ALMtoolbox
Productivity ALM Solutions

http://almtoolbox.com
USA & Canada: 866-368-5404
International: +972-722-405-222

Charts and Metrics included in *Development Intelligence* Solution for Rational ClearCase

Last update: June 2015

Background:

The following charts are made to assist development managers, QA managers and team leaders who rely on real-time visual information about the project status to help them make informed decisions.

These charts provide an overview of projects from several perspectives: from the project level down to baselines and releases, activities, change-sets and files, down to a code-line.

Users can export these charts to graphical JPG files or onto a dynamic dashboard accessible by a web browser.

This document demonstrates 14 charts.

About Development Intelligence

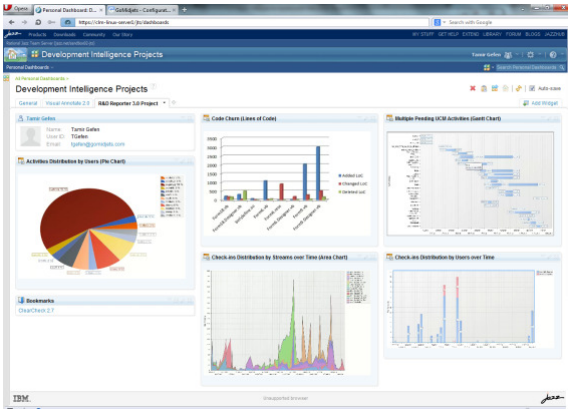
The challenge of project governance is providing the right level of reporting for each individual on the team. The needs of project managers greatly differ from those of testers, but both should be met if a project is going to stay on track. "Development Intelligence Solution" provides contextual information tailored to the individual. Each user can customize the layout and content of his or her Rational ClearCase dashboard, and because the dashboard is continually updated each user can view the most current information pertaining to his other work.

About dashboards and ClearCase Dashboard

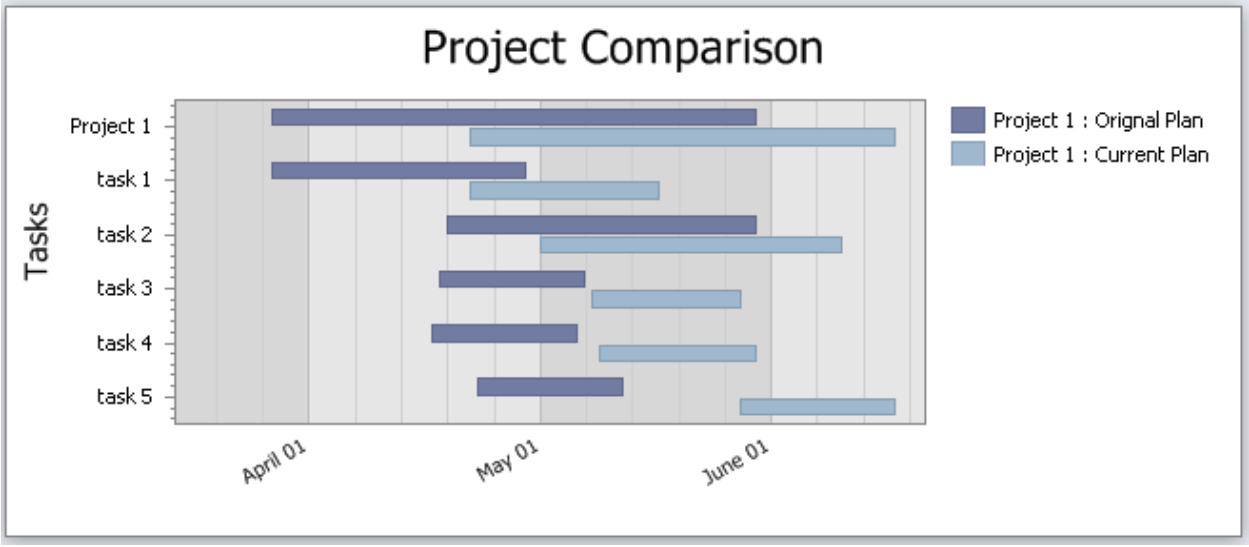
Dashboard is a user interface that, like an automobile's dashboard, organizes and presents information in an easily accessible way.

It can be customized for each stakeholder; Data can be updated in real-time through any web browser.

This example demonstrates the RTC dashboard as the data is retrieved from within ClearCase. To learn more: http://www.almtoolbox.com/blog/?p=167



1) **Project Comparison – Plan vs. Execution**



Description: Each activity is represented by a task. For each activity you can see the original plan duration (in purple) and current execution in light blue.

This helps you to easily compare plan vs. execution times, as well as the entire project's plan vs. execution (in the first line) as this chart accumulates all tasks' duration times.

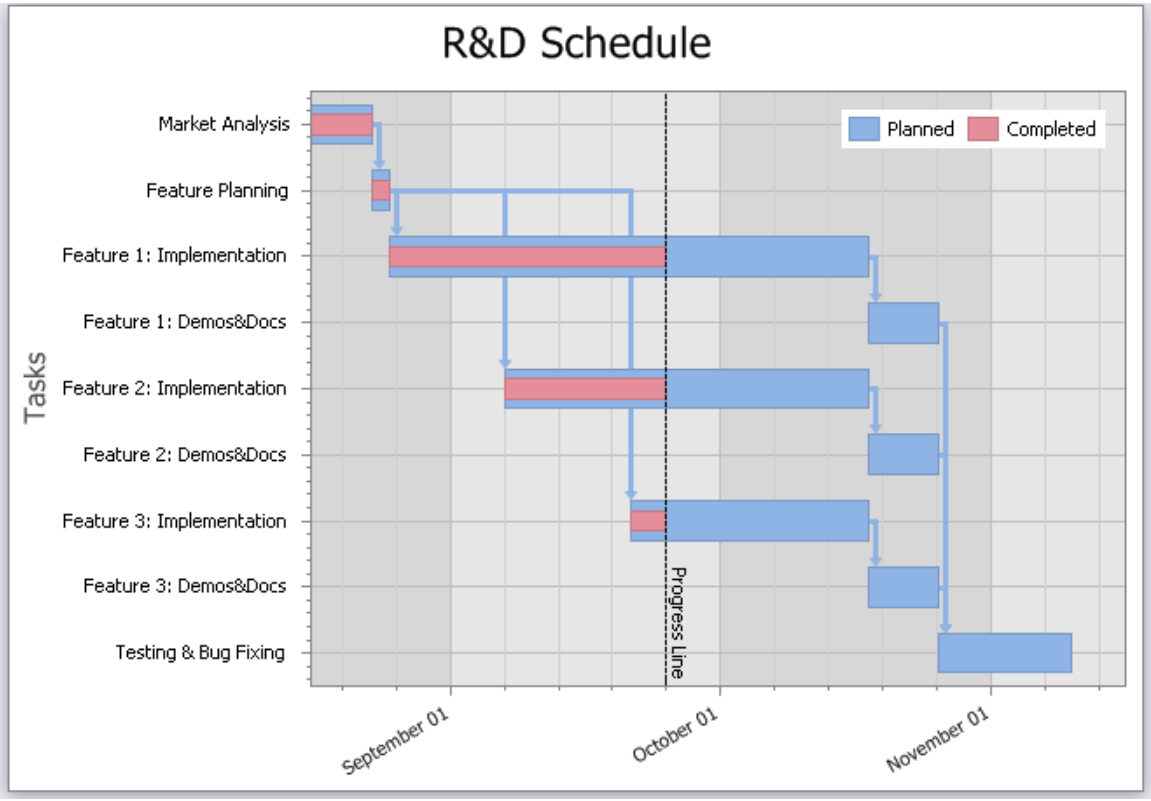
The chart is based on data retrieved from the following reports:

- Two baselines comparison report (from the same stream or different streams)
- Two streams comparison (from the same project or inter-project)
- Multiple Pending Change-sets report
- Full stream report
- Chart is displayed after the data has been filtered!

Technical requirements:

- ClearCase UCM
- Issue /Bug tracking system that exports data to CSV files (e.g. ClearQuest, RTC, Jira or BugZilla)

2) Project Comparison + Dependencies



Description: As the previous chart, each activity is represented by a task. For each activity you can see the original plan duration (in purple) and current execution in light blue.

Additionally, you can view the dependency links among tasks: you can see which tasks depend on which other tasks.

This helps you easily compare plan vs. execution times, as well as the entire project's plan vs. execution (in the first line) as this chart accumulates all tasks' duration times. Furthermore, you can see planning dependencies and easily estimate how delayed tasks affect other tasks and the entire project.

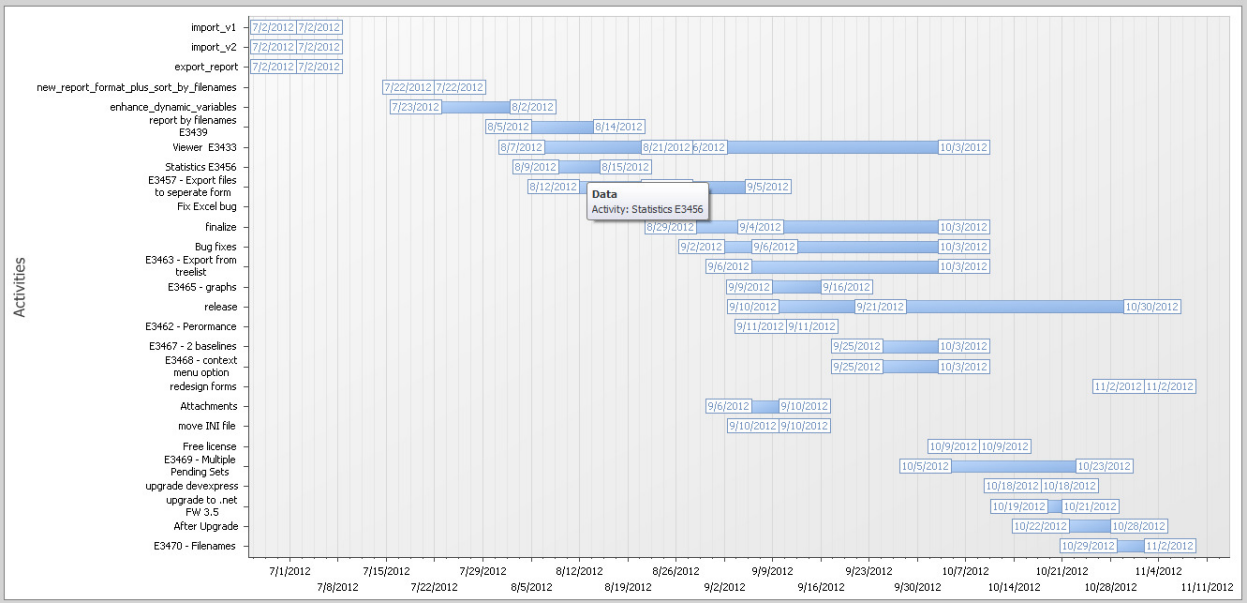
The chart is based on data retrieved from the following reports:

- Two baselines comparison report (from the same stream or different streams)
- Two streams comparison (from the same project or inter-project)
- Multiple Pending Change-sets report
- Full stream report
- Chart is displayed after the data has been filtered!

Technical requirements:

- ClearCase UCM
- Issue /Bug tracking system that exports data to CSV files (e.g. ClearQuest, RTC, Jira or BugZilla)

3) Activities Gantt chart



Description: Quite similar to the first chart shown, this chart displays the current execution information for all activities, over a period of time. For each activity it displays when the activity was created and when its last check-in was made. Users can see here a deployment of different users' activities and different projects.

This chart is highly detailed. It displays a tooltip with more information when hovering on the activities and it also offers a context-menu when clicking on the activity objects that provides more information.

Based on data retrieved from the following reports:

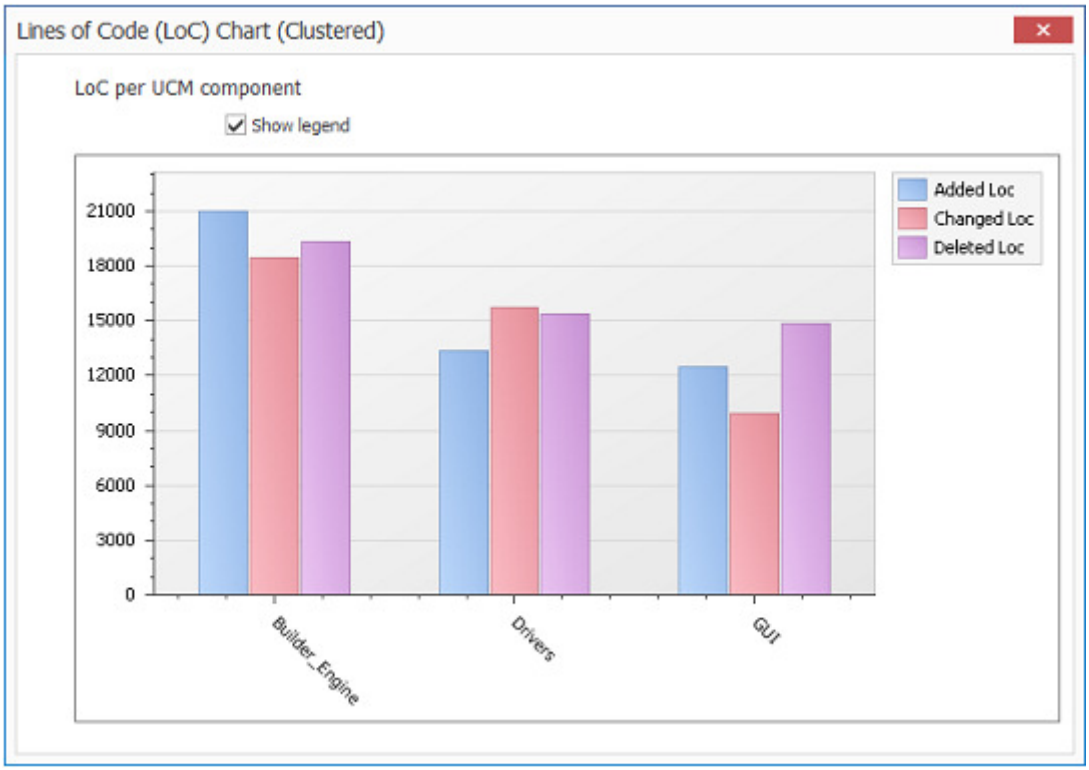
- Two baselines comparison report (from the same stream or different streams)
- Two streams comparison (from the same project or inter-project)
- Multiple Pending Change-sets report (displaying undelivered activities from all selected streams)
- Full stream report
- Chart is displayed after unnecessary data has been filtered out!

Technical requirements:

- ClearCase UCM

4) Lines of Code (LOC) per UCM component / Code Churn

Lines of Code (LOC) or Source Lines of Code (SLOC) provides a quantitative measurement for computer programming files in text form. LOC is used to measure files that contain code from a given programming language. The size of a specific file or section and work involved are typically indicated by the number of lines. To learn more: http://www.almtoolbox.com/download/lines-of-code-for-clearcase.pdf



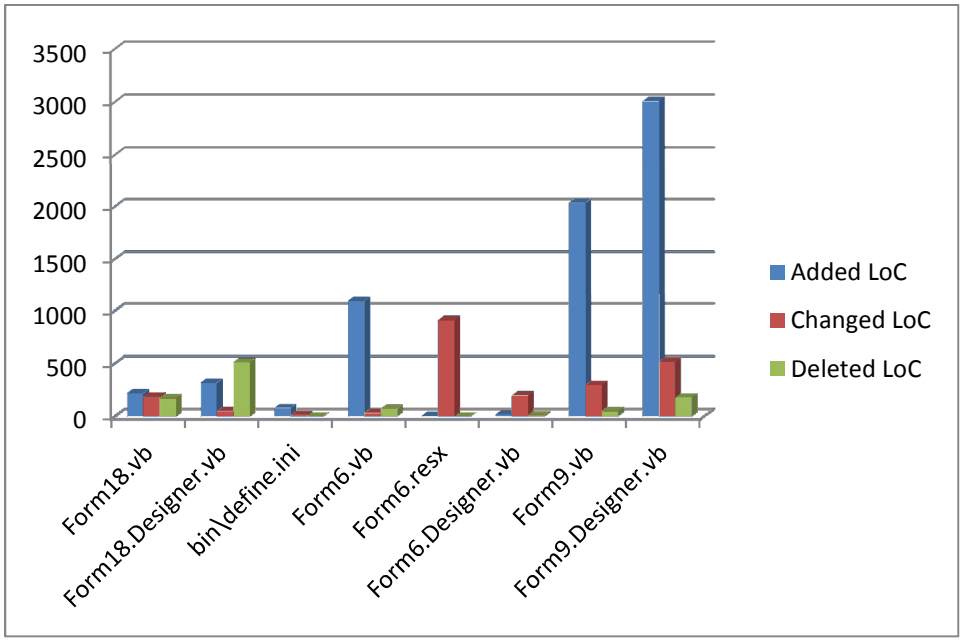
Description: This chart is part of the new "Two Composite baselines" report. When you compare two composite baselines, you get the differences for each component. We developed a new chart that completes the new report and demonstrates how many code lines have been added, removed and changed in total for each component. This enables you to see the maturity of the code in the relevant component (a component with fewer a few changes is considered to be more mature and stable).

You can export those charts into graphic files and create them automatically whenever you need.

Technical requirements:

- ClearCase UCM
- Using composite baselines

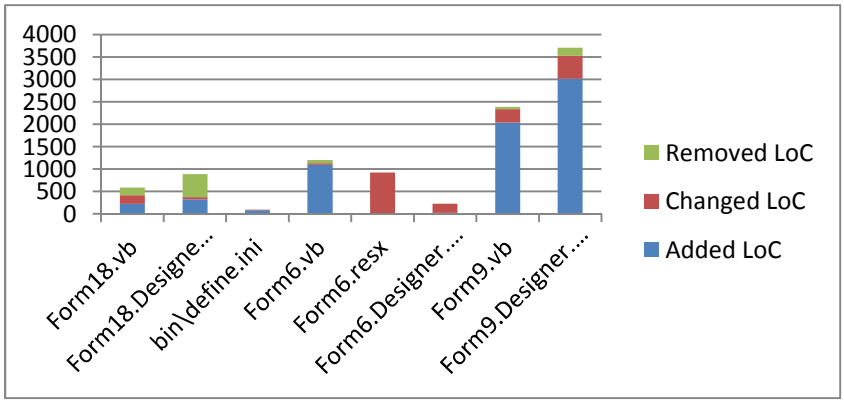
5) Lines of Code (LOC) per file / Code Churn



Description: This chart summarizes how many code lines have been added, changed and deleted for each file. The chart is based on a segment that compares two UCM baselines and can filter out unnecessary data.

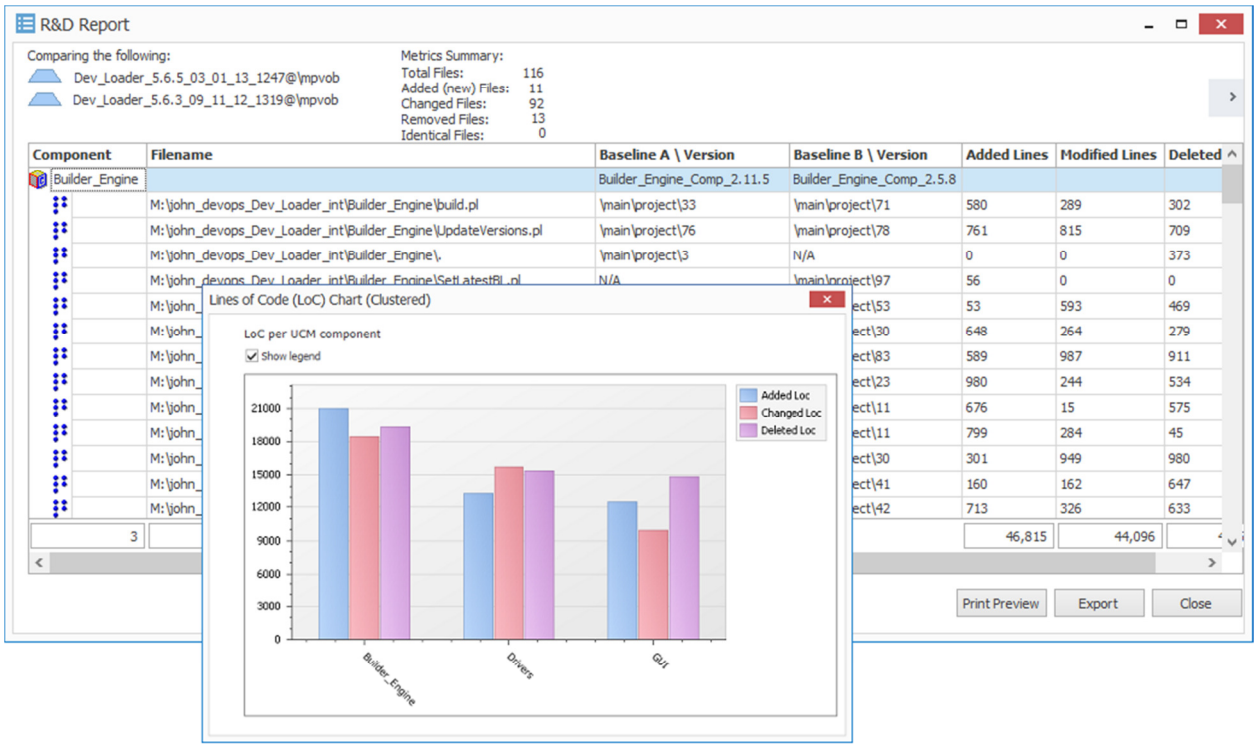
Our algorithm is based upon a well-known algorithm named CLOC. It automatically filters out empty lines and comment lines for many code-languages.

The following chart demonstrates the same data, but it accumulates added, changed and deleted lines together (for each file). It's useful when there are many files included in the report.



Based on a survey we ran, R&D managers and QA managers find those charts interesting because it helps them to measure productivity. If you measure the amount of bugs per release, you can easily obtain a quality mark based on the ratio of bugs to changed code lines.

LoC data can also be displayed on a table-based report: See the far-right columns, “Added LoC”; “Modified LoC” and “Deleted LoC”:



If you find LoC charts interesting, you can see charts (9) – (12) below which represent code lines distribution by file versions, by streams /branches, by usernames and by UCM activities.

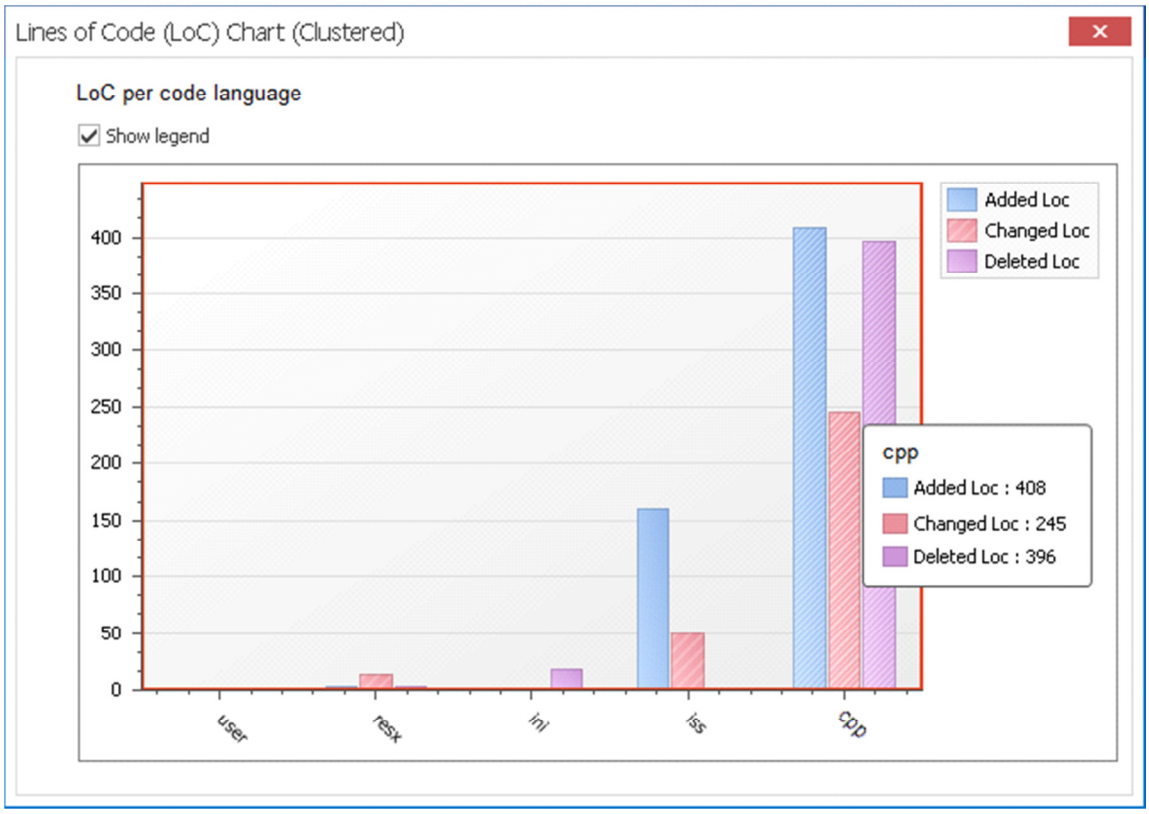
Based on data retrieved from the following reports:

- Two baselines comparison report (baselines may come from the same stream or different streams but they must share a foundation baseline)
- Chart is displayed after the unnecessary data has been filtered out!
- In the future we may enable those charts based upon on two streams comparison or even an entire stream report. To join beta, send email to beta@almtoolbox.com

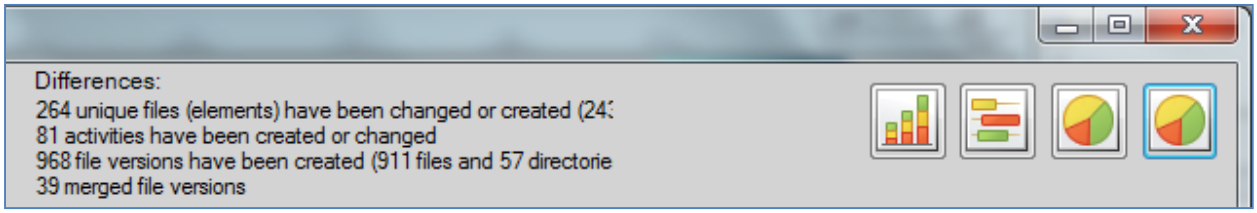
Technical requirements:

- ClearCase UCM

6) Lines of Code (LOC) per code language



7) Change metrics and code churn



Description: At the beginning of each report, we provide statistics:

- How many file elements have been changed or created
- How many activities have been changed or created
- How many file versions have been created (separated by files and folders)
- How many file versions have been created as a result of a merge operation
- All baselines created between
- All contributing users
- If Multiple Pending Change-sets report is launched, we also provide interim-summary (separated for each stream)

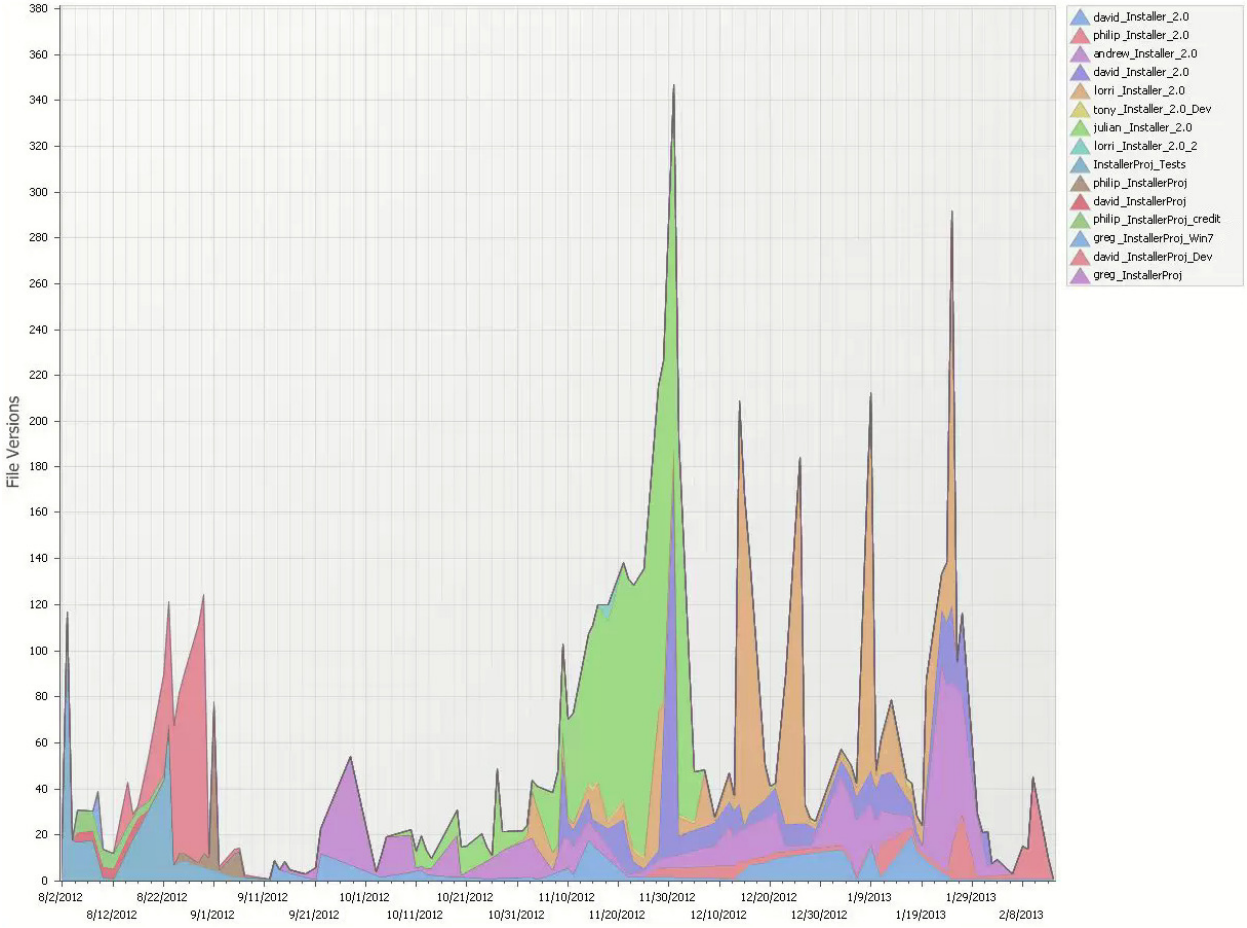
Based on data retrieved from the following reports:

- Two baselines comparison report (from the same stream or different streams)
- Two streams comparison (from the same project or inter-project)
- Multiple Pending Change-sets report
- Full stream report
- Chart is displayed after the data has been filtered!

Technical requirements:

- ClearCase UCM

8) **Check-in distribution over time by streams**



Description: This is area chart. It accumulates the amount of daily check-ins (each stream gets a different color).

The feedback we have received tell us that it's particularly useful when:

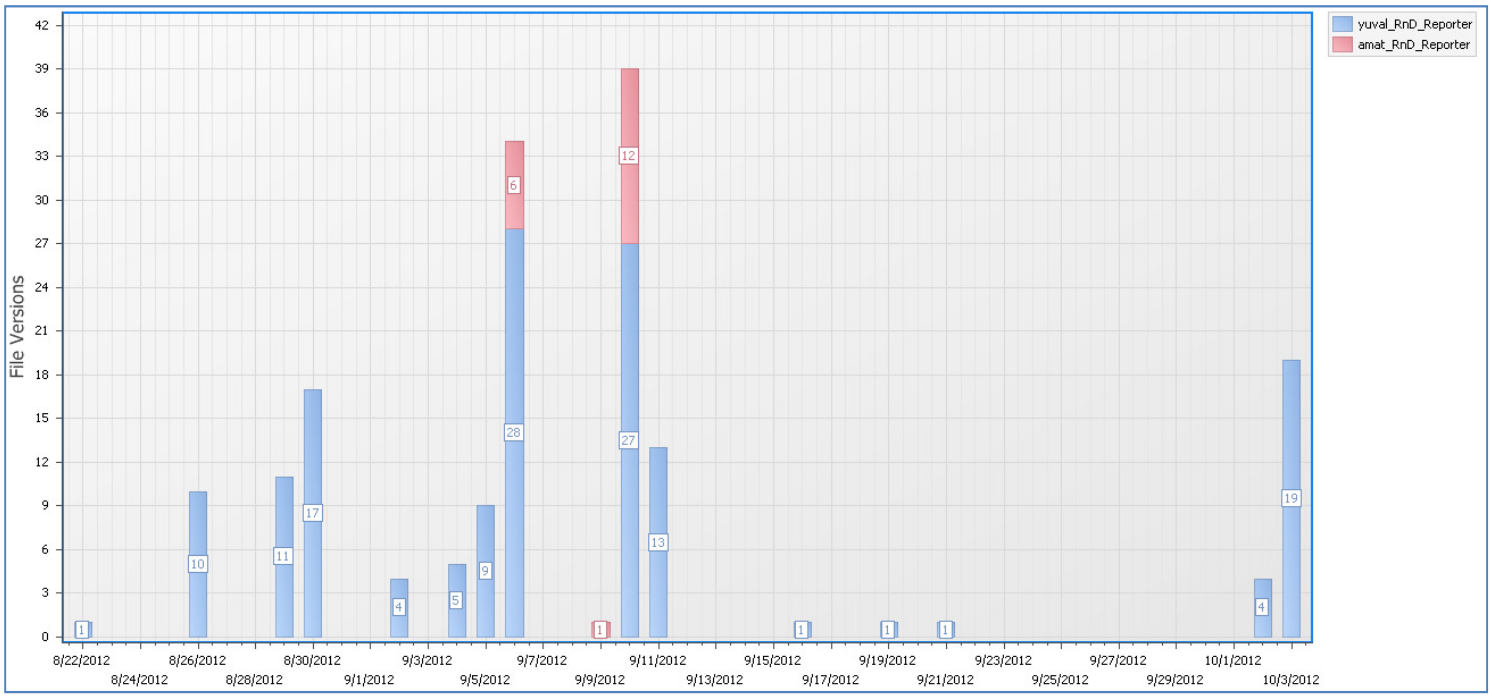
a) You have to quickly locate **busy** streams in order to split them out or share jobs among several developers. For instance, see the green area starting at 11/10/2012.
b) You have to quickly identify merge check-ins in main streams or integration stream, or even find out the frequency of merges (for Continuous Integration purposes). For instance, see the peaks of orange area that occur every 10 days on average.

Based on data retrieved from the following reports:

- Two baselines comparison report (from the same stream or different streams). You can exclude or include contributing activities (and change-sets).
- Full stream report

Technical requirements: ClearCase UCM

9) Check-in distribution over time by streams



Description: Like the previous chart, this bar chart displays an accumulation of daily check-ins per streams. (Each stream gets a different color).

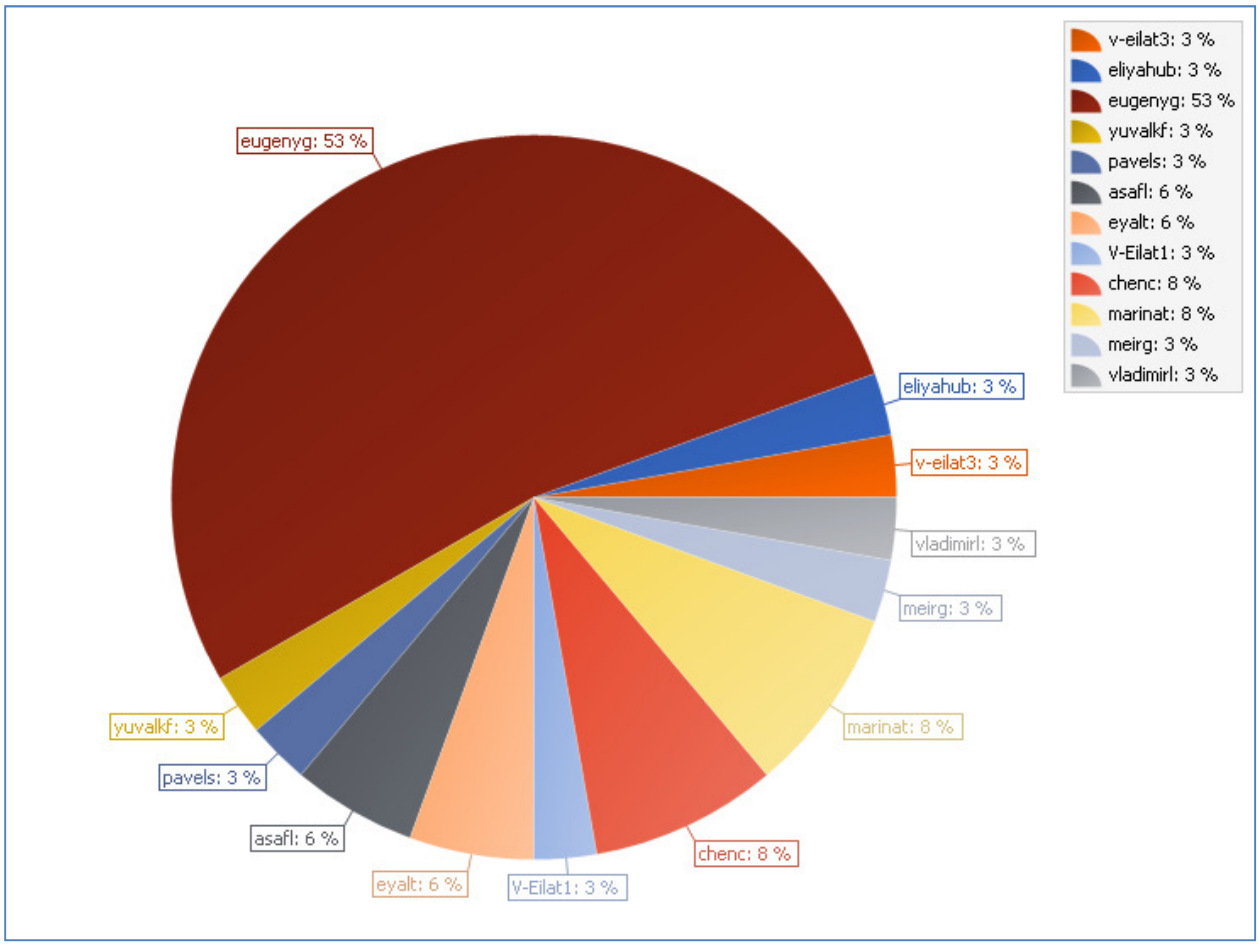
Based on data retrieved from the following reports:

- Two baselines comparison report (from the same stream or different streams)
- Two streams comparison (from the same project or inter-project)
- Full stream report
- Chart is displayed after the data has been filtered!

Technical requirements:

- ClearCase UCM

10) **Activities distribution by users**



Description: This Pie chart displays activities distribution by activity's ownership.

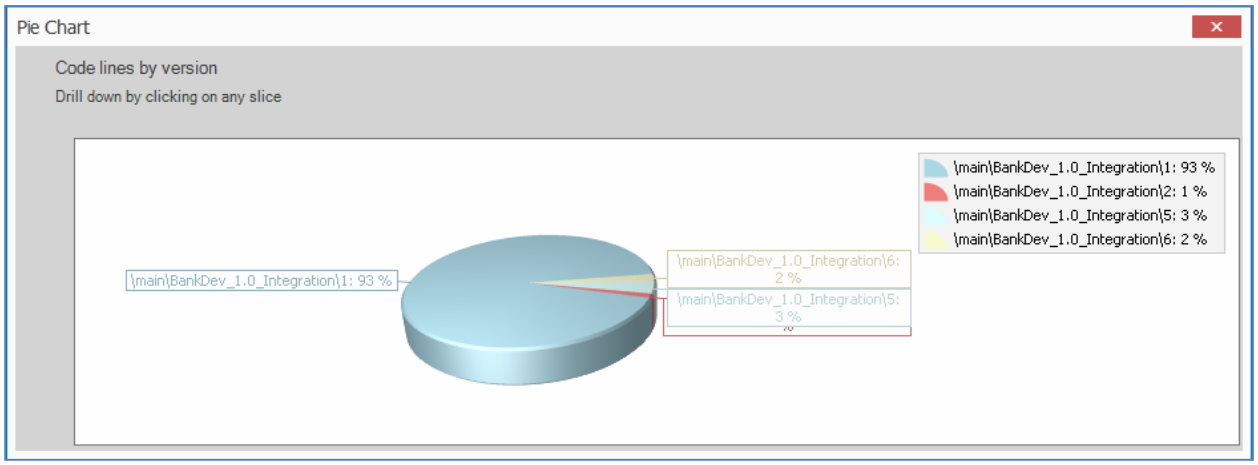
Based on data retrieved from the following reports:

- Two baselines comparison report (from the same stream or different streams)
- Two streams comparison (from the same project or inter-project)
- Multiple Pending Change-sets report
- Full stream report
- Chart is displayed after the data has been filtered!

Technical requirements:

- ClearCase UCM

11) Code lines distribution by file versions



Description: Using a given source file that is displayed by Visual Annotate, you can generate this pie chart, which shows code line distributions by file versions (in which file version each code line was inserted from a check-in operation).

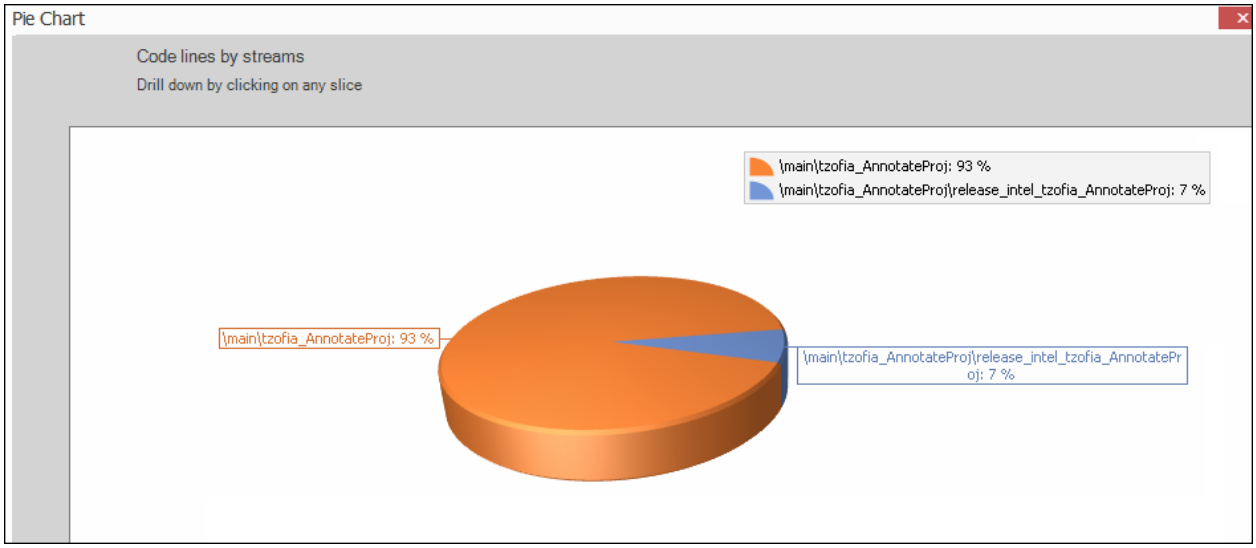
For instance: This screenshot shows that 93% of the code lines of this file were inserted through the \main\BandDev_1.0_Integration\1 version.

You can build up this chart for any source file, for each of its versions and for each of its branches (streams).

Technical requirements:

- ClearCase UCM or Base
- Visual Annotate 3.x

12) Code lines distribution by streams / branches



Description: Quite similar to the previous chart, this chart also shows code lines distribution – by UCM streams or branches. The chart accumulates all of the file versions that belong to the same stream\branch, and then displays each stream\branch separately. This chart helps find out the contribution of each stream\branch for a given source file.

For instance, this screenshot shows that 93% of the code lines were made on \main\tzofia_AnnotateProj stream.

Technical requirements:

- ClearCase UCM or Base
- Visual Annotate 3.x

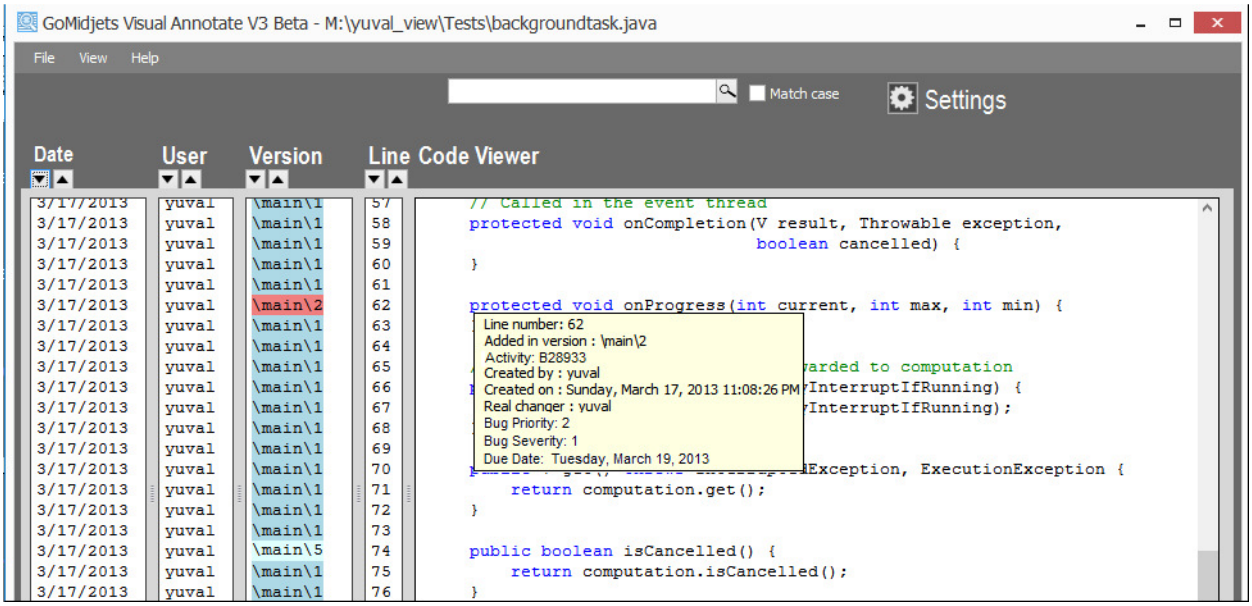
Just like the two previous charts, we also provide another two pie charts that display code lines distribution by users and UCM activities.

13) Code lines distribution by users
14) Code lines distribution by UCM activities

In future we plan to provide additional charts that will show where the code lines were actually changed, including the file versions, branches, streams, users and activities in which the code changed before it was merged to the file version that Visual Annotate's output is currently based on.

To get further details and test the beta version when it comes out, send an email to beta@almtoolbox.com

15) **Full traceability from a code-line to the associated bug\feature\issue**



Description: it's often essential to find the relations between a specific code line to the bug or feature in which it was made on. Locating it manually is time consuming and error prone. ***Visual Annotate*** enables you to display the information on a tooltip that is associated with each code line when the user hovers above with the mouse.

In the example here, you can see the output of running *Visual Annotate* on a Java file. The tooltip displays information about line #62: which version it was added from; on which activity, etc.

You can see that the last three lines on tooltip come from a bug-tracking system that associates UCM activity with the related bug. In this example you can see that line #62 is associated with a bug whose priority is 2; severity is 1 and due date is 3/19/2013

This is a strong feature that essentially saves developers time and helps them be more productive on a daily basis.

To get further details and test the beta version when it comes out, send an email to beta@almtoolbox.com

Technical requirements:

- ClearCase UCM or Base
- Visual Annotate 3.x
- Issue \bug tracking system that exports data into CSV file. E.g.: ClearQuest or RTC.

Users can also set CSV file manually (in which case an integrated tracking system is not mandatory).