be added.



Screen shot 59: Cube°n - Task Editor

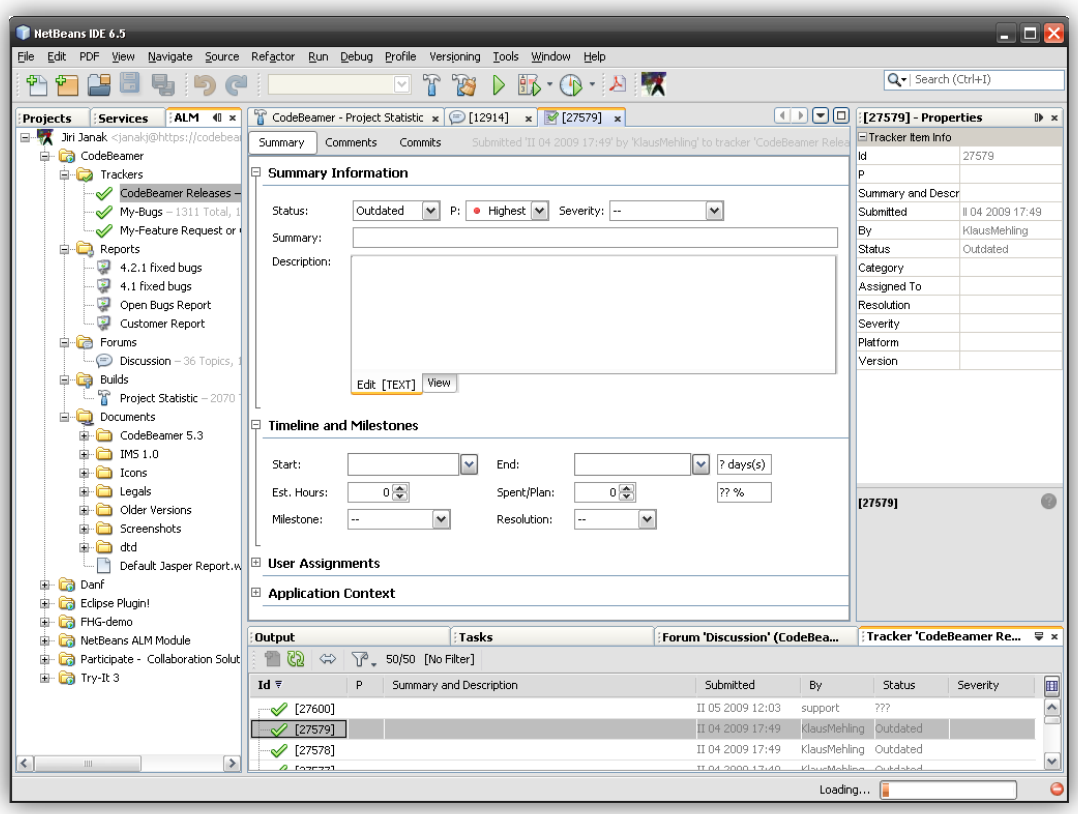
Another quite clumsy factor is that all edits are first stored locally, and the user must push the ``Synchronize`` button to send it to the repository.

Conclusion

It is quite obvious where developers of Cube°n take their inspiration. Cube°n offers most of the functionality of Mylyn; however, it is still quite behind its "older brother" in usability. At the moment though, there is nothing better for NetBeans IDE and at least there is some kind of tool for these users.

4.3 CodeBeamer ALM plug-in

Homepage:	http://www.javaforge.com/wiki/41101 (NetBeans version) https://codebeamer.com/cb/project/265 (Eclipse version)
Supported IDE:	Eclipse, NetBeans
Supported TS:	CodeBeamer
License:	Open source
Development status:	Production/Stable



Screen shot 60: CodeBeamer ALM plug-in for NetBeans

Intland Software offers to CodeBeamer also a plug-in, which brings their product features to both Eclipse and NetBeans IDE. With CodeBeamer ALM plug-in one can:

- connect to a CodeBeamer 5.0 server and access projects with one`s IDE;
- connect to one or more servers at the same time;
- work with multiple projects on different servers;
- display projects in an Explorer window;
- access tracker, forums, build, reports, and documents in a project;
- create a new tracker and forums;
- create, edit, delete, modify tracker items and forum postings;
- filter a tracker and forums with Groovy filter;
- access all documents that are part of a project; and
- automate project and process management using Groovy scripts that can control all project resources.

The difference between Mylyn/Cube°n and CodeBeamer ALM plug-in can be seen at first sight. The CodeBeamer ALM plug-in does not try to be universal, but focuses only one issue tracking system and tries to provide all its features.

[TSI-INT1] [TSI-INT3]

Installation

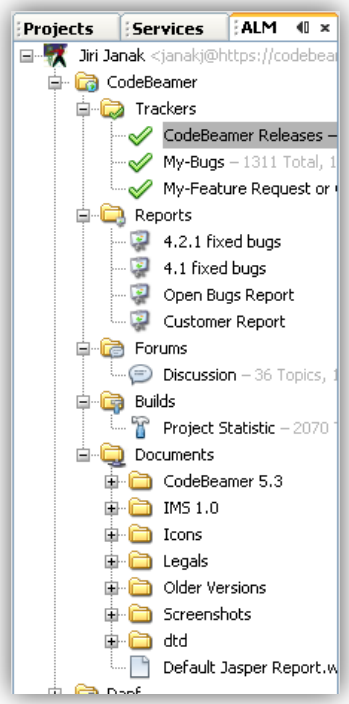
The CodeBeamer ALM plug-in is distributed as a package of modules, which need to be manually imported into the IDE plug-in repository. It is not as complicated as it may sound - everything is well-explained on the plug-ins home page.

[TSI-INT3]

User Interface

As usual UIs offer a few components; however, in this case configuration and usage of this plug-in is perhaps the easiest to understand so far. Usability designers have done a very good job.

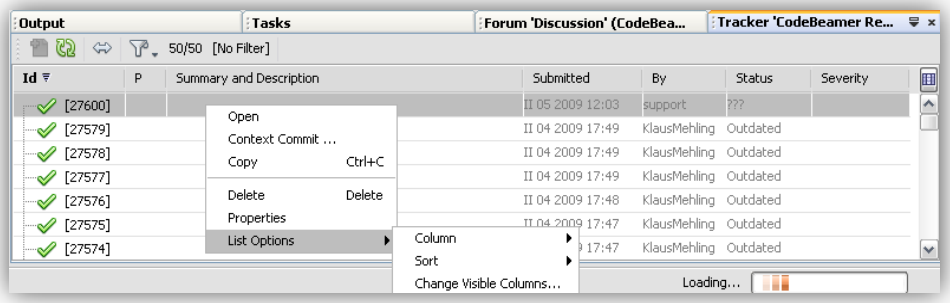
1. **Project Explorer**
   From this window, all project items can be easily managed. The user just chooses from the tree what he or she finds interesting. There is no problem to have multiple projects opened at the same time, same as having opened connections to multiple CodeBeamer servers.



Screen shot 61: CodeBeamer ALM plug-in - Project Explorer

2. Results View

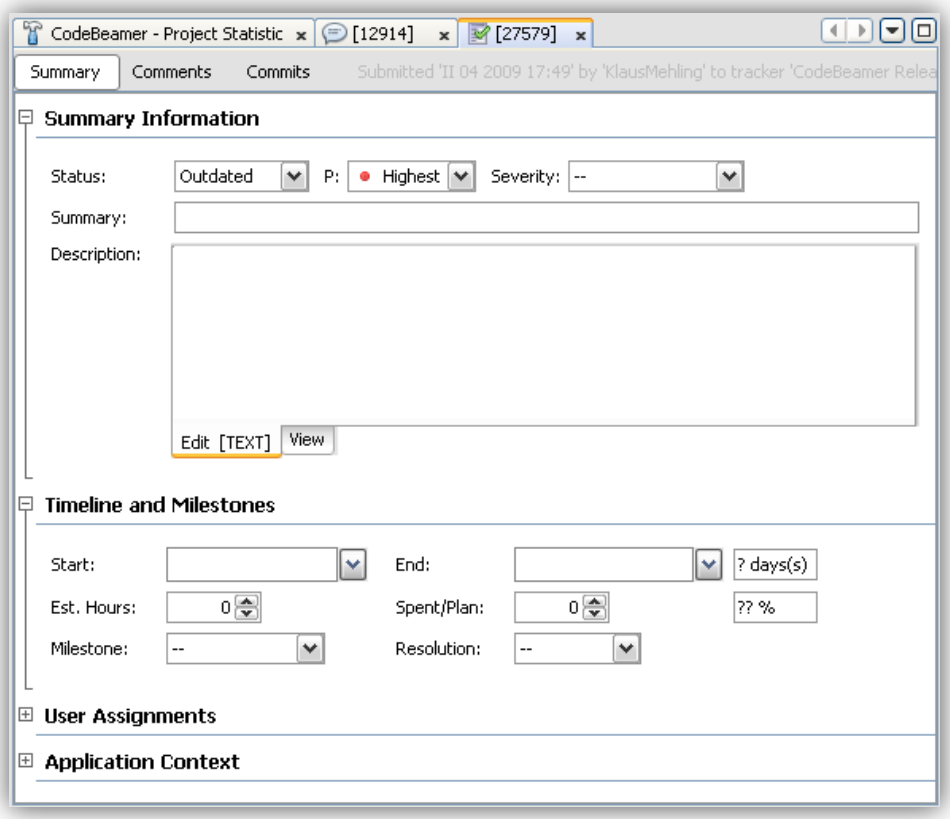
As a query is selected in the Project Explorer or a search is conducted, it is here where the results are displayed as a list of items with which one can very easily work.



Screen shot 62: CodeBeamer ALM plug-in - Results View

3. Editor

Within the editor, the user can view and edit all things managed by projects. Not only can one issue tickets in Tracker, but also discussion threads, reports, wiki pages, etc. All features are located in one place.



Screen shot 63: CodeBeamer ALM plug-in - Editor

Conclusion

The CodeBeamer ALM plug-in is a top product. It allows one to easily work with all what a CodeBeamer application can offer. A user interface is simple to use and very intuitive. Three main windows handle all the operations – that is the main benefit this plug-in brings. It is also very good since it is available for both Eclipse and NetBeans IDE environments.

One small minus is that sometimes the plug-in was quite slow – probably because it was loading too much of data at once from CodeBeamer (e.g. there was a query which should have returned about 200 results, and CodeBeamer tried to download all of them at once and the author had to wait until it finished before being able to do any further actions – the whole IDE stacked), but this can be and probably will be fixed in one of the upcoming versions.

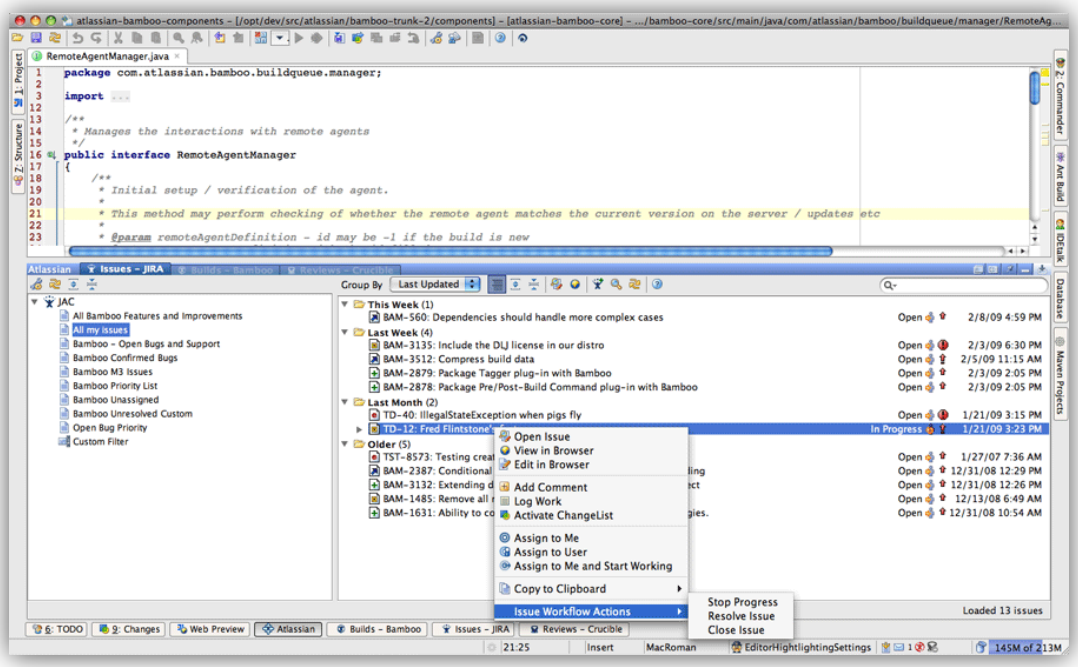
4.4 Atlassian IDE Connector

The Atlassian IDE Connector is an add-on that allows a user to work with the Atlassian products within IDE. Now, one does not have to switch between websites, email messages, and new feeds to see what is happening to the project and code. Instead, the relevant JIRA issues are seen. Crucible reviews and Bamboo build information is located there, in the development environment.

The Atlassian IDE Connector is available for IntelliJ IDEA and Eclipse. In the future, the author would like to support other IDEs too, such as Visual Studio.

Atlassian IntelliJ Connector

Homepage:	http://confluence.atlassian.com/display/IDEPLUGIN/IDE+Connector+Documentation
Supported IDE:	IntelliJ IDEA
Supported TS:	Atlassian JIRA (+ Crucible, Bamboo, FishEye)
License:	Open source
Development status:	Production/Stable



Screen shot 64: Atlassian IntelliJ Connector

Contrary to the Eclipse connector, mentioned below, the IntelliJ connector is built from by Atlassian. It uses RESTful web services for communication with Atlassian products. Currently, it supports the following:

- JIRA – for issue tracking;
- Crucible – for code reviews;
- Bamboo – for continuous integration; and
- FishEye – for source control system repository management.

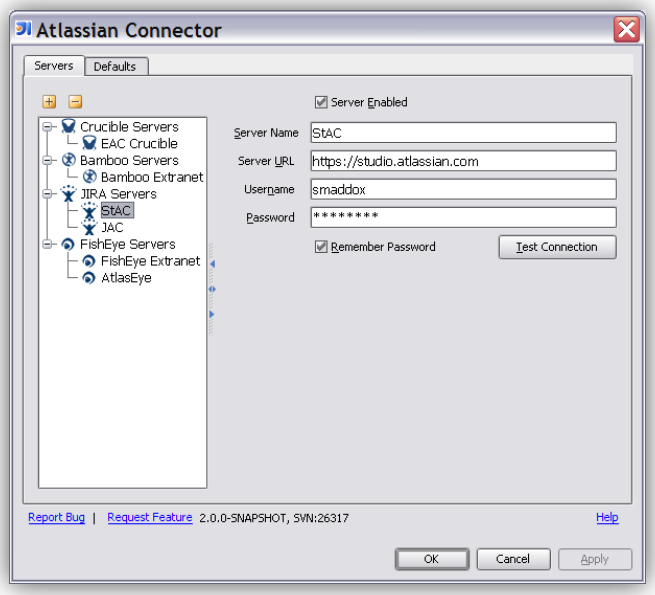
[TSI-ATL1] [TSI-ATL2]

Installation

Installing the IntelliJ Connector is quite easy:

1. Open the IDEA plug-in manager.
2. Right-click 'Atlassian IDE Connector' in the 'Available' plug-ins tab.
3. Select 'Download and Install'.

More complicated is to find out if the plug-in is working. After restarting, nothing new appears in the IDE until the connection is configured to the servers in 'Settings'.



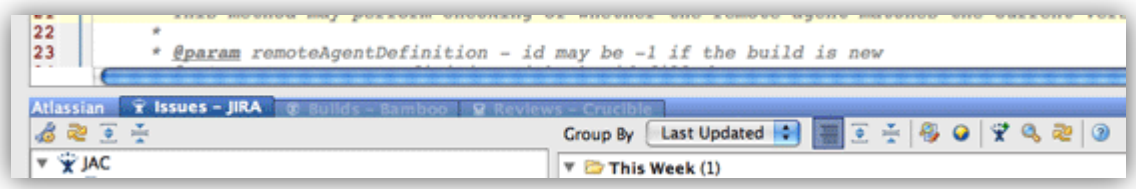
Screen shot 65: Atlassian IntelliJ Connector – Servers configuration

This can be quite confusing. Having a button visible somewhere or a new menu item would be better; however, this might be the usual style of work with IDE. The author is not a user of this IDE.

[TSI-ATL3]

User Interface

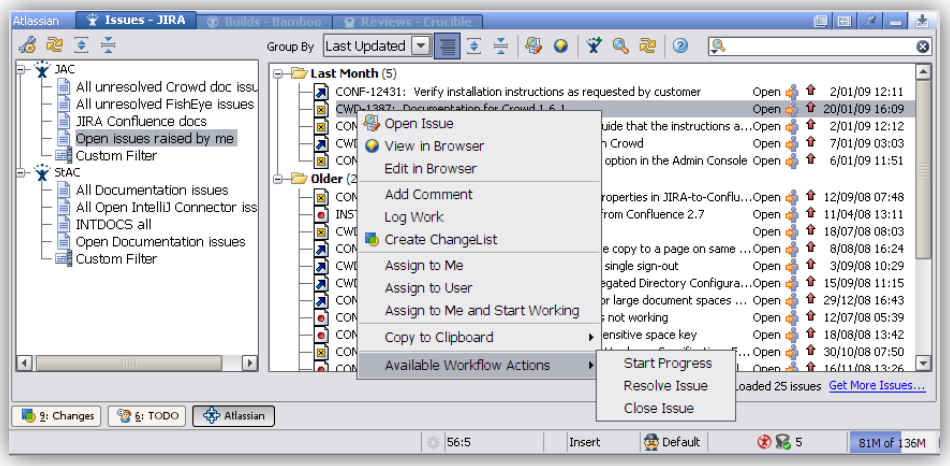
After configuration of connections, the screen is divided into two halves. The top is usually the IDE workspace, the bottom is the Atlassian Connector UI.



Screen shot 66: Atlassian IntelliJ Connector – Workspace divider with tabs

Tabs to switch between individual product windows are at the top of the Connector UI. There are four tabs currently available (depending on configuration) with the following functions:

1. JIRA tab (for Issue tracking & Request management)

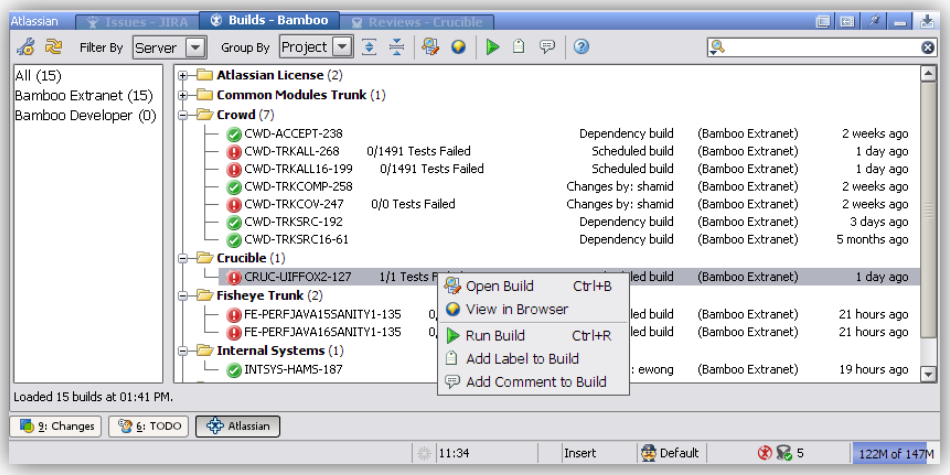


Screen shot 67: Atlassian IntelliJ Connector – JIRA tab

- View a filtered list of issues;
- Create a new JIRA issue;
- Comment on a JIRA issue and view comments;
- Create a changelist from a JIRA issue;
- Log work on a JIRA issue;
- View a JIRA issue in an IDE output tool window;
- View stack traces from a JIRA issue and click through to the relevant source file;
- Assign an issue to one`s self or another user; and
- Perform workflow actions on a selected issue;

Additional information can be found in the user guide: http://confluence.atlassian.com/display/IDEPLUGIN/Working+with+JIRA+Issues+in+IDEA

2. Bamboo tab (for Continuos integration)

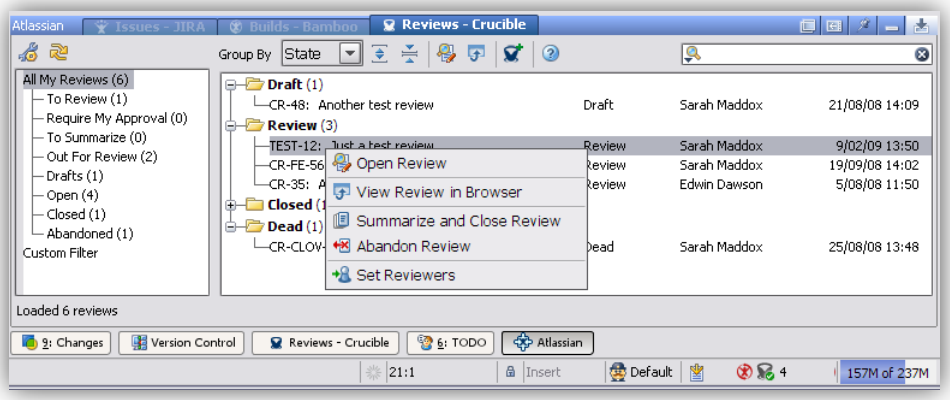


Screen shot 68: Atlassian IntelliJ Connector – Bamboo tab

- Receive notifications of failed builds;
- Re-run a build;
- Open the Bamboo build details in an IDEA output tool window;
- View a Bamboo build log;
- View failed tests and stack traces;
- Click a link in a stack trace to go directly to the code that failed;
- Re-run a failed test;
- View changed files;
- Compare the build version of a file with your local version;
- Compare the build version of a file with the previous repository version;
- Open the repository version of a file in your IDEA editor;
- Comment on a Bamboo build; and
- Label a Bamboo build.

Additional information can be found in the user guide: http://confluence.atlassian.com/display/IDEPLUGIN/Working+with+Bamboo+Builds+in+IDEA

3. Crucible tab (for Code review)

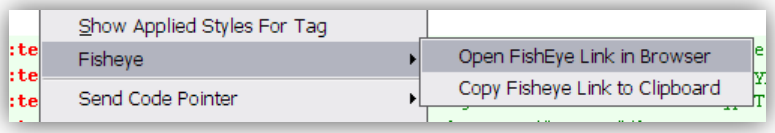


Screen shot 69: Atlassian IntelliJ Connector – Crucible tab

- Receive notifications of new and updated reviews;
- View your filtered reviews within IDE;
- View the review details and comments in an IDEA output tool window;
- Create a review;
- View the source code that is under review;
- View the diff;
- Add a review comment;
- Add a review comment on a source code line;
- Move to the commented code in the source view;
- Add a changelist to an existing review;
- Move a review through its workflow; and
- Complete a review.

Additional information can be found in the user guide:
http://confluence.atlassian.com/display/IDEPLUGIN/Working+with+Crucible+Reviews+in+IDEA

4. **FishEye support (for Source control systems repository management)**



Screen shot 70: Atlassian IntelliJ Connector – FishEye support

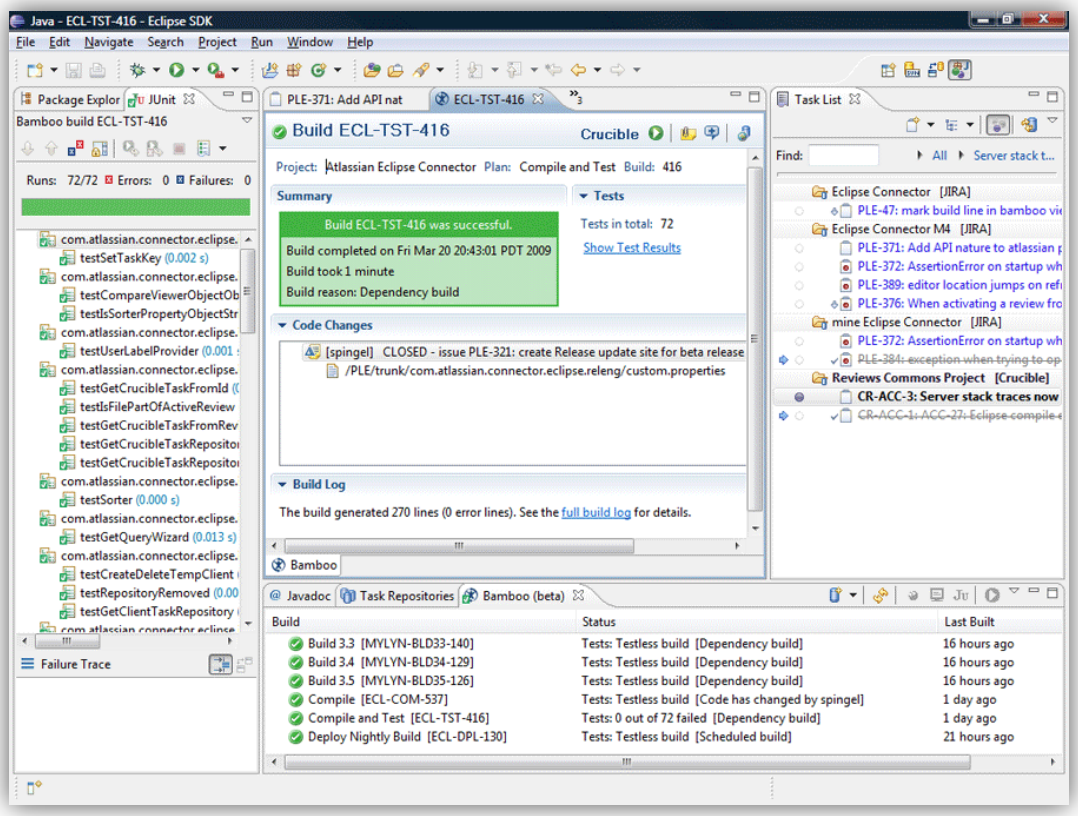
- Open a file in IDEA by supplying its FishEye URL;
- Open a source file in FishEye's web interface, with just one click from IDE; and
- Use the copy to clipboard option to share references to your file with others.

Additional information can be found in the user guide:
http://confluence.atlassian.com/display/IDEPLUGIN/Working+with+your+FishEye+Repository+View+in+IDEA

Atlassian Eclipse Connector

Homepage:	http://confluence.atlassian.com/display/IDEPLUGIN/Atlassian+Eclipse+Connector
Supported IDE:	Eclipse
Supported TS:	Atlassian JIRA (+ Crucible, Bamboo)
License:	Open source
Development status:	Beta

The Eclipse version of the connector is based on the Mylyn plug-in. Actually, it is just a bundled Mylyn with a JIRA connector and there are two added modules for working with Crucible and Bamboo, which were ported to the Eclipse platform from the previously-mentioned IntelliJ Connector. Due to this, everything that was said about Mylyn is valid also for this connector.



Screen shot 71: Atlassian Eclipse Connector

[TSI-ATL1]

Conclusion

The Atlassian community developed two plug-ins, which allows users to fully control their Atlassian development tools within IntelliJ IDEA or Eclipse. IDEA Connector is much more advanced and much more usable than its younger sibling for Eclipse, which is just an enhanced Mylyn; however IDEA Connector is already in version 2.0 and Eclipse Connector is still in the beta version.

IDEA Connectors user interface is simple to use and very intuitive and working with it was really simple. The Plug-in itself communicated with Atlassian tools very fast – the problems which the author had with the CodeBeamer ALM plug-in seemed to be solved in this product.

The next chapter will describe the development of the Atlassian NetBeans Connector, which is going to be inspired above all by the IDEA Connector; however, interesting ideas from other (previously mentioned) plug-ins will be also used.

5 Atlassian NetBeans Connector

NetBeans IDE is Java-based development environment. Nowadays, when a developer works on a project which uses, for example, JIRA for issue tracking, he or she must constantly switch between the development environment and browser, where tickets are opened for processing.

The goal of this work should be the development of a plug-in for NetBeans IDE, which would make work with Atlassian JIRA easier and faster, same like it is provided by Eclipse or IntelliJ IDE connectors mentioned in the previous chapter.

However as was already mentioned Atlassian provides a variety of advanced web-oriented applications for software development. So I have decided not to create just another one-product-only tool, but to co-operate with Atlassian developers and start the development of more complex plug-in, like is their IntelliJ Connector.

5.1 Specification & Design

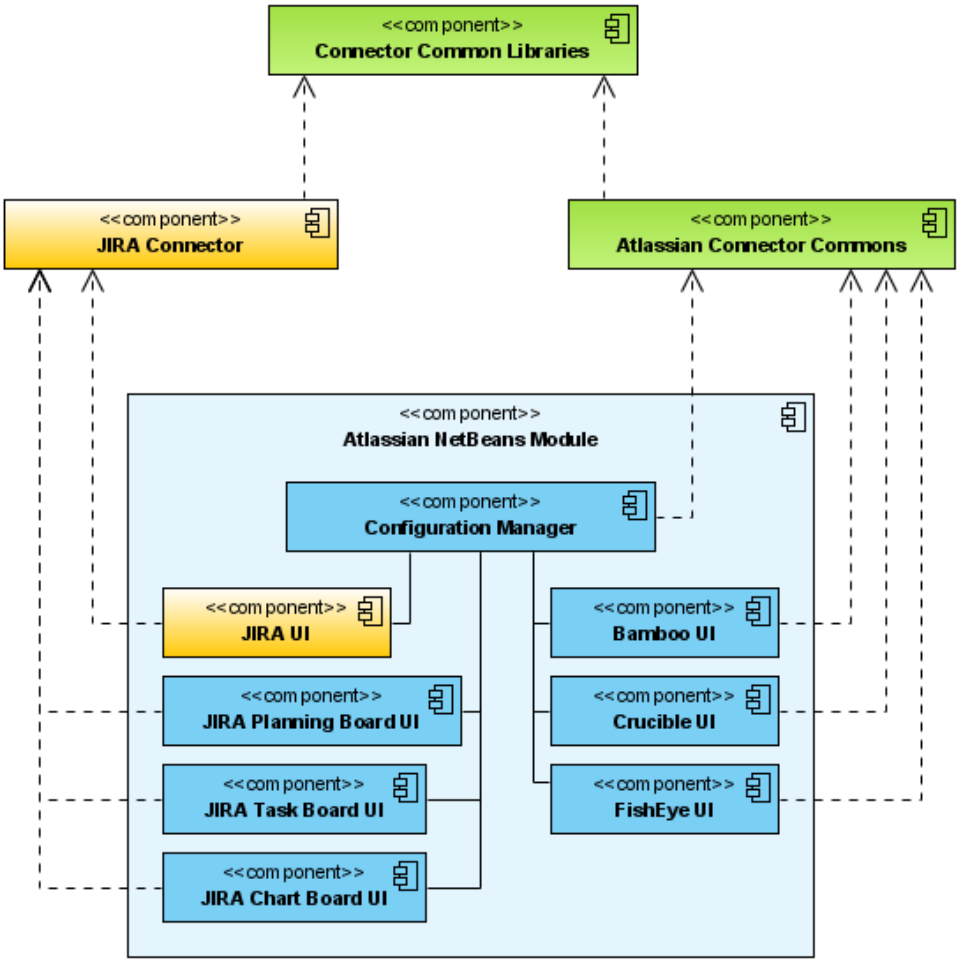


Figure 13: Atlassian NetBeans Connector components

Atlassian Connector Commons library

Due to its tight coupling with IDEA it was impossible to simple port IntelliJ Connector to NetBeans Platform. So together with Atlassian developers we have started to separate application logic from IntelliJ Connector to separate library, and result is Atlassian Connector Commons.

This library currently provides following functionality:

- Storage of all connection and configuration details. For persistance are used local XML files.
- Working with Atlassian Bamboo, Crucible and FishEye through REST web services.

Also this library is not finished yet. As I have already mentioned – the original IntelliJ Connector is very tightly coupled with IntelliJ IDEA IDE, so separating application logic is very complicated. However in it's current version 2.3.0 it is already stable and usable.

JIRA Connector library

As Atlassian is not satisfied with JIRAs webservice interface, they are currently working on new one. This lead us to decision not to include into Atlassian Connector Commons library, but instead of that to make separate library used just for connecting JIRA.

When making this library I have decided for using SOAP protocol, as it is used also by original IntelliJ Connector.

Atlassian NetBeans Module

As Atlassian Connector Commons and JIRA Connector contain application logic, this module contains presentation layer of whole application. ***In release 1.0 only JIRA UI is available*** (as it was primary goal of this thesis), providing following services within NetBeans IDE:

1. *JIRA UI (for Issue tracking & Request management)*
    - View a filtered list of issues;
    - Create a new JIRA issue;
    - Comment on a JIRA issue and viewing comments;
    - Create a changelist from a JIRA issue;
    - Log work on a JIRA issue;
    - View a JIRA issue in an IDE output tool window;
    - View stack traces from a JIRA issue and click through to the relevant source file;
    - Assign an issue to yourself or another user;
    - Perform workflow actions on a selected issue.

UI packages for other Atlassian applications are under development. In the end they should enable to perform following use-cases within NetBeans IDE:

2. *Configuration Manager*
    - Create, read, update, and delete connection details for multiple JIRA, Bamboo, Crucible, FishEye servers;
    - Test connection.

3. *Bamboo UI (for Continuos integration)*
    - Receive notifications of failed builds;
    - Re-run a build;
    - Open the Bamboo build details in an IDE output tool window;
    - View a Bamboo build log;
    - View failed tests and stack traces;
    - Click a link in a stack trace to go directly to the code that failed;
    - Re-run a failed test;
    - View changed files;
    - Compare the build version of a file with your local version;
    - Compare the build version of a file with the previous repository version;
    - Open the repository version of a file in your IDE editor;
    - Comment on a Bamboo build;
    - Label a Bamboo build.

4. *Crucible UI (for Code review)*
    - Receive notifications of new and updated reviews;
    - View your filtered reviews within IDE;
    - View the review details and comments in an IDE output tool window;
    - Create a review;
    - View the source code that is under review;
    - View the diff;
    - Add a review comment;
    - Add a review comment on a source code line;
    - Move to the commented code in the source view;
    - Add a changelist to an existing review;
    - Move a review through its workflow;
    - Complete a review.

5. *FishEye UI (for Source control systems repository management)*
    - Open a file in IDE by supplying its FishEye URL;
    - Open a source file in FishEye's web interface;

There are also mentioned JIRA Planning Board UI, Task Board UI and Chart Board UI. This is basically about the idea of making open source and free version of JIRA GreenHopper plug-in.

5.2 Deployment & Configuration

Deployment of the module

1. Download current deployment ZIP package.
2. Extract it into some temporary directory.
3. Open NetBeans IDE.
4. Select “Tools -> Plugins” from menu.
5. Select “Downloaded” tab and push “Add plugins…” button.
6. Select all provided NBM files and push Open button.
7. Modules will be installed.

Configuring servers

1. Select “Tools -> Options” from menu.
2. Select “Atlassian Connector” tab.
3. Push “Add server” button and provide connection details.

5.3 Conclusion

Release 1.0 was primary about creating Atlassian Connector Commons library, JIRA Connector library and JIRA UI as headstone of Atlassian Connector Module. Work on other UI packages are in progress. Recent projects state can be checked can be seen on projects home site.

Homepage:	http://kenai.com/projects/anc
Recent source code:	https://kenai.com/svn/anc~subversion/trunk/
Issues tracking:	http://kenai.com/jira/browse/ANC

6 Conclusion

Request management is an important part of every software project and using issue tracking systems is necessary, which brings us following benefits:

- improvement the quality of software;
- increase of the satisfaction of users and customer;
- ensuring requests accountability;
- improvement communication in the team and also to the customers;
- increase of productivity of the team; and
- the reduction of expenses.

However, even the best issue tracking system does not help if it does not have a proper request workflow and the responsible team does not learn to report requests correctly. There is enough to keep four basic rules:

1. **Be specific.** A detailed description should state what the awaited result is and what is occurring instead – what one may consider to be a mistake.

2. **Show as much as possible.** If the program writes an error message, it must be copied into the report. Adding screenshots to bug reports as much as possible is very important.

3. **Describe how to reproduce the problem. A** programmer will need to reproduce the problem on his or her own computer. After writing down how to do it, the user should go through these instructions him or herself. The environment in which the problem appeared must be described –computers configuration, operating system, network connection, browser, version of Java… Anything possibly associated with an error should be recorded. A good rule of thumb is to write more rather than less.

4. **Clearly differentiate between facts and assumptions.** Sometimes the bug is hidden elsewhere and with assumptions could lead the programmer to a false track. The focus should be on the symptoms and diagnosis should be left to the experts.

Which issue tracking system to use depends on the kind of a project being worked on, and on available resources (especially financial resources); however, from the author`s point of view, Atlassian JIRA is the best available product on the market at the moment - not only because of its possibilities of customization, extension, and collaboration, but also due to its distribution politics, which try to make JIRA available for everybody.

Due to this, the author has developed, in collaboration with Atlassian, a connector for their product for NetBeans IDE. The connector works, is available on the Atlassian website for download, and its development will continue.

Bibliography

Modern issue tracking

[MIT-AND] Andrle, T. (2006, June 4th). Jak správně hlásit chyby. Retrieved from Tomovo blog: http://cattleshow.net/tom/blog/archives/88

[MIT-ATL] Atlassian Pty Ltd. (2008). Issue Types. Retrieved from JIRA - Local help: http://jira.atlassian.com/secure/ShowConstantsHelp.jspa?decorator=popup

[MIT-BIR] Birlasoft Ltd. (2008). Bug Life Cycle. Retrieved from Scribd.com: http://www.scribd.com/doc/7289591/Bug-Life-Cycle

[MIT-BLA] Blair, S. (2004, October). A Guide to Evaluating a Bug Tracking System. Retrieved from Scribd.com: http://www.scribd.com/doc/7046090/A-Guide-to-Evaluating-a-Bug-Tracking-System

[MIT-GLE] Glenfils AG. (2008). ISO/IEC 20000 IT Service Management Standard. Retrieved from ITIL.org: http://www.itil.org/en/isoiec20000/index.php

[MIT-HAU1] Hauner, A. (2008, November). Bug tracking pro vývojáře. Retrieved from SoftEU s.r.o. Blog: http://blog.softeu.cz/prednasky/2008/prednaska-bug-tracking-pro-vyvojare/

[MIT-HAU2] Hauner, A. (2008, November). Hlásíme chyby v software. Retrieved from SoftEU s.r.o. Blog: http://blog.softeu.cz/prednasky/2008/prednaska-hlasime-chyby-v-software/

[MIT-IBA] IBA CZ, s.r.o. (2008). Úvod do CMMI.

[MIT-KAN] Kaner, C. (2002). Bug Advokacy. Retrieved from Cam Kaner Home Page: http://www.kaner.com/pdfs/BugAdvocacy.pdf

[MIT-LIV] LiveTime Software. (2008). ISO/IEC 20000 Compliance. Retrieved from Web 2.0 ITIL Service Desk, Help Desk and Support Software: http://www.livetime.com/webservicedesk/ServiceCompliance.html

[MIT-LUD] Ludwig Consulting Services, LLC. (2009). Practical information about requirements management. Retrieved from http://www.jiludwig.com/

[MIT-PRF] Profinit, s.r.o. (2008). Přednáška č. 7 - Configuration management. Retrieved from Softwarové inženýrství v praxi: http://www.profinit.eu/cz/podpora-univerzit/univerzitni-vyuka/swi129/harmonogram/p07_cm

[MIT-PRO] Procházka, P. (2008). Mapování a modelování jako nástroj pro zlepšování procesů. Brno: Ekonomicko správní fakulta Masarykovy univerzity.

[MIT-RAT] Rational Software. (2001). Rational Unified Process - Best Practices for Software Development Teams. Lexington: Rational Software.

[MIT-WIB] Wibas GmbH. (2008). Understanding Capability Levels. Retrieved from wibas CMMI Browser: http://www.cmmi.de/cmmi_v1.2/capabilitylevels.html#hs:null

[MIT-WIK1] Wikipedia. (n.d.). Capability Maturity Model Integration. Retrieved from http://en.wikipedia.org/wiki/CMMI

[MIT-WIK2] Wikipedia. (n.d.). Change Management (ITSM). Retrieved from http://en.wikipedia.org/wiki/Change_Management_(ITSM)

[MIT-WIK3] Wikipedia. (n.d.). Change Request. Retrieved from http://en.wikipedia.org/wiki/Change_request

[MIT-WIK4] Wikipedia. (n.d.). IBM Rational Unified Process. Retrieved from http://en.wikipedia.org/wiki/Rup

[MIT-WIK5] Wikipedia. (n.d.). ISO 9000. Retrieved from http://en.wikipedia.org/wiki/ISO9000

[MIT-WIK6] Wikipedia. (n.d.). Software bug. Retrieved from http://en.wikipedia.org/wiki/Software_bug

[MIT-WIK7] Wikipedia. (n.d.). Software development process. Retrieved from http://en.wikipedia.org/wiki/Software_development_process

[MIT-XIN] Xin, J. A. (2008). CMMI Glossary. Retrieved from CMM Practice: http://jiangxin.worldhello.net/doc/cmm_practice/cmm-gloss.html

Tracking systems overview

[TSO-AND] Andrle, T. (2006, May 18th). Trac. Retrieved from Tomovo blog: http://cattleshow.net/tom/blog/archives/85

[TSO-ATL1] Atlassian Pty Ltd. (n.d.). Atlassian Software - Products. Retrieved from http://www.atlassian.com/software/

[TSO-ATL2] Atlassian Pty Ltd. (n.d.). Watch the "JIRA & Greenhopper Case Study" Webinar Recording. Retrieved from The Atlassian Blog: http://blogs.atlassian.com/news/2008/11/webinar_recordi.html

[TSO-COS] CosmoCode GmbH. (2008). CodeBeamer. Retrieved from WikiMatrix: http://www.wikimatrix.org/show/codeBeamer

[TSO-CVR] Cvrček, P. (2008, February 2nd). Vyšla Bugzilla 3.2. Retrieved from SoftEU s.r.o. Blog: http://blog.softeu.cz/vysla-bugzilla-32/

[TSO-EDG] Edgewall Software. (2009). The Trac User and Administration Guide. Retrieved from http://trac.edgewall.org/wiki/TracGuide

[TSO-GAR] Garg, R. (2005, September 15th). Installing Bugzilla on Windows. Retrieved from Rajneesh's Logbook: http://rgarg.blogspot.com/2005/09/installing-bugzilla-on-windows.html

[TSO-GRE1] GreenPepper Software. (2009). GreenHopper for JIRA. Retrieved from http://www.greenpeppersoftware.com/en/products/greenhopper-jira/

[TSO-GRE2] GreenPepper Software. (2009). GreenHopper Intro. Retrieved from http://www.greenpeppersoftware.com/greenhopper-videos/greenhopper-intro/index.html

[TSO-GRE3] GreenPepper Software. (2008). GreenHopper JIRA Features. Retrieved from http://www.greenpeppersoftware.com/confluence/display/GH/FEATURES

[TSO-IBM1] IBM. (2008). Rational ClearQuest. Retrieved from http://www.ibm.com/developerworks/rational/products/clearquest

[TSO-IBM2] IBM. (2006). *IBM Rational Team API*. Retrieved from http://www.alphaworks.ibm.com/tech/teamapi/

[TSO-INI] Internet Info, s.r.o. (2005, September). Jednoduchy webovy issue tracking system. Retrieved from Root.cz: http://www.root.cz/diskuse/375/

[TSO-INS1] Intland Software. (n.d.). CodeBeamer. Retrieved from http://www.intland.com/products/codebeamer.html

[TSO-INS2] Intland Software. (n.d.). CodeBeamer Knowledge Base. Retrieved from https://codebeamer.com/cb/project/37

[TSO-KOL] Kolář, R. (2007, April 2nd). Trac. Retrieved from Linuxsoft.cz: http://www.linuxsoft.cz/article.php?id_article=1439

[TSO-LAW] Lawlor, B. (2007, November 12th). Managing Projects with JIRA and Greenhopper. Retrieved from DeCare Systems Ireland Blog: http://blog.decaresystems.ie/index.php/2007/11/12/managing-projects-with-jira-and-greenhopper/

[TSO-MOZ1] Mozilla Foundation. (2008). Bugzilla Homepage. Retrieved from https://wiki.mozilla.org/Bugzilla

[TSO-MOZ2] Mozilla Foundation. (2008). Integrating Bugzilla with Third-Party Tools. Retrieved from Bugzilla Guide: http://www.bugzilla.org/docs/2.18/html/integration.html

[TSO-ONL] Onlio, a.s. (2008). JIRA. Retrieved from Onlio HomePage: http://www.onlio.com/produkty/jira.html

[TSO-SMA] Smart, J. F. (2007, March 14th). What issue tracking system is best for you? Retrieved from JavaWorld.com:

http://www.javaworld.com/javaworld/jw-03-2007/jw-03-bugs.html

[TSO-STA] Stahl, P. (2007, December 5th). Trac - systém pro správu projektů. Retrieved from Scream.cz: http://www.scream.cz/2007/12/05/trac-system-pro-spravu-projektu-1-uvod/

[TSO-WIK1] Wikipedia. (n.d.). Comparison of issue tracking systems. Retrieved from http://en.wikipedia.org/wiki/Comparison_of_issue_tracking_systems

[TSO-WIK2] Wikipedia. (n.d.). IBM Rational ClearQuest. Retrieved from http://en.wikipedia.org/wiki/IBM_Rational_ClearQuest

[TSO-WIK3] Wikipedia. (n.d.). JavaForge. Retrieved from http://en.wikipedia.org/wiki/JavaForge

[TSO-WIK4] Wikipedia. (n.d.). Trac. Retrieved from http://en.wikipedia.org/wiki/Trac

Tracking systems IDE integration

[TSI-ATL1] Atlassian Pty Ltd. (n.d.). Atlassian IDE Connector. Retrieved from http://www.atlassian.com/software/ideconnector/

[TSI-ATL2] Atlassian Pty Ltd. (n.d.). Atlassian IntelliJ Connector 2.0 Now Available. Retrieved from The Atlassian Blog: http://blogs.atlassian.com/news/2009/02/atlassian_intel.html

[TSI-ATL3] Atlassian Pty Ltd. (n.d.). Introduction to the Atlassian IDE Connector. Retrieved from http://confluence.atlassian.com/display/IDEPLUGIN/IDE+Connector+Documentation

[TSI-ATL4] Atlassian Pty Ltd. (n.d.). Mylyn. Retrieved from http://confluence.atlassian.com/display/JIRAEXT/Mylyn

[TSI-ECL1] Eclipse Foundation. (n.d.). Eclipse Mylyn Open Source Project. Retrieved from http://www.eclipse.org/mylyn/

[TSI-ECL2] Eclipse Foundation. (n.d.). Mylyn Development. Retrieved from Eclipsepedia: http://wiki.eclipse.org/Mylyn

[TSI-ECL3] Eclipse Foundation. (n.d.). Mylyn Project Summary. Retrieved from http://www.eclipse.org/projects/project_summary.php?projectid=tools.mylyn

[TSI-FRE] Freeman-Manson, B. (2008, October 2nd). How Mylyn Changes the Way I Develop. Retrieved from Java-TV.com: http://www.java-tv.com/tag/mylyn/

[TSI-GUN1] Gunasekara, A. (n.d.). Anuradha weblog. Retrieved from http://theanuradha.blogspot.com/

[TSI-GUN2] Gunasekara, A. (n.d.). CubeOn project. Retrieved from Google Code: http://code.google.com/p/cubeon/

[TSI-IBM] IBM. (2005). Understanding the IBM Rational ClearQuest Client for Eclipse. Retrieved from http://www.ibm.com/developerworks//rational/library/04/r-3089/index.html?ca=dgr-eclpsw03ClearQuest

[TSI-INT1] Intland Software. (2007, October 8th). CodeBeamer & NetBeans. Retrieved from YouTube.com: http://www.youtube.com/watch?v=IAPJouj75cs

[TSI-INT2] Intland Software. (n.d.). Eclipse Plugin Home. Retrieved from https://codebeamer.com/cb/project/265

[TSI-INT3] Intland Software. (n.d.). NetBeans 6.0 ALM Plugin. Retrieved from http://www.javaforge.com/wiki/41101

[TSI-KER] Kersten, M. (2007, August 14th). Mylyn 2.0, Part 1: Integrated task management. Retrieved from developerWorks: http://www.ibm.com/developerworks/java/library/j-mylyn1/index.html

[TSI-MAR] Mareš, J. (2007, December 7th). Eclipse, Mylyn, IntelliJ IDEA a Teamcity v CZPodcast #19. Retrieved from Jirablog: http://jirablog.blogspot.com/2007/12/eclipse-mylyn-intellij-idea-teamcity-v.html

[TSI-OLO] Olofsen, K. (2009, March 23rd). Atlassian Eclipse Connector 1.0 - Beta now available. Retrieved from Atlassian Blog: http://blogs.atlassian.com/news/2009/03/eclipse_connector.html

[TSI-SRA] Srajer, M. (2007, February 28th). Mylar, aneb jak zvladnout obří projekt v Eclipse. Retrieved from Softwarové inženýrství blog: http://srakyi.modry.cz/blog/2007/02/mylar-aneb-jak-zvladnout-obri-projekt-v-eclipse/

Atlassian NetBeans Connector

[ANC-ARL] Arlow, J., & Neustadt, I. (2005). UML 2 and the unified process: practical object-oriented analysis and design (2nd ed.). Addison Wesley.

[ANC-ATL] Atlassian Pty Ltd. (n.d.). JIRA RPC Services. Retrieved from http://confluence.atlassian.com/display/JIRA/JIRA+RPC+Services

[ANC-LAU] Laukkanen, T. (2008). NetBeans JIRA Plugin. Retrieved from http://confluence.atlassian.com/display/JIRAEXT/NetBeans+JIRA+Plugin

[ANC-NET] Neto, S. (2008, February 24th). Netbeans: your first plugin. Retrieved from Personal blog: http://silveiraneto.net/2008/02/24/netbeans-your-first-plugin/

[ANC-SUN1] Sun Microsystems, Inc. (n.d.). Eclipse Plug-in to NetBeans Module Migration Cookbook. Retrieved from NetBeans Wiki:

http://wiki.netbeans.org/EclipsePluginToNetBeansModuleMigrationCookbook

[ANC-SUN2] Sun Microsystems, Inc. (n.d.). NetBeans Developer FAQ. Retrieved from NetBeans Wiki: http://wiki.netbeans.org/NetBeansDeveloperFAQ

[ANC-SUN3] Sun Microsystems, Inc. (n.d.). NetBeans Platform. Retrieved from http://platform.netbeans.org/

[ANC-SUN4] Sun Microsystems, Inc. (n.d.). NetBeans Platform Certified Training. Retrieved from http://edu.netbeans.org/courses/nbplatform-certified-training/

[ANC-TUL] Tulach, J., Epple, A., & Würthinger, T. (2008, September). Rich Client Programming - Plugging into the NetBeans Platform. Retrieved from Institut für Systemsoftware bei Johannes Kepler Universität Linz: http://www.ssw.uni-linz.ac.at/Teaching/Lectures/SpezialLVA/TulachWielenga/

[ANC-WIE1] Wielenga, G. (n.d.). Top 10 NetBeans APIs. Retrieved from NetBeans Platform: http://platform.netbeans.org/tutorials/nbm-10-top-apis.html

[ANC-WIE2] Wielenga, G., Tulach, J., & Boudreau, T. (2007). Rich Client Programming: Plugging into the NetBeans Platform. Prentice Hall.