D8.3: FastFix Formative Evaluation

Lead contractor: Prodevelop S.L.
Author(s): Javier Cano, Miguel Llácer, Marc Gil, Christophe Joubert, Sergio Zamarripa, Carmen Calle, Alessandra Bagnato
Submission date: June 2012
Dissemination level: Public

Project co-funded by the European Commission under the Seventh Framework Programme
© S2, TUM, LERO, INESC ID, TXT, PRODEVELOP
Abstract: This document describes the formative evaluation that has taken during the development of the FastFix project until the present date. The document describes the definition of the process, which decisions have been taken, how those decisions have been taken, how the process has been applied to the development of software artefacts and the improvements that the process has brought to the development of the platform.
Contents

1 Formative evaluation goals ......................................................................................... 6
  1.1 Nature of the project .................................................................................................. 6

2 Process description ....................................................................................................... 8
  2.1 Steps to create a formative evaluation process .......................................................... 8
    2.1.1 Analysis of the project ......................................................................................... 8
    2.1.2 Analysis of the development process ................................................................. 9
    2.1.3 Define metrics to collect .................................................................................... 9
    2.1.4 Tools to use ......................................................................................................... 9
    2.1.5 Testing plan ........................................................................................................ 10
    2.1.6 Monitoring ......................................................................................................... 10
    2.1.7 Reporting .......................................................................................................... 11
  2.2 FastFix iterative cycle ............................................................................................... 11
    2.2.1 Development stage ............................................................................................ 12
    2.2.2 Testing stage .................................................................................................... 13
    2.2.3 Evaluation stage ............................................................................................... 14
  2.3 Tools ....................................................................................................................... 14
    2.3.1 Subversion ........................................................................................................ 14
    2.3.2 Maven .............................................................................................................. 15
    2.3.3 Hudson ............................................................................................................ 15
    2.3.4 Sonar .............................................................................................................. 15
    2.3.5 Trac ................................................................................................................ 15
    2.3.6 StatSVN ........................................................................................................... 15

3 Process application ...................................................................................................... 16
  3.1 Tools at work ........................................................................................................... 16
    3.1.1 Build process ..................................................................................................... 16
    3.1.2 Quality of code ................................................................................................. 17
    3.1.3 Agile development ............................................................................................ 18
  3.2 Collected metrics ..................................................................................................... 18
    3.2.1 Source code ...................................................................................................... 18
    3.2.2 Build ................................................................................................................ 19
    3.2.3 Code quality ..................................................................................................... 20
    3.2.4 Issues .............................................................................................................. 21

4 Process results ............................................................................................................ 23
  4.1 Continuous integration ............................................................................................ 23
  4.2 Integration in target applications ............................................................................ 23
  4.3 Integration in organization processes ...................................................................... 24
  4.4 Performance requirements ....................................................................................... 24

5 Conclusion .................................................................................................................. 25
List of Figures

Figure 1 Steps to define a formative evaluation process ...........................................8
Figure 2 FastFix development and evaluation process ................................................12
Figure 3 Scrum agile development process ................................................................13
Figure 4 Integration testing .........................................................................................13
Figure 5 Evaluation stage .............................................................................................14
Figure 6 Tools hooked in development process .........................................................16
Figure 7 FastFix Hudson Server Dashboard ...............................................................17
Figure 8 Sonar violations .............................................................................................17
Figure 9 FastFix Sonar server dashboard ....................................................................17
Figure 10 Sonar violation detail ..................................................................................18
Figure 11 Lines of code progression ...........................................................................18
Figure 12 FastFix client build trend ............................................................................19
Figure 13 FastFix common build trend .......................................................................19
Figure 14 FastFix server build trend ..........................................................................20
Figure 15 FastFix common sonar report .....................................................................20
Figure 16 FastFix client sonar report ..........................................................................21
Figure 17 FastFix server sonar report .........................................................................21
Figure 18 Issues managed via Trac ............................................................................22
Introduction

This document describes the formative evaluation process that evaluates methods, concepts and prototypes as the project is being developed. This process allows for a continuous integration and enhancement of the platform. As a result, the development process is steadily monitored and evaluated, and early feedback is provided to the designers, developers and architects. The process has been defined based on the experience and feedback from two industrial end users, Txt e-solutions and Prodevelop. A set of open-source tools effectively supports the evaluation process. The process was successfully and seamlessly integrated on the FastFix agile software development lifecycle and the results obtained improved the overall FastFix design methodology and related platform.

This document presents in section 2 the formative evaluation goals by analyzing the nature of the project. Section 3 shows the creation of the formative evaluation process according to the FastFix development lifecycle and adopting a full open source and free software approach to tool selection. Section 4 describes the application of the process to the development cycle of FastFix. Section 5 details the results obtained from applying the processes during several iterations of the FastFix development lifecycle. Finally section 6 summarizes the results from the formative evaluation process.
1 Formative evaluation goals

One of the main challenges of software evolution is to provide software applications with a maintenance environment with maximum time efficiency at a minimum cost and the strongest accuracy. This includes managerial aspects of software evolution, like effort estimation, prediction models and software processes to give early feedback on the evolution of development metrics, global quantitative goals, non-functional evaluation, compliance with user requirements, and pilot application results. Our results were applied to a monitoring control platform for remote software maintenance.

The purpose of the document is to describe the formative evaluation process that evaluates methods, concepts and prototypes as the project is being developed. Formative evaluation aims both at ensuring that the goals of the under-development software are being achieved and at improvising the software, if necessary, by means of identification and subsequent remediation of problematic aspects. This process allows for a continuous integration and enhancement of the platform. As a result, the development process is steadily monitored and evaluated, and early feedback is provided to the designers, developers and architects.

As can be observed in existing development communities, there is a real need for a formative evaluation process that is not covered by current standard agile development processes. For instance, one of the most famous development communities is the Eclipse community. Several aspects of the Eclipse development process can be directly enhanced by our formative evaluation process. We can mention the strict unified timeline for integration, only guidelines to build components, no common continuous integration build system, development cycles of one year, no common consensus on metrics to gather, hardly no agile development achieved, outdated and hard to configure ticket management system.

The same applies to other communities like the TeamWeaver community, the Software Evolution community or the Polarion Software community where the development approach and the metrics used to evaluate the software already produced could be enhanced by this formative evaluation process.

The formative evaluation process has been defined based on the experience and feedback from two industrial end users, Txt e-solutions and Prodevelop. A set of open-source tools effectively supports the evaluation process. The process was successfully and seamlessly integrated on the FastFix agile software development lifecycle and the results obtained improved the overall FastFix design methodology and related tool set.

To successfully achieve the marked goals for the formative evaluation we need to consider and analyze several aspects of the project and more concretely of its development lifecycle.

Here follows a description of the evaluated aspects that have guided us to correctly select and apply a formative evaluation process for FastFix.

1.1 Nature of the project

The FastFix project is at the core a research project with an expected output resulting in a software platform prototype to perform remote software maintenance. As a research project among different partners from different countries involving academic and industrial partners the project has a particular nature that shares several common points with other research projects of international scope. Some of the most relevant aspects of the project are:
- Strong focus on research. Although the final goal is getting a working prototype the core of the project is performing the needed research and proof of concepts that allow confirming that the concepts proposed in the project can be achieved and are suitable for application in industrial organizations.

- Distributed teams. Several teams from different countries and in different locations are going to drive the development onward, with the communication and synchronization handicap that it entails.

- Development of new software artifacts. The central aspect for the formative evaluation process is the development of new software artifacts. As a result from the research step new software artifacts are to be developed that provide a reference implementation and a first prototype of the platform.

- Integration into common platform. All software artifacts development must share a common ground that allows the integration into a common platform. Components can vary in functionality and services provided but aim at being integrated seamlessly with other components to achieve a fully complete and integrated platform.

These aspects of the project will help better define and adapt the formative evaluation to the expected evolution of the process and the general working methodology that is mainly defined by these points.
2 Process description

Applying a formative evaluation process to a development project requires detailed analysis and planning of the project to pose the least risk to the normal progress of the project. The formative evaluation needs to be as least invasive ad possible and pose few or none obstacles to the normal development of the project. To correctly define a formative evaluation process a clear picture of the nature of the project and its most important aspects is essential.

Error! No se encuentra el origen de la referencia. shows a sample workflow to analyze the nature of a project, define the points to monitor and evaluate during the formative evaluation and specify the tools, metrics and responsible subjects to drive the evaluation process.

2.1 Steps to create a formative evaluation process

![Figure 1 Steps to define a formative evaluation process](image)

2.1.1 Analysis of the project

The first step deals with analyzing the natures of the project to be evaluated. There are several keys to extract in this area, producing an output that describes the key natures that the project has. The points to analyze to get the exact type of project include:

- Goal of the project, which can range from promoting collaboration and knowledge to producing a research prototype to creating a commercial product or expanding an existing product.
- Participants of the project, which can range from a small local team to a distributed team with many participants grouped as local teams in different locations.
- Timeframe of the project, small term projects have fewer applications than long term ones.
- Role of participants, from management to dissemination to development or evaluators.
Getting a correct identification of these different aspects of the project will help in identifying the metrics to collect in a later phase.

2.1.2 Analysis of the development process

For projects that include a development process, that is, a process aimed at developing software artifacts, there is a special interest in analyzing how the development is going to take place. The main result from analyzing the development process is to find points that give the best feedback when monitored.

The aspects to consider when analyzing the development process include:

- Linear or iterative project, iterative processes are more flexible when integrating feedback than linear processes and can benefit from a different set of metrics.
- Management of changes and issues, how they are managed defines if a given set of metrics can be helpful to the development process when given feedback from them.

2.1.3 Define metrics to collect

Taking into account the analysis of the project that has defined the basic aspects of the project and the development process this step deals with defining the metrics that can provide the best view of the project progress, deviations that may occur and feedback from which the development can benefit. There are several aspects that determine if a metric is good to be collected:

- Trivial to collect metrics should always be collected, barring special needs or restrictions. Many of the tools and systems that are used to manage projects nowadays already gather a set of metrics and statistics that would fit as trivial metrics. Examples as to when not to track these metrics can be privacy or security concerns or storage restrictions.
- Metrics that define progress of the project, these metrics need to be collected in order to be able to define the progress of the project in a quantitative way.
- Metrics that can provide feedback on the development of the project to the development process to make adjustments and take the optimal path to achieve the goals of the project.

When selecting the metrics to collect a crucial aspect to consider if the effort that is needed to collect each one of them. Metrics that require too much effort to collect can be more detrimental to the development of the project.

At the same time, those metrics that need to place too much intrusion to the development of the project will pose a risk to the correct development.

2.1.4 Tools to use

Having cleared the points of finding the project basic natures and the metrics that are to be collected it is time to define the set of tools to perform the task of metrics acquisition. The evaluation process needs to be as non-intrusive as possible. Several key points to be considered for the selection of the tools include:

- Ability to collect the defined metrics: it is mandatory that the selected tools have the ability to collect the metrics that we want to collect.
• Integration of the tools in the development workflow of the project, to prevent the intrusion of the tools in the normal workflow.
• Configurable alerts and notifications are a must to have an as much as possible automated evaluation process that requires the least effort diverted from the development of the project. Tools should be invisible until a deviation is detected and notifications to the involved parties are required.
• Familiarity of participants with existing tools. A last consideration to take is the familiarity of the participants of the project with the tools that are to collect the metrics. Familiarity with the tools usually brings more confidence to all participants when they are given feedback from collected metrics.

As a summary it is important to remember that the tools to collect the metrics need to be non-invasive, need to collect metrics that are worth collecting, need to be able to monitor and notify when deviations are detected and need to be transparent to participants to provide confidence in the metrics collected.

### 2.1.5 Testing plan

With a good definition of the metrics to collect and the tools to use it is necessary to consider the testing plan that will be used to collect some of the metrics of the project. It is expected that the testing plan will provide the most interesting and useful metrics for the project and a plan that integrated the tools into the testing plan is essential to deliver quality metrics.

Two non-exclusive aspects can be considered to adapt the existing plan for metrics collection:

• Find hooks in the testing plan where sensors can be introduced to collect the metrics of interest. These metrics will usually deal with performance or functional metrics that need to be collected from the execution tests that are performed during the runtime execution of the product.
• Wrap some of the steps of the resting to collect metrics around them. This wrapping can be applied when the execution of the steps produces some results that can be taken into account from a metrics collection point of view.

The collection of metrics during testing can be of most benefit to iterate development processes that have room to get feedback and improve over it. To get the most benefit of the iterative aspect of a process development it is crucial that the testing execution and the metrics collection are controllable and repeatable on demand to have the ability to compare different executions and measure progress.

### 2.1.6 Monitoring

After planning the metrics to collect, the tools to use and the fitting of the metrics collection in the testing plan there needs to be a plan for the monitoring of the collected metrics. The goal of monitoring is to watch for deviations in the project that pose a risk to the correct development.

The aspects to consider planning for the monitoring include:

• Define a responsible for monitoring: one of several of the parties of the project must be aware of the whole formative evaluation process and the monitoring tools.
• Automate some of the monitoring. It is typical of tools to provide automatic vigilance of the metrics they collect to detect trends that show a deviation from what is
standard for a correct progress. Use of these tools can automate some of the parts of the evaluation and can provide mechanism that have a fast notification mechanism when deviations happen.

The ultimate goal of placing a monitoring responsible is to look for deviations and take corrective actions when the metrics collected show one such deviation. In the case that a deviation is detected there are several actions to be taken to correct it, shown here from least aggressive to most aggressive:

- Review the metrics collected: false positives divert effort from the normal development of the project without need.
- Notify directly involved parties: most deviations are produced by an oversight that is easily corrected when it is pointed out to the responsible party.
- Notify all partners: some deviations raise problems that cannot be solved by its direct responsible or that needs changes that affect the project as a whole. In that case the attention of all the participants is needed for the deviation to be solved.
- Find alternatives: looking for an alternative way of achieving the same goal that the problematic part of the project reaches can solve the deviation.
- Review the viability of achieving that goal of the project: it is not uncommon that goals of the project need to be reviewed and relaxed to get a compromise between expected outcome and viable outcome of the project.

### 2.1.7 Reporting

For the definition of the reporting to be one as part of the formative evaluation process, there needs to be a clear consensus among all participants of the project. Aspects such as the points included in the reports, the target audience of the reports and their periodicity dictate how reports are to be created and in which level of detail. Some of the points can be detailed:

- The audience of the reports determines many of the contents to include in them.
- The periodicity of the report has a tight relation to the development process being iterative or not, allows creating incremental reports or forces for a unique and monolithic report at the end.

For any of the different ways in which evaluation reports can fit a formative evaluation process there are some common aspects that fit them all:

- Highlight good areas of progress. It is important to mark the areas of the project that are doing well. The formative evaluation is an excellent tool to find deviations in a project but is also a good tool to find the points where the project is doing well and that aspect provides also good feedback to all participants.
- Collect the areas to improve. Be it that they need improvement because deviations have been detected or because they show a trend that will lead to a deviation, the areas to improve can be detailed along with some helpful information such as causes that have led to the deviation and corrective actions that can be taken to correct or prevent the deviation.

### 2.2 FastFix iterative cycle

The FastFix formative evaluation proposes several distinct phases. It aims to be generic and flexible in nature to be adaptable to different existing industrial development processes. In particular, it is adaptable to the development process existing with the organization
frameworks of FastFix industrial end user case study providers and within FastFix development team.

The FastFix development and evaluation process can be split in three states. Figure 2 FastFix development and evaluation process shows the three stages of this process: development (in blue), integration testing (in green) and evaluation (in orange). Each stage is composed of several steps according to the tasks to be performed in each one of them.

As can be seen in Figure 2 FastFix development and evaluation process, the formative evaluation phase is strongly linked to the development stage of the targeted software system. Therefore, it tends to take place during the middle part of the development project. The aim of this phase is to provide orientation and refinement on the evolution of each of the solutions and on the global result.

![Figure 2 FastFix development and evaluation process](image)

**2.2.1 Development stage**

Following an agile development process, the process starts with incremental internal releases that are produced based on the requirements specified for the platform. In the case of FastFix platform, development uses Scrum, an agile development model. This allows the project to have a continuous evaluation strategy. The main evaluation tools during this phase are the control elements of the Scrum development approach and the metrics provided by the software tools of the platform development environment.

Scrum uses backlogs, which are, in short, a set of features with estimated time efforts. Big features are decomposed into smaller elements or tasks that may be done by one developer and for which the development effort is easy to estimate. Figure 3 Scrum agile development process describes the Scrum agile development process.
Software projects are developed in sprints, which are periods that cover usually 3 to 4 weeks. Each sprint starts with a sprint-planning meeting where the backlog for this sprint is defined. At the end of each sprint, a sprint review is carried out, which may incorporate the customer.

Using Scrum gives the manager the following benefits:

- Continuous reviews and short iteration periods ensure the quality and the fulfillment of success criteria;
- Frequent incorporation of feedback and appreciation of change makes the process very flexible; and
- Sprint planning and daily scrums make the project management effective and flexible for changes.

Tools and methods to monitor the development process will be described in Section 4. Only working qualified versions are issued and sent to the trial integration stage. The project roadmap aligns internal releases with integration tests to be executed after each milestone.

### 2.2.2 Testing stage

Following each internal release, there is an integration trial with an industrial application in order to provide early feedback. This phase tests the latest product batch of the target system produced by the previous development phase. Then, a complete plan of integration trials is set up. The definition of these trials specifies the metrics to acquire during the integration trial execution. As shown in Figure 4 Integration testing, the execution of the trials takes place on the target under as real world conditions as it is possible.

During this process, data and metrics are collected and aggregated. The aggregation and ordering of the collected data produces a feedback report with the metrics acquired during this trial case. This provides a measurement of the introduced global improvements on the latest product batch of the target system. Tools and methods to obtain metrics will be described in Section 4. They allow identification of symptoms of safety errors during execution, critical errors like performance degradation or changes in user behaviour.
2.2.3 Evaluation stage

The evaluation phase analyses the feedback produced by the integration trial phase. As shown in Figure 5 Evaluation stage, the metrics and data produced as feedback by the integration trial phase are analyzed by the continuous evaluation process.

![Figure 5 Evaluation stage](image)

From the analysis of the feedback the formative evaluation conclusions taken from the metrics data allows the improvement of both the development process and the integration trial process:

- Development process improvement. From the analysis of the feedback data three important items can be extracted:
  - Bugs: new bugs can be found or existing bugs can be closed as they have been closed.
  - Change requests: improvements on existing functionalities can be requested to make improve the target platform.
  - New requirements: As the target platform is used more intensively, new requirements that were not initially described can be found.

- Trial integration process improvement: From the analysis of the feedback data, new metrics, tools, or methods to acquire, can be suggested for the trial integration process next iteration.

To conclude, the data gathered during the trial after each internal release provides feedback for the development process on both issues and new requirements. Feedback strategies are defined and management tools for them are deployed, through issue management systems. As a result, the formative evaluation process allows a tight control of product families and software versioning.

2.3 Tools

In order to apply the designed formative evaluation process we need a set of tools that will automatically manage as many aspects of the process as possible. The FastFix project has focused on open-source tools to achieve this. We present here the set of tools that we have in place for the integration of the formative evaluation process on top of the development process.

2.3.1 Subversion

Subversion is the tool of choice for managing source code versioning. Licensed under the open source Apache License it is integrated into the development process as the base tool to
host the code because of the familiarity of the partners of the project and the maturity of the tool. Many other tools rest on top of it to provide their functionalities.

2.3.2 Maven

Apache Maven is released under an open source Apache License and is used as the glue that keeps the development and the tools together, serving as project configuration management tool. Maven is used to define the structure of the project, the build mechanism, the unitary testing mechanism, dependencies management and the grouping of projects in components. Specific addons to get better compatibility with OSGi had been included in the maven set-up as well as integration with Sonar analysis, JavaDoc html documentation generation and licensing from dependencies.

2.3.3 Hudson

Hudson is licensed under the open source Eclipse Public License Public and MIT license, is used as the continuous integration and testing server. It provides a platform that continuously grabs the latest code from the repository and performs a full build along with all the tests defined. The results from the build, either correct or failed, are notified to a list of interested parties. Hudson stores the results of the builds chronologically so that reports showing the evolution of different modules are available at any time.

2.3.4 Sonar

Sonar is released under the open source Gnu Lesser General Public License, is sued for static and dynamic code analysis to facilitate development and product increments stages, offering detailed reports about the quality status of the code. Hudson and Sonar are linked do that after each correct build in the Hudson sever there is a trigger that executes a Sonar analysis on the built code. Sonar stores the results of the analysis chronologically so that reports showing the evolution of different modules are available at any time.

2.3.5 Trac

Trac is released under the open source BSD License and is used as an issues management, tickets categorization, ticket lifecycle, notifications and planning system. To better support the FastFix testing phase addons have been added to support modifications via email or to add compatibility with several report generators to source code version control systems. It is integrated with subversion and has an internal wiki.

2.3.6 StatSVN

StatSVN is released under the open source Gnu General Public License and is used to create reports on repository activity. To monitor that the project advances and all parties contribute to their areas of work the reporting is created every night and details the overall activity from the start of the project to the current date. StatSVN is connected to the subversion repository but is connected to the Software area only.
3 Process application

Having established the tools to be hooked to the various phases of our formative evaluation methodology, we have applied it to the FastFix SCRUM based development case study code base.

As shown in Figure 6 Tools hooked in development process, each phase of the FastFix software development process had one of the selected open source tools hooked.

![Diagram of toolhook in development process](image)

**Figure 6 Tools hooked in development process**

3.1 Tools at work

3.1.1 Build process

Developers have had access to the Hudson continuous integration system scheduled builds, with a build of the whole platform scheduled and easily accessible. Also Hudson has been configured to provide notifications to the interested parties. As shown in Figure 7 FastFix Hudson Server Dashboard, Hudson was used to execute builds and to keep track of the history of builds, tests and changes. Specific add-ons to get better compatibility with Sonar have been added as well as specific add-ons to get better notifications.
Hudson configuration created notifications after each build depending on the results of the build. For correct builds only observer parties were notified of the build. For failed builds all parties that had contributed to the latest change-sets were notified so that they can review the changes and check their changes looking for errors.

### 3.1.2 Quality of code

Sonar code analysis system analyses code just after every correct build, developers and evaluators can have a very fast grasp on how the code quality evolves and on how the more critical components have to be improved. As it is shown in Figure 9 FastFix Sonar server dashboard, the red components are the most critical ones.

Sonar also offers detailed views of each one and every of the problems that it finds during the analysis, offering the exact location where the problem is located and hints on how to solve it. As shown in Figure 10 Sonar violation detail, violations that are found in the code are marked along with the candidate solution, hinting at the optimal solution as part of the analysis.
3.1.3 Agile development

TRAC issues management system had been used as a planning tool, to define milestones and their tasks, to manage responsible for tasks and internal releases. It was highly appreciated the Trac minimalistic approach to web-based management of software projects and its way to simplify effective tracking and handling of software issues, enhancements and overall progress. The organization capabilities of Trac allow to manage the tickets in different views that give different perspectives on the status of the project, from the planned items for the next sprint to the issues that have appeared during the testing phase.

3.2 Collected metrics

3.2.1 Source code

The number of lines of code that are part of the software of the project has been steadily increasing, shyly during the first year when the bulk of the research has taken place and more abruptly from M12 onwards. The graph taken from StatSVN shows the progression of the lines of code.

![Software: Lines of Code](image-url)

Figure 11 Lines of code progression
This is the expected development for the execution of the project as the research challenges are solved, the production of code escalates to produce the needed software artifacts that put in practice the concepts developed during the research.

### 3.2.2 Build

The build process is managed mainly by the Hudson server that rests on the build configuration specified in Maven project configuration files (pom.xml). The Hudson server is configured to that each night there is a build job for each of the components, in this case FastFix Common, FastFix Client and FastFix Server that grab the latest state of the source code and perform a clean build from scratch. The Hudson server is configured so that after each build notifications are sent to interested parties. If the build is correct, the notification is sent to observers of the build process. If the build fails, the notification is sent to all parties that have contributed to the latest change set that has affected the repository and that might be responsible for the problem that prevents a correct build from happening.

The latest trend in build stability is shown here:

![Common Build Trend](image1)

![Client Build Trend](image2)

**Figure 13** FastFix common build trend

**Figure 12** FastFix client build trend
3.2.3 Code quality

Sonar shows reports about the code quality of the analyzed source code by performing static and dynamic analyses. The three components of FastFix, common, client and server, have seen an overall improvement in the quality of the code that form them but they still show room for improvement. The summary of the quality report for each component is shown here.

![Figure 14 FastFix server build trend](image)

![Figure 15 FastFix common sonar report](image)
Sonar reports have been used to correct several defects and errors that were pointed out and to keep the number of blocker and critical elements to a minimum.

Sonar reports show here that there is some work still ahead to improve the quality of the produced code. Important points to tackle are the duplication of code that is present in the common and server components of FastFix and in the number of critical violations. These points will be addressed in the next months in the development effort before the end of the project.

### 3.2.4 Issues

The Trac ticketing system has been used form the very start of the project to trac tasks and error that have raised during the development of the FastFix project up to this date. The main task of Trac is in keeping a log of all the items that are still to be solved, a backlog for the agile development process. Up to this date there has been a noticeable number of items that have been treated through Trac.

Trac has served also as a tool to discuss some topics that needed the input and the expertise of several partners, being used sometimes in a similar way to a forum to exchange ideas, post opinions and solutions.

The planning component of Trac is also used to manage the active tickets and as a planning tool to define the scope of the backlog for the current sprint and for each milestone.
Figure 18 Issues managed via Trac shows the current status of the tickets in Trac for FastFix.

![Tickets management diagram]

Figure 18 Issues managed via Trac
4 Process results

From the application of the formative evaluation process and the monitoring of the progress and evolution of the development during the months that the project has been in execution the benefits that the formative evolution process has brought can be summarized in some actions that have benefited the development of FastFix.

The next sections describe how this process has benefited the development in a concrete way.

4.1 Continuous integration

One of the major points where the formative evaluation process has been very helpful is in the correct building of each component and in the correct integration of all components in a single platform.

The continuous integration system has constantly shown when and where errors occurred in the construction process. Many of those problems brought to light other problems of compatibility or integration nature that has made the team try a different approach that has usually lead to a more smooth build process and to a better integration.

As an example, the FastFix platform was originally conceived as a platform formed by a client and a server component. During the execution of the project there has been a separation of responsibilities that have provoked the creation of new components such as the common component, that holds the latest change for removal.

In further iterations the build process has been reviewed to include more components with a clearer responsibility that are separated from the core FastFix platform but that are part of the project such as sensors for different applications and operating systems, hooks to different applications or dependencies that are common to several components.

The continuous integration server has also served as an early warner of errors that have been introduced during development that halted the correct build. These errors were usually minor in nature, such as forgetting the commit of a new file, changing a dependency to an incorrect version or moving a file from one location to another without updating configuration files. Despite the minor nature of these errors, they can cause serious problems if they are not caught quickly to correct them before they are hidden by further modifications. Hudson has detected many of these errors and the fast response of the development team has corrected the errors when they are easily solved.

4.2 Integration in target applications

From M14 onwards the development of the platform has been tightly coupled with the integration in the target applications for the scenarios targeted as test cases for development.

Aiming at getting an early integration in target applications has steered the development of FastFix to adapt to the particularities of industrial applications, working with the restrictions in place and the performance overhead that is acceptable.

As an example of integration in target applications, FastFix sensors have been targeting the integration in MOSKitt (a desktop Eclipse RCP application), in Espigon (a multilayered JEE web application) and TXTExecute (a .NET multilayered web application).
Targeting the integration in these three industrial applications has driven the development to provide the necessary extensions and configuration capabilities in the FastFix platform to be able to adapt to different technologies, architectures and systems.

For instance, the sensors bridge mechanism has been developed to allow both direct RMI communication for Java applications or standard REST communication via exchange of Json messages.

### 4.3 Integration in organization processes

From M14 onward the development has also targeted the integration of the FastFix platform to the processes that industrial partners have already in place. Each organization has its own processes and the restrictions that can be found in each organization infrastructure and in clients and end users infrastructure is very heterogeneous and has plenty of quirks.

The integration with organization processes has pointed the development of FastFix to having an architecture that provides the general workflows and concepts along with a default implementation but that offers several extension points that allow providing an alternative implementation to achieve the same goals with alternate methods that are more suitable to each organization.

As an example, the bridge with the ticketing systems has the definition of the service declared as interfaces that are decoupled from the actual implementation. The default implementation of the ticketing bridge is for the Trac ticketing system. An alternative implementation exists for the Jira 3.1x ticketing system that Prodevelop uses and an implementation for the Eventum Web tracker is planned to comply with the requirements from Txt e-solutions.

### 4.4 Performance requirements

From the start of the project the performance impact that the platform can have on those systems in which it is integrated is being considered in each design of the architecture and in each implementation effort that is performed.

The architecture of the platform has been refined in several iterations to evolve from a rigid structure to a flexible structure based on services, with some them having the capability to be shut down when they are not needed or cannot be available or delaying their execution to batch processes when the computer has some idle resources available.

For instance, the self-healing and fault replication mechanisms need heavy computations to provide some of their functionality. The execution of these heavy tasks is planned to take place at times when the monitored system has idle resources and is under a light load so that the performance hit does not hinder the normal use of the application to end users.

Another example of configurable performance that has been introduced after iterating the development through the integration in industrial scenarios is the configuration of sensors. Each FastFix sensor is connected to a FastFix client instance to which it sends events. Under some circumstances the sensing of some values can pose a performance overhead to the monitored application that is not acceptable. To solve this issue the sensors now receive a configuration command from the FastFix client on each heartbeat that determines the level of sensing to be performed to reduce the performance impact or even to deactivate the sensor altogether.
5 Conclusion

This report presents the formative evaluation process that has and is taking place in the FastFix project. We have presented that the goals of the formative evaluation are to early evaluate the design, architecture and methods of the FastFix platform to detect issues early and correct them.

The first part of the document presents how a formative evaluation process is prepared, depending on the nature of the project, taking into account several aspects of the project and executing a series of steps that give as output a defined formative evaluation process.

Then the process in place for the FastFix project is detailed, with a focus on open-source tools, and also the hook points of the formative evaluation process are exposed with focus on being as unintrusive as possible. The Hudson and Sonar servers serve as excellent free software tools to apply on top of a project that is being developed using Subversion as source code repository and Maven as project configuration tool.

We have seen that the application of the formative evaluation process has brought a tangible benefit to the development of FastFix in aspects that include:

- Continuous integration: the Hudson server has been an invaluable tool to iterate over the building process and reach the conclusion that the first concept of a platform that was only separated in client and server components was not viable. After several iterations a better division of responsibilities has been established with a clearer build process.
- Code quality: the Sonar server has shown the weakest parts of the code regarding quality. Several aspects that are associated to bad quality code have been tackled by having frequent and easy reports from Sonar that detailed the hot spots of the code where problems are to be addressed.
- Integration test: early integration into industrial applications, environments and processes along with support from industrial partners to correct the points that did not match what was expected in industry has driven the development of FastFix to be a platform based on services that can have their implementation changed to adapt to different technologies, operating systems, external services, etc.
- Issue management: having a central tool where issues could be tracked, tasks could be defined and roadmaps could be planned has proven to be a valuable tool in keeping all the partners aware of the status of the development and the actions required by each one at each point in time.

It is worth noting the benefit introduced by an evaluation system that is provided by open-source and free-software tools that require a minimum level of intrusion in the normal development of the software and that keep themselves quiet until problems rise that require the attention of one party to solve. Even at the time of notifications the tools keep their notifications at a minimum, only communicating with the parties that can have direct responsibility, to keep the evaluation as unintrusive as possible.