Implementing CMMI using a Combination of Agile Methods

Julio Ariel Hurtado Alegría [1] and María Cecilia Bastarrica[2]
[1] Departamento de Sistemas, Universidad del Cauca
Calle 5 #4-70, Popayán, Colombia
ahurtado@unicauca.edu.co
[2] Departamento de Ciencias de la Computación, Universidad de Chile
Blanco Encalada 2120 Santiago, Chile
cecilia@dcc.uchile.cl

Abstract

This paper explores the possibility for software companies of getting a CMMI certification of their processes by applying agile practices. For this purpose, starting with CMMI maturity level 2 generic goals and practices, we analyze the applicability of a series of agile methods, identifying their individual or combined contribution in the fulfillment of each process area. The main result of this research is the definition of a "fulfillment delta" required for a small or medium size company to reach CMMI level 2 using agile methods. We present an application case where a small company applied a combination of XP and Scrum for implementing the requirement management area. We compare the theoretical fulfillment delta with this company's results.

1 Introduction

Currently, the software industry represents an important economical activity for every country; it offers multiple possibilities for business and it promises to be a great opportunity for developing countries. In Latin American countries, software industry is generally immature, and companies face low productivity that threatens growth and increases the existing dependency with respect to developed countries. In spite of the comparative disadvantages, the Latin American software industry has grown lately, so the generation of strategies for developing the area would allow the countries to take advantage of the opportunity.

Software quality assurance through software process improvement and certification is one of the strategies software companies could engage with a two fold goal: the first one is to improve their international image, so that they can participate in a global market; the second one, the need to make their projects more efficient and effective administrative units.

One of the characteristics of Latin American software industry is that it is mainly formed by small and medium size companies. Several of these companies have chosen to apply agile methods in their software development processes basically due to the small initial investment required, and taking advantage of their personnel competitiveness. Nevertheless, provided that most software development standards and models have been created based on rigorous traditional processes and methodologies, it is necessary to define what new practices these companies need to accomplish in

order to be able to get a certification using agile practices. We call this set of new practices the "fulfillment delta".

The SIMEP-SW project is building Agile SPI (Agile Software Process Improvement), a framework intended to support process improvement for the Colombian software industry. Its main goal is to motivate small and medium size companies towards improving and certifying their development processes. The framework includes practical recommendations for process implementation, in order to facilitate their certification, as well as a tool for process definition. In this paper we have established the requirements for a company to achieve a CMMI certification using agile practices.

In the context of the SIMEP-SW project, a pilot experience was carried out in order to validate the theoretical results of this research. The requirement management process area of a small Colombian software development company –Unisoft– was implemented according to the guidelines suggested by a combination of XP and Scrum. The results were compared with the theoretical "fulfillment delta".

In section 2 a list of related work is presented. Section 3 describes the main background elements, including a description of the SIMEP-SW project, CMMI and some agile methods. In section 4, the possibility of implementing CMMI using agile methods is stated. The main theoretical results of this study are described in section 5. An application case is presented in section 6. Finally, section 7 shows the main conclusions and describes our future work.

2 Related Work

There have been other initiatives that related agile methods with certification models. One of the most relevant related work is that of migrating agile methods to standardized practices [18]. In this work, both the engineering perspective and that of the agile world have the same importance in software development. For this purpose, a framework intended to implement agile values and principles within a particular organization was developed. In this context, it is necessary to use consistently RUP as a development process and CMM as a reference model for certification. In this work, the approach instantiation suggests that the process capability can be certified at a defined level, including explicit process instantiation and configuration.

In [30] an experience is presented where a company undergoes a CMM level 2 and ISO9001 certification using XP and Scrum. In this experience both agile methods contributions are combined: XP was used for the technical processes and after a year, Scrum was introduced to support organizational and managerial issues. Success was achieved in a particular case: the company reached a certification in both quality standard models. This method was called Xp@Scrum. It was also applied for making more agile the development process in "Software Global", from Motorola Argentina, a company already certified as CMM level 5, after noting difficulties in adapting its complex process to projects that were either small or whose requirements were unstable or vaguely defined. With this work it was established that Xp@Scrum is not incompatible with higher levels of CMM either [19].

Mark Paulk evaluates XP from a CMM perspective [23], and concludes that XP includes very good engineering practices, even though caution must be exercised because some of them may be controversial or even contradictory. By evaluating XP from a CMM perspective, he expresses how both ideas can be combined in an adequate and synergic way with other managerial ideas and

practices. While XP provides a system programming perspective, CMM provides an organizational process improvement perspective. The idea is that companies can take advantage of each of them by adapting and adopting their practices.

In [20] a maturity model for XP is proposed so that an organization could adopt it in successive stages. All practices suggested by XP are difficult to implement in a single step and they need to be well implemented in order to achieve all XP real benefits.

Although all these work are valuable empirical evidence of process certification, they are all particular experiences, either because they report the case of a particular organization or they use a specific method. In our work, before implementing improvement processes through agile methods, we take the approach of evaluating how a set of agile methods may make it possible to achieve a CMMI certification using their practices. Besides, there is not much experience trying to reach CMMI with agile methods, only CMM.

3 Background

The SIMEP-SW project's goal is to provide the necessary tools to motivate Colombian software companies to improve their development processes in order to facilitate the positioning and competitiveness in national, regional and international markets. The project intends to create, apply and test an improvement system that integrates elements of internationally recognized quality, evaluation and improvement models, together with the characteristics of the Colombian industry, and that could eventually be replicated in other industries with similar characteristics. As part of the strategy, the project intends to establish the applicability of practices from the agile world for implementing the requirements of CMMI.

3.1 CMMI

There are currently several internationally recognized quality and improvement models and standards. Among them, CMMI and ISO can be remarked. The SEI has developed the CMMI process model [27], its evaluation method SCAMPI [28], and the associated improvement method IDEAL [9]. The International Standard Organization (ISO) has developed the ISO 15504 [11] process model that is based on the ISO/IEC 12207 norm [14] and the the first amendment [16], its evaluation method ISO 15504 (fourth part) [12], and its improvement associated method ISO 15504 (seventh part) [13]. It is also important to consider the family of ISO 9001:2000 norms [15]. The improvement and quality models that are more widely accepted by industry worldwide are ISO 9001:2000 and CMMI. The latter integrates quality models CMM and ISO/IEC 15504, also known as SPICE.

CMMI models [27] involve the concept of CMM, established by the Capability Maturity Model for software (SW-CMM) [24] in a new level that promises the continuous growth and expansion of the CMM concept to multiple disciplines or knowledge bodies such as SW-CMM, IPD-CMM [26], and SA-CMM [5], among others. CMMI models are tools for helping organizations improve their development, acquisition and maintenance processes and services. Companies can use a CMMI model to help establishing their improvement objectives, to improve processes themselves, and to provide guidelines to assure a stable, capable and mature process. Provided that there are multiple CMMI models available, a decision should be made about which one is the most appropriate for

the particular needs of the organization. Either a staged or continuous representation must be selected, as well as the body of knowledge to be included as part of the model. Each CMMI model has been designed to be used along with other CMMI models, making it easier for the organizations to improve different areas in a consistent manner.

The set of CMMI models contains and is produced from a framework that has the ability to generate multiple models and the associated training and evaluation material. These models show the contained bodies of knowledge, such as systems engineering or software engineering, in the required combinations (CMMI-SE/SW/IPDDS/SS or CMMI-SE/SW). Representations are ways of presenting CMMI components, i.e. the best practices it promotes. These representations can be either staged or continuous.

The CMMI staged representation organizes process areas in five maturity levels in order to support and guide process improvement. The grouping in process areas indicates which areas need to be implemented in order to reach certain maturity level. Maturity levels represent a path that illustrates the complete organization evolution along a process improvement work. For each process area, a list of specific goals and practices is defined starting from general goals and practices. The staged representation uses four common characteristics for organizing generic practices: high management compromise, abilities to be developed, implementation orientation, and implementation verification. The maturity level of a certain organization allows us to predict its performance in some given disciplines. Each maturity level establishes an important part of the organizational process, and prepares the organization to reach the following maturity level.

The continuous representation uses capacity levels to measure the process improvement reached. Capacity levels are applied to the organizational process for each process area. There are six capacity levels numbered from 0 to 5. The CMMI continuous representation also includes capacity profiles, objective level and equivalent level as organizational elements of the model components. It groups process areas according to similar categories and capacity levels designed for process improvement within each process area. Capacity profiles represent process improvement paths to illustrate the evolution of each process area improvement. The equivalent level is used for relating process areas capacity level with maturity levels in the staged representation.

3.2 The Agile Methods

Seemingly opposite to the standardized processes, there is currently another trend formed by the so called agile methodologies [6], that is motivated by a profound consciousness of the chronical software crisis, by the responsibility assigned to traditional methodologies as the cause of this crisis, and by the desire to propose solutions. The term “agile” applied to software industry was created in February 2001, after a meeting in Utah, USA, were the “Agile Alliance” was created, an organization devoted to promote agile software development concepts and to help organizations in the adoption of such concepts. The starting point was a document that summarizes the agile philosophy: the agile manifest [3], that includes a set of principles and values that support the philosophy. We here describe some of these methodologies.

Extreme programming (XP) is the method created by Kent Beck, Ron Jeffries and Ward Cunningham [2]. XP was created for small and medium size software development teams where requirements are vague, change rapidly or are very critical. XP was designed having in mind the problems with traditional development methodologies with respect to deadlines and client satisfaction. The

main goal of XP is to achieve the client satisfaction trying to maintain his/her trust and confidence in the product. XP recommendations are oriented towards getting high quality software.

Scrum has been applied by Jeff Sutherland and more formally elaborated by Ken Schwaber [25]. Soon afterward, Sutherland and Schwaber joined efforts for refining and extending Scrum by applying industrial control methods together with methodological experiences at Microsoft, Borland and Hewlett-Packard. Scrum is not conceived as an independent method, but a complement to other methods such as XP or the Unified Process (UP) [17]. As a method, Scrum stresses management values and practices, and it does not include practices for the technical parts (requirements, design, implementation); that is why it is advisable to use it in combination with another agile method. Scrum is a management and control process that implements process control techniques.

Agile modeling (AM) is a complement to agile methodologies to facilitate modeling and effective documentation of software based systems. AM is a set of practices guided by principles and values to be applied by software engineers. AM does not define a specific modeling procedure in full detail, it only provides advice to make the modeling more effective. AM can be applied in requirements, architecture and design modeling [1]. AM is a modeling strategy intended to take care of the fact that agile methods do not care about modeling or documentation as work products.

The Evo method, created by Tom Gilb, is the oldest agile method [8]. In 1976 Gilb dealt with topics such as iterative development and evolutive management; later he treated these topics more deeply and he published "Evolutionary Development" in 1981 [7]. In the 90's, Gilb continued developing Evo, which influenced other methods such as XP, Scrum and even UP [17]. This model is made of five main elements: goals, values and costs, solutions, impact estimation, evolutive plan and functions.

Crystal is a family of methodologies created by Alistair Cockburn [4]. These methodologies are based on the fact that, comparing software construction with another engineering process makes us think about software "specifications" and "models", about its completeness, correctness and operation. Cockburn thinks this questions are not contributive because after a while they become obsolete and irrelevant. There are four variants of the Crystal methodologies: Crystal Clear for up to 8 people teams, Yellow for teams from 8 to 20 people, Orange for 20 to 50, and Red for teams of 50 to 100 people. It is promised to continue with Brown, Blue and Violet. The most exhaustively documented is Crystal Clear (CC). CC can be used in small projects with medium criticism, though it can also be applied to critical projects with some extensions.

Feature Driven Development (FDD) is an agile, iterative and adaptive method. Different from other agile methods, it does not cover the complete software life cycle, but only the design and implementation phases. It is considered adequate for major mission critical projects [22]. FDD applies an iterative development with the best found practices to be effective within industry parameters. It stresses quality aspects and it includes small tangible deliverables, together with the precise control of the project progress.

Adaptive Software Development (ASD) focuses on the problem of developing large and complex systems. The method uses an incremental and iterative approach, with continuous prototyping. It intends to achieve "the strictly necessary rigor", and for this purpose it proposes to execute the least required control [10].

Dynamic System Development Method (DSDM) is a framework for rapid application development (RAD) that is popular in the UK [29], and that has been promoted as a *de facto* standard for

developing business solutions with strict time frames. DSDM can complement other methodologies such as XP, UP, or a combination of them.

4 CMMI Practice Areas and Agile Methods

The SIMEP-SW project intends to establish the possibility of reaching a CMMI certification by using agile methods, and the way this could be achieved. With this purpose, we took the conceptual approach between both domains: CMMI and agile methods.

The conceptual schemas in Figures 1 and 2 show the requirements a process must fulfill in order to achieve a certification. CMMI structure is determined by its components, both for the staged and the continuous representations. These components are: process areas, specific goals, specific practices, generic practices, typical work products, subpractices, notes, discipline amplifications, generic practice elaborations, and references.

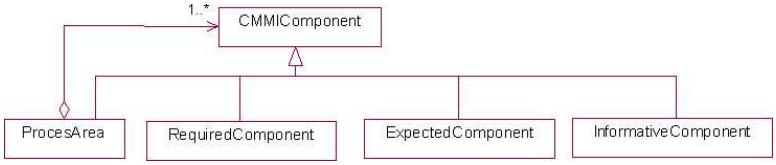

Figure 1: Types and relationships of CMMI components (abstraction of the CMMI model)

The main model component is the *process area*; it consists of a group of related practices that when implemented together they satisfy a set of goals relevant for significantly improving the area. Figure 2 shows component types using the elements defined in Figure 1 as stereotypes, and it graphically describes the relationships among these components within the model. CMMI components defined in a process area are: required component, expected component, and informative component.

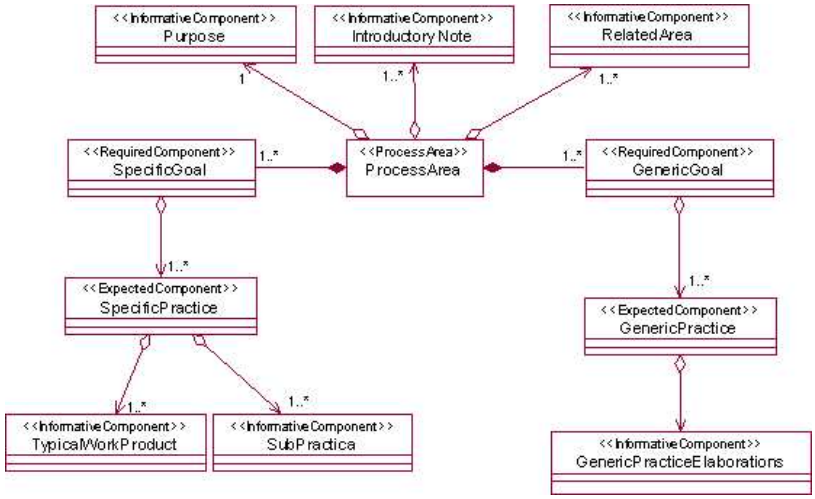

Figure 2: CMMI components

Required component. Describes the requirements the organization must achieve in order to satisfy the process area. This achievement should be visibly implemented in the organizational process. Required components in CMMI are general and specific goals. Satisfaction of these goals is used during evaluations to check if the process area has reached them.

Expected component. A set of expected components describes what an organization must implement in order to achieve a required component. These components guide people in charge of evaluating or improving a process. They include general and specific practices. Before goals are considered satisfied these practices or alternative practices implementation must be described in the process plan and applications.

Informative component. Provides details that help organizations to think about the form of reaching required and expected components. Some examples of informative components are: sub-practices, typical work products, discipline amplifications, generic practice elaborations, goal and practice titles and notes, and references.

It was necessary to develop a conceptual model in order to specify agile process elements based on the common elements in all methods. Figure 3 shows the preliminary conceptual model of the catalog of process elements we are developing, using SPEM for the stereotyping [21]. We have used SPEM because it is the language used for process modeling in Agile SPI.

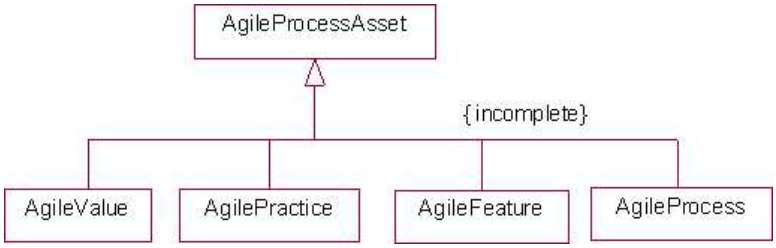

Figure 3: Preliminary conceptual model for agile process assets

Different methods define values, principles, processes and other characteristics. *AgileValue* refers to values and principles that define the method's practical orientation (e.g. continuous communication), *AgilePractice* is an agile practice recommended by some method (e.g. pair programming), *AgileProcess* is the agile process followed by an agile method (e.g. iterative or incremental process in XP), *AgileFeature* is any other characteristic that cannot be expressed in terms of the other elements (e.g. the main element Evolutive Plan in Evo).

In order to establish how far agile methods could contribute to implement CMMI, we took two different approaches, a general and a specific approach about how agile process assets can fulfill individually of conjointly CMMI requirements. Figure 4 shows the relationship between these two models and the two approaches. The general approach, represented by the circle in the right, started from the general description of the process areas, their general goals and their generic practices in the context of the respective process area. By doing this, we obtained an initial idea about how agile assets could allow an organization to reach a CMMI certification, which level of capacity or maturity, and which requirements these agile assets do not cover. The second approach,

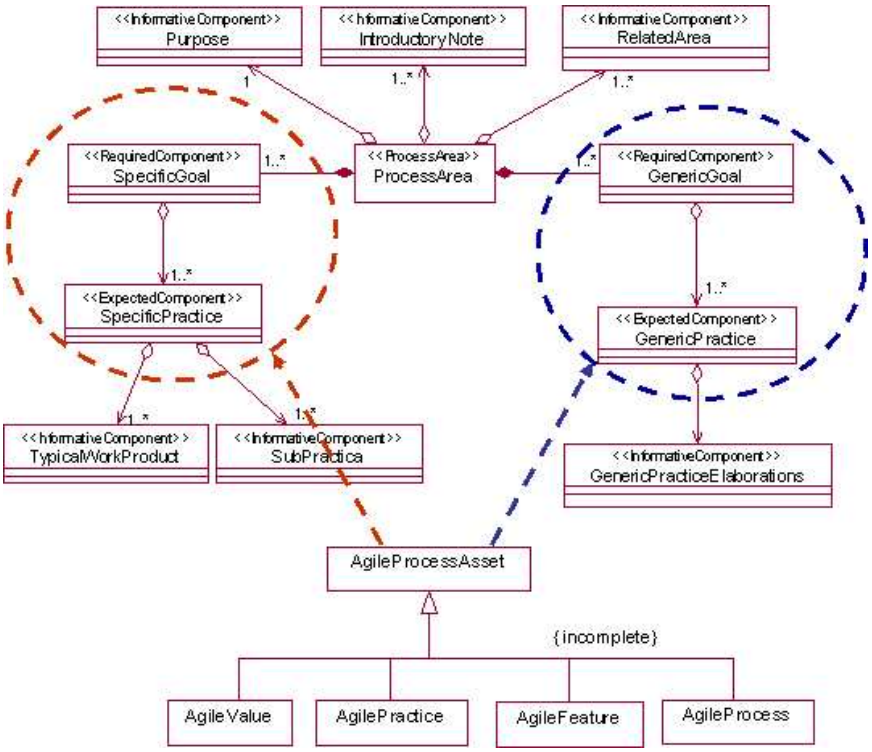

Figure 4: Implementation relationship between CMMI requirements and agile process assets

represented by the circle in the left, was considered in order to get more detailed information about the fulfillment of specific goals and practices.

In this way we limited more precisely the work that was necessary in each area to finally reach the set of process assets that jointly, and under the agile world philosophy, would make it possible for a small or medium size company to implement CMMI applying an agile process. This second approach was applied in the areas where we found more needs in the first approach.

In this work we show the general results corresponding to the process areas in profile 2 of the continuous representation, in order to evaluate the distance in the specific CMMI area of Requirement Management. We start from the general goals measuring each area's capacity level either reached or to be reached with a percentage between 0% and 100%. In the same way we listed the requirements needed to reach capacity levels 2 and 3 in process areas corresponding to maturity level 2, and we called it "fulfillment delta". Capacity levels 4 and 5 are not relevant because they are not relevant to achieve a CMMI certification in any representation.

We considered level 2 for this first approach, and we expect to develop the "fulfillment delta" for level 3 soon, provided that our general goal is to define a set of process assets and make them available as an infrastructure to implement process areas related to process management.

For each process area related to maturity level 2 we completed a table like the one shown in Table 1. It was then used to measure the capacity of each process area by mapping the contribution of each agile method to the area, and we deduced a capacity level that could be reached, the percentage of fulfillment that could be reached, and the work that should be done to complement

what is still lacking. If the combination of the elements in the agile methods can completely implement the area, we assume that a managed process can be established within the organization as a basis for this area.

Area	Requirement management process area. ML-2
Purpose	Manage product requirements and product components, and identify inconsistencies between requirements, project plans and work products

Values (V), principles (Pr), practices (P), artifacts (Art), roles (R) that favor the area implementation	**Method**	**Agile Method contribution to the process area**
V1. Continuous communication, V2. Feedback, Pr2. Customer participation, Art. User histories, R2. Customer, R4. Tracker, R5. Trainer, P5. Testing, P9. Continuous integration	XP	It provides basic elements to manage requirement, but it does not provide management elements *per se*. The customer is a main actor in the project management. It considers the existence of a process trainer.
V3. Daily meetings, Art. Backlog, R2. Product owner, P1. Product requirement, P5. Sprint requirement, P5 Daily scrum meeting	Scrum	Provides two essential elements to achieve requirement management: Backlog, that allows to trace a requirement, and a role responsible for tracing it, the product owner. The daily meeting promotes continuous improvement.
Pr3. Evo steps provide requirements specified in an evolutive manner	Evo	Considers requirements evolution
Art4. Delivery sequence, Art10. Requirement file, R2. Expert user, R5. Information radiator, P6. Process prototype, P10. Revision, P15. User point of view.	CC	Provides two important management artifacts: the requirement file and the information radiators. User points of view are also relevant. Provides training on the general process.
F2.Construction of the feature list, R6. Domain expert, R7. Delivery manager, P8. Progress report	FDD	Provides a feature list and a project progress report, both related to requirements. There are two roles that may play an important part in requirement management: the domain expert and the delivery manager.
Pr5. Change tolerance, R5. Customer representative	ASD	A customer representative is available to play the role of requirement manager and to promote change tolerance
F2. Business study, R3. User representative, R4. Visionary, P1. User compromise, P6. All changes are reversible, P7. High level requirement baseline, P4. Delivery acceptance subject to adequacy to business purposes	DSDM	It is favored by having the user as part of the team and with a high compromise. It promotes change reversibility and it recommends a high level requirement baseline to ease evolution.

Conclusion

The Requirement Management area is viable. Although none of the agile methods defines a complete process for requirement management, it is possible to build one integrating elements present in different methods. Using a base artifact (Requirement list, requirement, user history, feature) it is possible that a small organization can define a model for requirement management
The reachable capacity level is CL-2 in 75% and CL-3 in 50%
Fulfillment delta:
GP 2.8 Project monitor and control. There are no explicit elements for monitoring and controlling requirement management.
GP 3.1 Establish a defined process. It is only possible to define it within the organization that applies it.
GP 3.2 Collect improvement information. Although several methods promote communication strategies in order to improve project work, none of them promotes concrete elements to use this information to improve requirement management.

Table 1: General contribution of agile methods in the fulfillment of the requirement management process area

5 Results

By applying the same approach to other areas, we got the information in Table 2. Here we summarize the results of the first approach for each process area in maturity level 2, excepting subcontract

management because it is usually not applicable to small organizations. So, if we assign an equivalent weight to all process areas, we can calculate the general percentage required to reach maturity level 2. We also calculated the general percentage to reach maturity level 3 and included it at the end of Table 2.

Area	**Methods with major contribution**	**Fulfillment percentage (Capacity level + Percentage)**
Requirement management	XP	CL-2 in 75%, CL-3 in 50%
Measurement and analysis	XP, ASD, Scrum	CL-2 in 75%, CL-3 in 50%
Project planning	XP, Scrum	CL-3 in 100%
Monitoring and control	XP, Scrum	CL-3 in 100%
Subcontract management	Not applicable	-
Product and process quality assurance	FDD, Crystal, Evo	CL-2 in 75%, CL-3 in 50%
Configuration management	none	CL-1 in 10%
Fulfillment percentage for maturity level 2		72%
Fulfillment percentage for maturity level 3		60%

Table 2: Summary contribution value of agile methods in the fulfillment of process areas in maturity level 2

CMMI Requirement Management area can be implemented using agile practices. Although none of the analyzed agile methods defines a complete requirement management process, it is still possible to build one using a combination of elements found in diverse methods. Using a base artifact (e.g. requirement list, user history, or feature), the organization may define a requirement management model based on management practices in Scrum, or any other method it might consider adequate.

The Measurement and Analysis process area is not defined by any particular method; however, it is possible to implement it as a combination of elements of different methods. Two interesting strategies to consider are ASD time boxes and project speed measurement defined in XP. Any of these strategies can support quality and productivity measurement. Scrum's practices can also be of great value.

The Project Planning area is covered by almost all agile methods; the difference is that planning is applied to short iterations and it is not a required document. Planning is the basis for work, but plans may change; the goal is to develop a product that satisfies the customer. Plans are adjusted along the project, risk is taken into account by prioritizing activities, and participants are involved in these activities.

Project Monitoring and Control is exercised as part of agile methods but not in a very organized way, with the exception of some as XP where there is something more formal and Scrum where management is the most relevant strength. It is thus possible to implement this area with the contribution of these methods.

Product and Process Quality Assurance is a weak area in all agile methods because it is not very clearly presented. FDD includes a basic structure that goes beyond testing. Crystal Clear offers an alternative to synchronize the process in time, and Evo has an approach where the product and the process are improved as an inherent part of the development project. Provided that Scrum is the strongest in management, its practices can be applied in a quality assurance process.

Configuration Management is the weakest area because it does not even reach capacity level 1 using techniques from agile methods. Some methods such as XP or Crystal, propose a technical environment mainly focused on code configuration management. It is possible to extend these

techniques to manage other configuration items and to use tools to support some of the routine tasks. AM favors the usage of intermediate artifacts that may be considered as configuration items for the process as well.

The fulfillment delta necessary to reach CMMI maturity level 2 is given by the practices and other agile elements to support configuration management of items other than code, and the inexistence of practices or other agile elements to support monitor and control, quality assurance, requirement management, and measurement and analysis.

The fulfillment delta necessary to reach CMMI maturity level 3 must be completed with those other process areas of level 3. These areas are the establishment of a defined process, and the collection of improvement information in the areas of requirement management, measurement and analysis, quality assurance and configuration management. A summary of the achievable levels of capacity is shown in Figure 5.

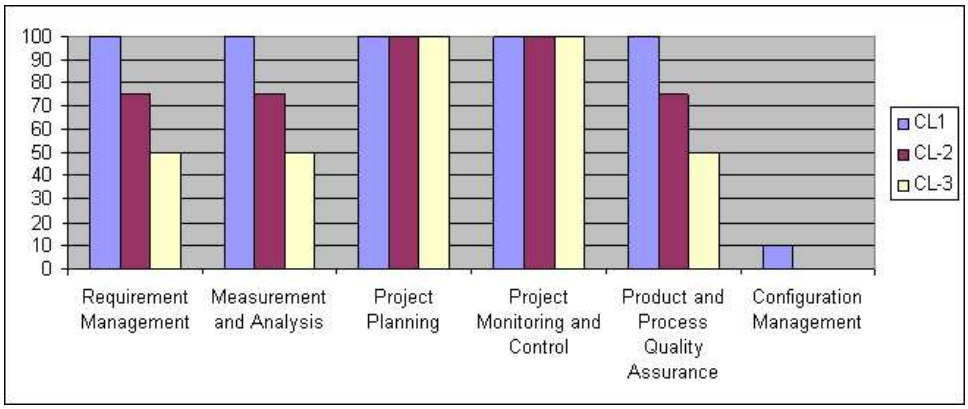

Figure 5: Capacity level achieved in each practice area

6 Case Study: Requirement Management with XP + Scrum

Using the knowledge gained in this research, the following step was to apply it in a real world case in order to corroborate the consistency of the found fulfillment delta.

Unisoft Ltda. is a small company formed by 9 people. Its processes do not satisfy any of CMMI process areas. Its first process improvement project is chosen to be in the area of requirement management because it is considered to be one of the most critical for the organization, it has one of the highest priority for its management, and it was initially only fulfilled in a 32%.

The process applied was as follows:

1. the requirement management process was designed in Unisoft as a discipline that would allow them to reach CMMI requirements using agile methods based on the theoretical findings explained in Section 5. Only practices in XP and Scrum were applied;

2. a mini-assessment was done, corresponding to the requirement management process area and according to the CMMI model. The particular fulfillment delta was empirically determined;

3. the measured fulfillment delta for requirement management was analyzed and compared with the one theoretically defined.

The company did not have a defined way of neither engineering nor managing requirements. Requirement management process in Unisoft was defined basically using XP, with the contribution of some practices of Scrum, that is stronger in management issues. The agile process assets theoretically identified were considered: the life cycle defined by Scrum was used for requirements as well as the requirement elicitation list; both were described in a similar way as the company was used to.

User histories are the way XP captures customer requirements, guides planning, design, testing, integration, and the general project development. The XP user histories were introduced in Unisoft as a way of requirement elicitation. A set of user histories is developed in an XP iteration, and they may be affected by the work in subsequent iterations. User histories have been adopted by Unisoft as a basis for planning and test design, so system verification and validation is also achieved more easily. For managing this information, XP makes the user involved as part of the development team; however, it is necessary to control the management process by assigning a person responsible and some resources (the person in charge may even be the customer, if he/she has the skills).

The project leader was defined as the product owner (as prescribed in Scrum); he was in charge of managing engineering practices and requirement management (plans and execution), of managing changes and of continuously evaluating the state of the execution compared to plans and resources. He defined the practices (based on Scrum's sprint requirement), managed changes (according to management policies closely related to values and principles in the Agile Manifesto), and he was in charge of accepting, rejecting and documenting the history of these changes.

Associated with the requirement process, a light requirement management process was defined, including a requirement tracking process for different parts of the software lifecycle and a management process for the technical process of requirement engineering (elicitation, analysis, documentation and validation). This configuration fitted the company.

6.1 Case Analysis

In the theoretical analysis, the areas were divided according to the CMMI model but, in the application case, areas were not approached separatedly. In particular, the implementation of the requirement management area needed other areas to be developed as well: (1) technical solution: provided that the requirement management is present along the whole software development and it particularly manages the requirement engineering activities; (2) quality assurance: when defining and managing activities related to requirement validation; (3) configuration management: provided that requirements are also configuration items. Even though the applied analysis did not include the technical solution, part of it had to be implemented according to Scrum recomendations keeping the scenario of the agile methods exclusive contribution.

The areas of planning and tracking and control had a better initial evaluation, reaching 82% and 66% respectively. So, when requirement management was incorporated into the pilot experiment, in practice these areas improved the final evaluation (84%) because there was a control of the whole process of requirement management. These empirical results are better than the one theoretically

found (75%). We can then conclude that reaching these other two areas helped lowering the fulfillment delta in requirement management (and probably in other areas too).

In our case study, this difference could have been larger if these areas had not been so well evaluated, and could have been worse if the technical solution area had not worked so hard. This suggests that the original analysis should be complemented so to include correlations among areas and in this way the fulfillment delta could be lower. So the stated fulfillment delta we found theoretically in this article can be considered as the maximum partial fulfillment delta required for a set of agile process assets to implement CMMI. Considering that configuration management is strong at the code level, that part of it is implemented with the requirement management (provided that requirements are configuration items), and that there are few other development artifacts when using agile methods, configuration management would not be a problem any more, making it possible to reach in practice a fulfillment delta between 5% and 10%.

7 Conclusions

It is important to realize that is possible to reach maturity level 2 using agile methods if the following issues are addressed:

- Each process area must be explicitly defined within the organization. Agile methods provide almost all the required elements for this purpose.
- Process areas such as requirement management, measurement and analysis, quality assurance and configuration management should be complemented with elements obtained from other sources, in order to achieve a fulfillment delta 0.

In this work we have found that it is possible for small software development organizations to achieve a CMMI certification implementing agile methods. We have defined the fulfillment delta and tested it in practice in a small Colombian software development organization.

The weakest process areas in agile methods were found to be requirement management, measurement and analysis, and product and process quality assurance. With respect to configuration management, the effort should be even harder because agile methods cover only a little part of this area.

We also conclude the need to count on a set of process assets that allow organizations to define and instantiate a process through elements of agile methods. We thus need a process asset catalog that includes techniques, practices, process elements (roles, disciplines, phases, life cycles, etc.) that make it possible to implement the requirements in a practical way. All processes must have the following basic elements:

- a clear policy, obtained from values and principles of the agile methods;
- workflows implementing in detail each part of the life cycle: artifacts, activities and roles, obtained from artifacts, practices and roles defined by agile methods.

Each process is particular and unique because it depends on the goals and the structure of the organization where it is applied. So, it is important to define a catalog of process assets in

a structured and organized way, and providing a series of application guidelines, so that a small organization can adopt and adapt them to be able to reach a CMMI certification. In the near future, we plan to build this catalog structure and populate it with some example process assets. In a not so near future, we plan to completely populate the catalog and use it in academic as well as industrial environments. We will then be able to measure its effectiveness to improve software development processes consistently with SIMEP-SW goals.

Acknowledgments

This work has been developed as part of the SIMEP-SW project - Colombian Software Development Process Improvement System, funded by Universidad del Cauca, Colciencias, and SITIS Ltda. under contract 421 of 2003 Colciencias-Unicauca.

References

[1] Scott Ambler. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. Wiley Computer Publishing, 2002.

[2] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.

[3] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for Agile Software Development, February 2001. `http://agilemanifesto.org`.

[4] Alistair Cockburn. *Agile Software Development*. Adisson-Wesley, 2002.

[5] J. Cooper and M. Fisher. Software Acquisition Capability Maturity Model. Technical Report CMU/SEI-2002-TR-010, Software Engineering Institute, 2002.

[6] Martin Fowler. The New Methodology., April 2003. `http://www.martinfowler.com/articles/-newMethodology.html`.

[7] Tom Gilb. Evolutionary development. *ACM SIGSOFT Software Engineering Notes*, 6(2):17–17, April 1981.

[8] Tom Gilb. *Principles of Software Engineering Management*. Addison-Wesley, 1988.

[9] Jennifer Gremba and Chuck Myers. The IDEAL^{SM} Model: A Practical Guide for Improvement. *Bridge*, (3):19–23, 1997.

[10] Jim Highsmith. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House, 2000.

[11] ISO. Software Process Assessment - Part 2: A reference model for processes and process capability. Technical Report ISO/IEC 15504 TR2:1998, International Organization for Standardization, 1998.

[12] ISO. Software Process Assessment - Part 4: Guide to conducting assessment. Technical Report ISO/IEC 15504 TR2:1998, International Organization for Standardization, 1998.

[13] ISO. Software Process Assessment - Part 7: Guide for Use in Process Improvement. Technical Report ISO/IEC 15504 TR2:1998, International Organization for Standardization, 1998.

[14] ISO. Tecnología de la Información Proceso de Ciclo de Vida del Software. Technical Report ISO/IEC 12207 - UNE 71044, Asociación Española de Normalización y Certificación, 1999.

[15] ISO. Quality management systems. Technical Report ISO 9000:2000, International Organization for Standardization, 2000.

[16] ISO. Information Technology - Software Life Cycle Processes Amendment 1. Technical Report ISO/IEC 12207 AMENDMENT 1, International Organization for Standardization, 2002.

[17] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[18] Mark Lycett, Robert D. Macredie, Chaitali Patel, and Ray J. Paul. Migrating Agile Methods to Standardized Development Practice. *IEEE Computer*, 36(6):79–85, June 2003.

[19] Patrcio Maller, Claudio Ochoa, and Josep Silva. Agilizando el Proceso de Producción de Software en un Entorno CMM de nivel 5. *Revistas del IEEE América Latina*, 3(1), March 2005. Special Edition - JISBD'2004. IX Jornadas de Ingeniería del Software y Bases de Datos.

[20] Jerzy R. Nawrocki, Bartosz Walter, and Adam Wojciechowski. Toward Maturity Model for eXtreme Programming. In *27th EUROMICRO Conference 2001: A Net Odyssey*, pages 233–239, Warsaw, Poland, September 2001. IEEE Computer Society.

[21] OMG. SPEM, Software Process Engineering Metamodel Specification. Version 1.0. Technical Report 02-11-14, Object Management Group, 2002.

[22] Stephen R. Palmer and John M. Felsing. *A Practical Guide to Feature-Driven Development*. Prentice Hall, 2002.

[23] Mark C. Paulk. Extreme Programming from a CMM Perspective. *IEEE Software*, 18(6):19–26, December 2001.

[24] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles Weber. Capability Maturity Model. Technical Report CMU/SEI-93-TR-24, DTIC Number ADA263403, Software Engineering Institute, 2003.

[25] Ken Schwaber. The Scrum development process. In *OOPSLA '95 Workshop on Business Object Design and Implementation*, Austin, Texas, USA, October 1995. ACM Press.

[26] SEI. IPD-CMM - Integrated Product Development. Technical Report CMU/SEI-MM-97-001, Software Engineering Institute, 1997.

[27] SEI. CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1) Staged Representation. Technical Report CMU/SEI-2002-TR-012 ESC-TR-2002-012, Software Engineering Institute, 2002.

[28] SEI. Standard CMMISM Appraisal Method for Process Improvement (SCAMPISM), Version 1.1: Method Definition Document. Technical Report CMU/SEI-2001-HB-001, Software Engineering Institute, 2002.

[29] Jennifer Stapleton. *Dynamic Systems Development Method - The method in practice*. Addison-Wesley, 1997.

[30] Christ Vriens. Certifying for CMM Level 2 and ISO9001 with XP@Scrum. In *Proceedings of the Agile Development Conference (ADC'03)*, pages 120–124, Salt Lake City, Utah, USA, 2003. IEEE Computer Society.