



Agile Software Development of Embedded Systems

Version : 1.0
Date : 2005.10.28
Pages : 25

Authors

G. Sánchez
P. Soret
D. Villarroel

Status

Final

Confidentiality

Public

Agile Deliverable D.2.5

Agile Requirements Management

Abstract

This document describes an approach for management of the requirements engineering phase according to agile practices in small and medium sized software development companies oriented to embedded world.

The document is addressed to both the project managers and the software developers of the company, because it not only describes how the agile practices can be used to specify the requirements but it also addresses the advantages from the business and managerial point of view.



I T E A

INFORMATION TECHNOLOGY

FOR EUROPEAN ADVANCEMENT



TABLE OF CONTENTS

1. Introduction.....	3
2. IMPLEMENTATION OF AN AGILE RM PROCESS	4
2.1 AGILE METHODOLOGIES STRENGTHS AND WEAKNESSES.....	4
2.2 AGILE METHODOLOGIES AND REQUIREMENTS ENGINEERING	5
2.3 GOALS IN REQUIREMENTS MANAGEMENT	6
2.4 APPLICATION OF THE AGILE METHODS	8
3. IMPLEMENTATION OF THE AGILE REQUIREMENTS MANAGEMENT IN THE PROJECT DEVELOPMENT	13
3.1 PREVIOUS TASKS.....	13
3.2 ACTIVITY DEFINITION.....	15
3.3 ACTIVITY CONTROL	16
4. METRICS	17
4.1 METRICS JUSTIFICATION IN REQUIREMENT CAPTURE.....	17
4.2 METRIC IMPLEMENTATION	18
4.3 CONCLUSIONS.....	23




CHANGE LOG

Vers.	Date	Author	Description
0.1	2005.09.05	SQS	First Draft
1.0	2005.10.28	SQS	Update of metrics and english version

APPLICABLE DOCUMENT LIST

Ref.	Title, author, source, date, status	Identification

	Agile Requirements management Deliverable ID: D.2.5	Page : 3 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

1. INTRODUCTION

This document describes an approach for management of the requirements engineering phase according to agile practices in small and medium sized software development companies oriented to embedded world.

The document is addressed to both the project managers and the software developers of the company, because it not only describes how the agile practices can be used to specify the requirements but it also addresses the advantages from the business and managerial point of view.



2. IMPLEMENTATION OF AN AGILE RM PROCESS

2.1 AGILE METHODOLOGIES STRENGTHS AND WEAKNESSES


There's no universal methodology which can be translated literally into any software development project. Every methodology needs to be somehow adapted to the project context: technical and human resources, company's own development cycle, type of system.

One of the disadvantages of traditional methodologies is the huge effort, which is necessary to tailor them for the context of a Project, especially in the case of small organizations and changing requirements. Agile methodologies offer a solution easy adaptable for a great amount of projects with such characteristics. One of the most remarkable qualities in an agile methodology is its simplicity, both in its learning and in its deployment, which reduces the costs of implantation and facilitates its adoption in a development team. This has led to an increasing interest in the agile methodologies. Nevertheless, there are also a series of disadvantages and restrictions for his application, such as:

- they are thought for small medium sized groups (Beck suggests that the size lays between the range from 3 to 20, others say not more than 10 participants),
- The physical working environment should enable frequent communication and collaboration between all the members all the time - which includes from project leaders to development and testing teams, and also the customer.
- If any of these above mentioned parties are reluctant to accept the practices and principles, it can take to the failure of the process (the atmosphere of work, the collaboration and the contractual relation are the key),
- the use of technologies that can't provide a fast feedback or that are not suitable to support changes,
- And so on.

A body of decided knowledge respect to the theoretical and practical aspects of the use of agile methodologies, as well as a greater consolidation of the application results is still need.

Some of the current research lines in agile technologies are oriented towards aspects such as:

	<p align="center">Agile Requirements management Deliverable ID: D.2.5</p>	Page : 5 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

- metrics and specific evaluation of the process,
- specific tools to support agile practices,
- Human aspects and team work.


2.2 AGILE METHODOLOGIES AND REQUIREMENTS ENGINEERING

There is an increasing tendency to offer closed-budget projects to the customer. This business approach requires a fine-detailed planning in order to minimize the project risks (specially the delay in the development and the date of release and the over costs). It is because of this, that a increasing number of different agile methods are becoming more and more popular, and beginning to spread not only among web development companies where they had their origin but also among several sectors and industries, which used to be more traditional, because, in the end every software development company – either belonging to the web services or to the embedded systems world seeks to optimize their development processes. The reduction in time of the traditional lifecycle guarantees fully operative releases perfectly in addition to a faster error detection and identification improvement capacity. These small but fully functional releases consist of a reduced version of the application which should be implemented, corrected, and improved in a defined short period of time and in a way which is transparent to the users.

Another one of the most recurrent problems in the tailor-made developments is the little implication of the customers during the development of the project.

Deficiencies in requirements (for example, incomplete or ambiguous requirements) are one of the biggest causes of the failures in the software projects. From the study of hundreds of organizations, Capers Jones discovered that the Requirements Engineering is deficient in more than 75 percents of all the companies. In other words, to have the appropriate requirements can be only the most important one and difficult part of software project, and a must to take the project to a happy end.

Because this key importance of a good requirements specification, the purpose of the perspective described in this document is to focus on the requirements engineering phase and look for an approach which makes requirements gathering, their estimation and their validation easier.

	<p align="center">Agile Requirements management Deliverable ID: D.2.5</p>	Page : 6 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

2.3 GOALS IN REQUIREMENTS MANAGEMENT

Therefore, one of the primary targets of this agile requirements management approach is to achieve this validation of requirements of each release as rapidly as possible. The intention is on the one hand to have a detailed estimation of each component of the application and on the other hand to be sure that the system being developed corresponds to what was expected. The validation of the requirements of each release by the user (with the user acceptance tests) increases the guarantees of success of the project.

This application of the agile methodologies described here has as one of the main objectives the continuous interaction with the customer and end users. The requirements specification document should consist of clauses or requirements that can be easily understood by all the stakeholders so they are able to agree or contract on the basis of those requirements, both from a quantitative as well as a qualitative point of view. The ambiguity of the requirements, which is one of the biggest causes of the failures in the software projects, should be eliminated.

This approach for an agile requirements management is aimed to develop a software product in an agile, fast and flexible, but nevertheless disciplined process, which brings simple solutions.


The key principles of action:

- To shorten the development cycles
- To involve the customer into the development from the very beginning and until the end of each cycle.

The work techniques provided are aimed to diminish the impact that the continuous changes imply in an embedded software development project.

The advantages in the development pursued by shortening the development cycles and reinforcing the communication with the customers are:

- In a certain moment, to focus only in a defined (small) problem.
- To solve it in agreement, immediate way, and not to drag it throughout the project.

	<p style="text-align: center;">Agile Requirements management Deliverable ID: D.2.5</p>	Page : 7 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public


- To begin each new development cycle (iteration) on the basis of an intermediate and fully operative version, which has already been verified and accepted by the customer.

The process for requirements gathering and management described here is mainly based in the Extreme Programming methodology, since this is the most popular of all the agile methodologies and there are a good number of previous successful applications. Like the rest of the agile methods, the idea of closed stages in the development of software (like the pure waterfall model) is rejected in favour of an iterative development process, which divides the whole development of the product in series of small cycles.

Each of these cycles will be a release which allows the customer to verify how the project is evolving and whether it should continue the same way. These iterations of reduced size solve the disadvantage that at the end of the project the customer finds a system that doesn't meet his necessities and initial requirements. The customer can - after studying the intermediate results - modify the initial priorities; to include, modifying or eliminating functions of the system; to change the direction of the project or to even abandon it. If the customer decides to modify or to add some features, the changes in the requirements in the development are analysed and agreed, the scope of the project and the priorities are replanned again, and the developer will consider the changes, implementing them in a next iteration of the system.

Similarly to XP, four fundamental values are considered:

- **Communication.** It is essential to keep all stakeholders informed about what is happening, and the evolution of the development. This includes not only the different members from the development team, but also project leaders, business managers and of course the customer.
- **Simplicity.** Any design or architecture must be an item that everyone implied in the development should be able to understand and modify with no need to have special knowledge. All design documents should be friendly and easily understood, without investing a lot of time. For this purpose, user histories and UML notations are a good help for simplicity.
- **Feedback.** The specification of the system should be as dynamic as the system that is modelling. The specification of the system and the development of the project should

	<p style="text-align: center;">Agile Requirements management Deliverable ID: D.2.5</p>	Page : 8 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

be expressed in such a way so that they allow to manage the changes that appear during the development and make it possible to re-plan the activities to adjust the process according to the current status of the development and the estimations of time. In this point, the feedback of true data collected in previous iterations is of the greatest importance.

- **Determination.** A developer must be decisive to rejecting the code that doesn't work or which does not match the specification in the user story and to begin from scratch again if necessary. Also he must have a character and courage to be actively involved in the projects.

2.4 APPLICATION OF THE AGILE METHODS

The application of these agile processes should be a practical approach.

The next paragraphs describe how this agile requirements management has been adapted and deployed for a project in a project in a small company. So even the process description follows an iterative cycle of theory-deployment which serves for fine-tuning.

The first step was to analyze different agile methodologies and practices which cover the requirements engineering phase. It was decided to take XP as a basis, due to its simple and customer oriented way for requirements gathering, and also because of the fact that this is the most popular of all the agile methodologies and there are a good number of previous successful applications documented and a wide bibliography.

After having trained both the company managers and the developers in the XP practices, and performed a careful selection of the projects (and customers) to which they could be applied, it was decided to define a company own requirements process by applying and adapting the practices.

The **practices adopted** are:



User stories/planning game. The customer exposes his necessities relating histories. Then, the development team considers the work that will take to implement each history and the work that the team can produce by iteration (defining the length of iteration like three weeks). With this information, the team informs the customer of the calculated estimations and together with the customer they draw up a plan with him, which will be kept updated throughout the project. The customer is who decides which stories will be implemented first and in what order. After the process of planning of the versions, the customer and the development team know exactly which stories are going to be implemented in each version of the system and the dates of delivery of each version. The planning of the versions is used to create planning of the iterations for each individual iteration.

The process of planning of the iterations takes the planning of the versions to a more detailed stage. Just before each iteration, the programmers meet and decide what programming tasks must be made to implement all stories corresponding to the iteration. Each pair of programmers takes responsibility of certain tasks.

In addition, a series of functions or roles will be settled down, to represent on the part of the personnel, to be able to solve doubts and to previously verify the functionalities that appear the customer. These roles are: programmer, customer, tester, tracer and coach.

Customer on site. For the application of the agile practices it has been essential the active participation of a person of the customer. Special emphasis in this practice has been made since it is considered vital for the success of the project. For the application of the customer on site practice it was agreed that there is at least a representative of the customer company with which can be consulted and asked, and also reported to, so that he can follow the development of the project, as well as propose modifications. The customer must be in communication with the development team during the project. The participation of the customer is essential in because it is intended that the customer participates in the software tests and that all the modifications are taken care of as fast as possible. On the other hand, all the members of the team (programmers, testers of the code and instructors) work together, in a same space.



Simple design. This practice is being adopted like a repository of functions to reuse, but owned by each programmer. The objective is to optimize this reusability and the future composition by means of the integration of the components. The first stage will consider the creation of a standard naming convention for the requirements, and traceability throughout the design, coding and test and in later stages this simple design and other practices already detected. An application must only be written to fulfill the actual requirements: it doesn't have to think about possible future extensions or about what will happen.

Coding standards. It is already made in the present time, basically for nomenclature of variables and functions. It is tried to extend the policy of codification for the developments and fits the possibility of including a tool of verification of the standards.

Inside of a project team is demanded that all the programmers write code of similar way (same type of commentaries, same way to name the classes, the variables, the methods, etc.), so that the code always offers a similar aspect. Due to this, any new programmer will be able to understand immediately the work already done.

But the main application regarding requirements will be the introduction of a **naming standard** for the requirements, which will also enable traceability throughout the development (and test).

Collective code ownership. Releases are built on the continuous integration of code. To facilitate continuous integration, two other practices are the key: one is a common coding standard, so that every programmer can understand the code written by any other member of the development team, and the other practice is collective code ownership, which means that all the programmers interact with a common code. For that reason the software is maintained within a common repository, where every day the changes and the advances in the software made by each of the programmers who take part in the release are updated and uploaded.

Open workspace. This practice was already being applied. All the programmers work together in a common room (work space).



40 hours week. The programmers do not always have a total dedication to the project. The experience lived during the prototypes has demonstrated that these dedicate part of their time to solve questions, implantations and repairs. Indeed the standards of codification and common repository, try to resolve the problems that can be derived from these habits.

Acceptance tests. Tests are written before the own code. With the aid of the customer, the programmers define the tests that must pass the software. Soon, the programmers dedicate themselves to write code that passes those tests. The process of test of software is constant, and it only finishes when the project finishes.

Metaphor. In certain way this practice can be considered generic of all the projects. In both cases it is made in the first meetings with the customer. The creation of releases will suppose a significant improvement in this aspect, because the knowledge of the customer of the application will increase of gradual form to the delivery of these releases.

Small releases. This is one of the high-priority practices of the project. If it is successfully implemented, this already means a great improvement for the project, as well as a guarantee for the success of another tasks. The timing of the releases it as well as their size, that is, their contents and scope are not fixed beforehand, but they had been agreed of specific form with the customer upon each one of the prototypes. The releases of the project are furthermore divided in iterations or cycles. The version of the system for each cycle provides the code for a small set of functions (in each cycle or iteration only a few histories of the users are implemented). On the other hand, the project is also incremental: each new block of code is got up to the application as soon as it is prepared and thus it is possible to show the advances to the customer in the programmed monthly meetings.

Continuous Integration. This is considered other of the pillars of the project. Each release must assure the continue integration with all the functionality of the previous release. So that each release is absolutely operative in the customer. Software is constructed and integrated several times a day like part of the development process (each new code that has successfully passed the tests incorporates to the already existing code), so that all the programmers of the project development team know how the development is evolving and how much of the software is ready.



Time Estimation and Effort Estimation. These measurements are considered especially important to make future estimations on the basis of historical data of previous experiences, as well as to make a planification as accurate and as realistic as possible.

Pair Programming. This is one of the practices more hardly understandable and that greater rejection they cause in the companies. Nevertheless, it has been implemented in because it makes it possible to reduced errors, to obtain several solutions of a subject and to provide a fast learning.

The **no-undertaken or postponed practices** are:

Unit Testing. This practice has been postponed because it has been considered too complicated for the first stage, and it demands learning to make the tests, a tool to automate them and quite a great effort. Nevertheless, it could be tried to adopt it in a future.

Refactoring. At the moment this practice hasn't been considered. The tests are going to make of manual form and by the customer. This practice implies to have testing knowledge and it can not even be considered without the support of a suite of automated unit tests.



3. IMPLEMENTATION OF THE AGILE REQUIREMENTS MANAGEMENT IN THE PROJECT DEVELOPMENT

3.1 PREVIOUS TASKS

Theoretical basis for carrying out the project was acquired through an specific training on Agile Methodologies, consisting on the explanation of different agile methodologies, their way of implementation and benefits, with an special emphasis on XP (the methodology that has been more taken advantage of), until the particularity of Agile requirements management approach.

The course began with an explanation of the most common errors detected on development companies.

In the beginning of this course it was explained the detected errors most common in the development companies. How on the basis of these it is tried to evolve the development processes and how the "process" should be modelled. This part of the course caused an ample debate since the company has a wide experience in size developments and, consequently, in these errors. After an ample debate the difference was made between bad deliberate practices (those caused by ignorance, by the existing majority between these companies or by incorrect assumptions) and forced (those which, by particularisations of the client, supplier or Project, are the only way to make them).

The individual practice proposals were numerous but in all cases it was recognized that additional estimations would have to be made to include these practices. It is in this first prototype where the most important practices for the company and in a gradual manner and on the base of success/failure of these try to extend them.

It was agreed that the change from the traditional process of development to an agile methodology cannot be assumed by the company suddenly but in the form of practices being adopted in an incremental way.

After several meetings a workflow was prepared to be applied in a real project.

The preparation of workflow has entailed the following tasks:



- Familiarisation with the terms of the application and consequently with the Agile requirements management approach.
- Identification of the generic nomenclature of requirements to reuse it in the different future projects. Being this one the base to be able to extract the metrics.
- Throughout the project numerous practices of the agile methodologies have been reviewed with the idea to implant them. Finally those that fitted with the type of projects and the policy of development of this company have been chosen.

Referring to project reach, tasks to carry out are:

- Consolidate the theoretical base to apply the agile requirements management approach methodology: criteria to identify, to name and to assign requirements.
- Specify the requirements of the projects with the identified nomenclature
- Process of validation of requirements on the part of the client.
- Establish a consensus to divide the application in releases in mutual agreement with the client. This entails on the one hand, to plan what requirements are going to be implemented in each release, and on the part of the client the validation of the requirements corresponding to such release.
- Criteria of estimation of hours by means of removal of implicit tasks in requirements.

The final target of the project is to be able to calculate the established metrics and all those additional ones that are detected throughout it.

Those metrics will have a hierarchic structure. Metrics that they have a level in the tree different from lowest will be calculated like the sum of those on the inferior level.

The basis of nomenclature stipulated for the requirements is the following one:

Name_of_aplication__ Name_of_module__ Name_of_requisite_Version_number

On the basis of these 4 concepts all the metrics will be made. The variables that also take part for the calculation of such metrics are:

- Planned effort (hours)
- Actual effort (hours)




- Number of programming errors
- Number of concept errors
- Number of days
- Iterations
- Validated / non validated requirements
- Development time

3.2 ACTIVITY DEFINITION

The implementation requires of the intervention of two great user groups: client and supplier of a software development. Both groups are also beneficiaries. The supplier refers to all the people who specify, design, and develop and implant the software asked for by the client. The different profiles of these two great groups are the following ones: supplier (manager, head of project, programmer, tester, maintainer...), client (manager/head of purchases, all the people of the department/company that must specify the functionality of the future system).

The base process that includes all the others is:

- Identification of the nomenclature of requirements (manager / supplier head of project).
- Meeting of supplier and client to identify the functional and non-functional requirements (head of project/analyst of the supplier and specifications of requirements of the company/department of the client).
- Identification of the application modules and creation of these in the system (head of project/analyst of the supplier).
- Identification of different releases (head of project of the supplier and person designated by the client) and validation on the part of the client of the requirements concerning such release (client).
- Allocation of the release requirements to the different programmers/analysts (project head).
- Registry of the requirements (programmer/analyst).

	<p style="text-align: center;">Agile Requirements management Deliverable ID: D.2.5</p>	Page : 16 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

- Verification of nomenclatures and criteria according to the established policy (head of project of the supplier).
- Calculation of metrics (head of project of the supplier).
- Future estimation on the basis of existing metrics in the application (manager of the supplier).

Throughout the process is fundamental the management of the changes in the requirements. The changes in the requirements throughout the development process are not only represented by the existence of new requirements or modifications to existing requirements - which is reflected in the nomenclature of the requirements -, but also its impact is observed through other metrics which come from planning and estimations of time and/or effort.

3.3 ACTIVITY CONTROL

In the project there are two control points to be emphasized:

- Verification of release according to the established policy. Once identified the scope of the release it is needed to verify that the requirements have the correct nomenclature, a description client-understandable and re-usable in future projects. This control is the base for the correct calculation of the metrics.
- After the previous step all the data corresponding to the release are sent to the later validation on the part of the client. It's at this moment when the specific task of programming will have to begin and not before.

Another task that guarantees the control of the activity is the continuous contact. This communication channel has been settled down for any doubt or significant modification that must be made in the application at the request of those, which causes a new version of the requirements.



4. METRICS

Metrics for the requirements elicitation and management processes have become very popular lately due to the demand coming from high management people for having at their disposal the visibility of the current state and the progress and evolution of the requirement captures, so to say, a control panel of the requirements process. The control panel shows the values associated to each of the metrics parameters, or else the adjusted/weighted final value which would represent the level of efficiency in the requirements gathering. In this way, the state of the mentioned capture can be monitored and you can know not only that a particular moment is “good” or “bad” but, which is more important, you can measure the variations that can be produced and that will cause the capture level to increase or decrease, and be able to make estimations for future projects with similar requirements.

4.1 METRICS JUSTIFICATION IN REQUIREMENT CAPTURE


The need to define and implant metrics is something frequent in different areas of Information Systems, and the requirements gathering should undoubtedly be one of them, although traditionally it tended to be kept apart. Measurement of the requirements process should be done not because it is something which is becoming “fashionable”, but because there are clear reasons which justify it such as:

- **To know the state of requirement capture** in a short, simple way, avoiding “false expectations” of collecting those requirements and allowing a dynamic growing of capture systems according to the company’s real needs.

- **To anticipate needs** so that you can foresee the appropriate investments (both from the point of view of the project budget and from the perspective of the number of personnel resources which will be necessary to develop the project) and therefore guarantee, at least, the fulfilment of requirement capture objectives.

- To have tools which will allow those people in charge to write **detailed reports on the state of their systems** and to detect the variations that can be produced in the capture.

With the previous arguments, it may seem a simple task to justify the need to implant requirement capture metrics adapted to a particular environment. And it is indeed, except

	Agile Requirements management Deliverable ID: D.2.5	Page : 18 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

when we start to actually do it and we find problems coming from defining that type of metrics in a traditional way.

Each solution is unique due to the strong component of creativity which surrounds a particular solution. This fact requires adaptability and, for that reason, it can be very helpful to apply the principles of some agile methodologies.

There are different approaches to the definition of requirement capture metrics. The problem is that in most cases those approaches can be too complex and expensive to be implanted in small or medium-sized companies. Therefore, the responsible managers for the requirements engineering should be provided with solutions that will allow a quick introduction of requirement capture metric models, by simplifying the definition of metrics and presenting simple models to be implanted in order to obtain results in a quick way,

In a small or medium-size business the vision that requirement capture metrics present is an indisputable need. However, metrics must change in order to meet the company's new demands concerning requirement capture.

4.2 METRIC IMPLEMENTATION

In the application of requirement capture metrics two different processes are to be found:


- 1) Definition and development
- 2) Implementation

Definition of metrics program

If, as we have explained, requirement capture metrics is going to be used to assess the work of the requirement capture division and used as well to improve the method of the mentioned capture, they are not an end in themselves. However, they are to be implanted like any other project of the section of requirement capture.

We can identify two activities for the definition of metrics program:

- **Identification and definition** of the firm's current requirement capture program. That is to say, to know the initial situation referring to requirements capture processes, as well as the firm's assets.

	<p style="text-align: center;">Agile Requirements management Deliverable ID: D.2.5</p>	Page : 19 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

- **Development and selection of specific metrics** which will allow the implementation of a measurement system of the effectiveness and impact of the identified requirement capture controls

However, tackling a metrics definition project is not trivial. A traditional cycle of life in waterfall, where the customer's objectives are to be defined, given a solution, the how, and finally put into practice, is not the most appropriate way to develop this type of projects characterized by a great amount of uncertainty. The uncertainty which affects any requirement capture project is due to three factors:

The customer - He has different needs at different times. The active value of a company will vary and, with it the needs concerning requirement capture will also change.

Technology – technological advances can be considered from two points of view: on one hand, technological advances which are implemented in a project; and, on the other hand, advances related to techniques or systems that can offer requirement capture some support.


People – on one hand, people directly related with the specification, setting off and operation of the requirement capture metrics processes; and, on the other, people who are not related to this project, but accept the needs of the company in requirement capture.

Each solution is unique due to the strong component of creativity which surrounds a particular solution. This fact requires adaptability and, for that reason, it can be very helpful to apply the principles of some agile methodologies like XP and especially on the requirements management.

Besides, the periodic revisions and reconsiderations with all the team involved which the agile methodologies propose adapt to the needs of definition of requirement capture metrics, where metrics can stop being effective or interesting and fulfilling functions at a determined time, and consequently, only by redefining metrics it will be adjusted to fulfil new objectives.

According to this philosophy, the definition of a metrics program should be as simple as possible

Plan of implementation of metrics program

	<p style="text-align: center;">Agile Requirements management Deliverable ID: D.2.5</p>	Page : 20 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

In order to have an idea of our starting point, as we often we don't start from scratch , it is very interesting to do an analysis from bottom to top, identifying which measures are already carried out in the company. From those measures we must decide the metrics which should be generated, which is nothing but the comparison of the measures throughout time, relating their differences with concrete actions. The last step is to extrapolate the company's objectives which can be fulfilled with the metrics we already have.

Up to now, nothing has been done but to obtain an outlook of the present state and the possibilities that it gives us with hardly any effort. The next step would be to define new objectives from those results, to prioritize them and analyze them again from top to bottom, this time. This analysis will produce a series of measures to be carried out, the way in which the measurement should be done , who should take them , how they should be related in order to generate the metrics we are looking for, and, last but not least, which, how and when those metrics should be made public. We shouldn't forget that we have to give quantifiable results to evaluate other programs or requirement capture policies.

Implementation


Once we have got clearly defined the metrics to be implanted in the organization we must put them in black and white on a document (Plan of metrics program implementation), we must implant collecting mechanisms and data analysis.

This epigraph may form part of a continuous project of metrics definition, and it is about its setting off and operation/working. The results to be obtained must be useful to identify problems or deficiencies, making the introduction of corrective measure possible now.

That is to say, proposing the purchase of a specific tool which will allow requirement capture, or organizing a formation course for all the company's staff on requirement capture, as well as providing resources/funds for the course and its development. The metrics established already will give an idea of the effectiveness of this program and will allow continuous implementation.

We are presenting now the metrics selected to carry out pilot schemes in projects. They are undoubtedly the metrics that best fit in with our objectives, where recollection of different measures is easier and, besides, their results are the most reliable. The selected metrics are:

Incurred effort/estimated effort (hours) per requirement, module, priority, programmer, validated or non-validated requirement, per iteration.

	Agile Requirements management Deliverable ID: D.2.5	Page : 21 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

Incurred effort (hours) per requirement, module, priority, programmer, validated or non-validated requirement, per iteration.

Estimated effort (hours) per requirement, module, priority, programmer, validated or non-validated requirement, per iteration.

Number of programming errors per requirement, module, priority, programmer, validated or non-validated requirement, per iteration.

Number of concept errors per requirement, module, priority, programmer, validated or non-validated requirement, per iteration.

Number of new functionalities per iterations. Number of initial functionalities inside every module to develop and number of functionalities actually developed. This metric is related to requirement distribution in every release / Iteration

Development time: Estimated and actually spent.


The difference between what was scheduled and what was finally done will give an idea of the deviation produced at the requirement stage/moment, and it will be very useful, especially for further/future budgets and for the estimation of a development deadline.

This type of metrics will corroborate the importance that requirements have in software projects. And more specifically, their user's validation. One of the great problems of customized software development is that the user's requirements may be ambiguous, incomplete or not valid. Thanks to continuous integration we make sure that the project grows on solid foundations which are unlikely to change in a future time, previous to/before the end of the project. For that reason, the objective of the Project is to apply jointly continuous integration and other practises of agile methodologies which will allow to obtain incremental/increasing releases perfectly functional.

If this concept of continuous integration is applied from requirement specification we can guarantee the following stages of development which depend on it. We also considered the possibility of applying continuous integration to development by complete testing of each incremental release obtained, but unit testing. In order to solve this problem, continuous integration is going to be performed and it is going to be based on validation tests of a version compared with another.

All the metrics defined are based on the correct nomenclature of requirements:

Application_name__module_name__requirement_name__version_number

	Agile Requirements management Deliverable ID: D.2.5	Page : 22 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

For example, with the metrics Incurred effort/estimated effort we could calculate the deviation affecting this requirement in all the projects we have realised. We could plan on the basis of the real resultant effort of hours.

The above metrics have a hierarchical structure, the same which can be deduced from the nomenclature itself; that is to say, the. Incurred effort/estimated effort (hours) of an application will be the cumulative result of all the incurred efforts for each requirement of each module /unit of the application, divided by the estimated effort which will be the cumulative result of all estimated efforts for each requirement in each of the modules/units of the application.

These structures defined on the basis (requirement) of any type of customized development will allow us to calculate metrics either independently or jointly.

Additionally you can add as many variables as necessary so that we can obtain a more detailed analysis and, therefore, higher added value.

For those projects where requirement validation is impossible for any reason whatsoever, we can equally count the faults/mistakes and deviations occurred.


Which would serve to corroborate the importance of requirement validation through recounting concept mistakes of the mentioned requirements.

The larger number of data about requirements, modules/units, and applications we have, the better prepared we will be to plan new applications.

Therefore, the reason for choosing those metrics is based on:

- Having real knowledge of the estimated and incurred hours in tasks we have defined. The consequence of the analysis can entail improvement of the process by implanting correcting actions.
- Future estimations, more suitable to experiences which actually happened in the company. Therefore, we can obtain a more realistic estimation and better resource assignment. Identification of tasks which may be more or less profitable, and consequently, identification of projects.

It would be important to underline the power of the system through a correct specification of requirements which are characteristic/appropriate of the application and of the process itself

	Agile Requirements management Deliverable ID: D.2.5	Page : 23 of 25
		Version: 1.0 Date : 2005.10.28
		Status : Final Confid : Public

of the company which is to use it.

4.3 CONCLUSIONS

As for metrics, software measure is a challenging, time-consuming task. Small software development teams demand a smaller set of metrics, which may be handled more easily and which will provide constructive feedback on their own process of development.



Agile Requirements management
Deliverable ID: **D.2.5**

Page : 24 of 25

Version: 1.0
Date : 2005.10.28

Status : Final
Confid : Public