# A SME-oriented extension of Agile development process

Gianmario Motta[1], Daniele Sacco[2], Thiago Barroero[3]

Dipartimento di Ingegneria Industriale e dell'Informazione
University of Pavia, Via Ferrata 1, Pavia, Italy
[1]motta05@unipv.it, [2]daniele.sacco01@ateneopv.it, [3]thiago.barroero@unipv.it

**Abstract.** This work intends to contribute on Agile i.e. an agile approach to the software development cycle. Agile is already popular in many organizations and our work intends to define an Agile methodology for small businesses extended by a customer-driven and goal oriented approach to requirements analysis. Our objectives are (a) an overview of Agile adoption and its basic characteristics, (b) a discussion of the possible issues in Agile practices that may affect small businesses, (c) an extension of the Agile methodology by Goal Oriented Analysis, (d) explanation by an example from a case study of real small business implementing our methodology.

**Keywords:** Requirement Engineering, Software Development Process, Software Lifecycle, Software Delivery Model, Small Business Needs

## 1 Introduction

Several researches suggest that the size of a business and its level of adoption of IT are related. These studies reveal that IT coverage increases with the size of business (more than 20 employees, almost 100% of adoption) [1],[2],[3]. However, in front of the exponential growth of IT, the rate of the adoption by small businesses is still low. This gap may be explained by the unique characteristics of small businesses, that can be defined as 'natural barriers' [4],[5],[6]: (a) small business generally have limited access to the market information, (b) management techniques are rarely used by small business, (c) small business relies on short term planning, uses both informal and dynamic strategies, and does not use standard operating procedures, (d) small business controls less resources than large organizations.

Furthermore, IT adoption belongs to 'cultural barriers', which typically depend on management vision, that may include lack of [7]: (a) IT knowledge combined with difficulties to find useful and impartial advice, (b) use of external consultants and vendors, (c) understanding of the benefits that IT can provide, and how to measure them. However, small businesses have a great advantage: they can be flexible. They are able to preserve work relationships, bring a 'personal touch' to operations, reach niche markets. The constant pressure also persuades them to be inventive and innovative in their operations [7].

Given these characteristics, we assume that software development process should be agile to support small businesses. In 2001, the Manifesto for Agile Software Development was published to define a new approach by providing adaptive planning, evolutionary development and delivery, and rapid and flexible response to change. The Agile Manifesto states that the "highest priority is to satisfy the customer through early and continuous delivery of valuable software" [8]. Agile has been one of the first approaches summarizing new perspectives in software development; for instance in 1999 Truex, Baskerville and Klein [9] already identified new emergent goals in information systems development, such as (a) always analysis, (b) dynamic requirements negotiations, (C) incomplete, usefully ambiguous specifications, (d) continuous redevelopment. Indeed, Agile development follows the same way by facing uncertain and changing situations. Management and customers can review and change the project thanks to the short iterations of development process; in this way, every feature is kept up to date; this overcomes the traditional waterfall model where requirements are defined at the beginning and the customer is not involved until the final user acceptance tests. There are many Agile methodologies and their complete overview would be out of scope, but in summary, an agile approach differs from the classic waterfall development of custom software by these characteristics:

- Requirements are defined gradually, step by step
- Requirements are defined and implemented on a partial, abridged version of the target software system
- Each provisional implementation is reviewed by users and requirements are refined
- Such iterations continue until the users' needs are satisfied and the system is complete.

The structure of the Agile delivery model focuses on project execution and adaptation. The model we refer to was first described by Highsmith [10], where he describes the departure from traditional software development phase names - Initiate, Plan, Define, Design, Build, Test – by replacing them with: (a) Envision, to indicate the criticality of vision, (b) Speculate, to indicate that it is of no use to "plan" uncertainty away but actually to face it step by step. (c) Explore, with its iterative delivery, (d) Adapt, to adapt to current conditions and start again with Speculate phase, (e) Close, whose primary objective is knowledge transfer. Basic design elements of Agile are (a) user stories, (b) development tasks that implement a single user story, (c) features realized by user stories, (d) capabilities as collections of features; typically a set of user stories is developed in one or more iterations and a capability must be completed by a milestone [10, 11].

Agile is used increasingly, moving from high-tech companies to mature industries as insurance, telecom and financial services. Actually, companies need an iterative process that enables small teams to build software functionalities in an environment that is responsive to business change, to improve time to market, development costs, and quality [12]. A study by Forrester Research [13] shows that enterprises are rapidly moving to Agile development: almost 35 % of the surveyed 1,298 developers and IT professionals use Agile methods. In large and small companies, adoption of Agile is almost the same [14]. Agile implementation has positive feedbacks: accelerated time-to-market, enhanced ability to manage changing priorities, increased productivity, enhanced software quality and improved alignment between IT and business objectives [15]. A survey [16] confirms Agile benefits: flexibility and quality software that meets customer needs. Other substantial benefits are knowledge sharing and lower risk of project failure [16]; Version One survey [15] reveals that 22% of respondents did not experience a failed Agile project.

Thus, Agile development benefits organizations. The continuous testing, instead of a final testing phase, is an effective practice. Short iterations and sprint reviews improve organizational commitment. Such constructive and frequent feedback helps to keep teams engaged in the project. An agile process, finally, approaches customers to development; customers feel more satisfied and involved in decisions.

## 2 The Agile issues

Agile is very demanding both on overall organization of the development and on analysis techniques.

On the organizational side, Agile can be constrained by organizational and managerial resistances. Lack of skills and project complexity are also barriers. In an Agile development, customer collaboration is a must, and a heavy customer involvement is critical. Difficult communication between customers, managers and developers leads to the failure of the whole Agile process.

On the analysis side, Agile requires a responsive requirement elicitation technique. In a classic Agile approach, requirements are elicited by user stories : "A user story is a brief statement of intent that describes something the system needs to do for the user" [17]. The description used by user stories is understandable to all project stakeholders. User stories do not consider only the functional perspective, but also the value defined by the user. In fact, user stories are very different from use cases on several aspects [17]. User stories are helpful to fill the gap between developer and user, but sometimes more precision is required . They may have some limits:

- It is hard to split business requirements into independent user stories, so dependencies may be introduced without appropriate modeling or architecture.
- Systems cannot be only described by using words, that may have different meanings to different people. Thus, interpretation may mislead if a model is not used.
- User stories are only functional. The customer cannot understand all non-functional requirements.

Thus, user stories should be replaced by a more robust technique. This is precisely our purpose. In order to understand needs, an analyst should explicit their vision and diverse viewpoints [18]. Traditionally, requirements elicitation is accomplished by conceptual modeling techniques which propose an abstract view about what the system should do [19]. Traditional conceptual modeling allows to understand the semantics of information, but it often fails in enabling acceptance by users. Researches show that many large projects fail because of an inadequate understanding of the requirements. Davenport stated: "IT is an effective implementation vehicle of innovation, but only when coupled with the approach, enablers, and other implementation factors" [20].

In order to get participative and effective requirements elicitation a possible way is to focus on the goals of stakeholder classes. The concept of goal is prominent in recent approaches to requirements elicitation and GORE[1] approaches emerged in this research area. Goals are prescriptive statements of intent whose satisfaction requires the cooperation of actors in the software environment. According to Pohl "goals represent the objectives an actor wants to achieve when requesting a certain service and it is used to describe an objective to be achieved in the macro-system, e.g. business goal, personal goal etc." [21]. GORE uses goals for eliciting, elaborating, structuring, specifying, analysing, negotiating, documenting, and modifying requirements [22]. GORE works at different level of granularity: the analyst identifies goals and refines them until they are reduced to alternative collections of requirements. In particular, requirements should be specific to each class of stakeholders since different stakeholders have different needs and goals.

## 3 Extension of Agile

For requirements identification we propose GOA[2], a technique for requirements engineering described by Bolchini and Paolini. GOA is a lightweight and intuitive methodology [23]. In contrast to the task analysis that focuses on what users do on the system, GOA identifies the objectives of all stakeholders, facilitating the exploration of design alternatives and leading to a more comprehensive set of requirements [24]. GOA doesn't belong to a particular software development methodology and our purpose is to use it in the Agile methodology.

GOA is useful at the initial stage of requirements analysis and task analysis is appropriate in the later stages of design, such as the detailed design of the interaction. GOA identifies seven categories of requirements: content (labeled with C), structure of content (S), access paths to content (A), navigation (N), user operation (U), system operation (O), presentation (P). In this way it provides a solution to user stories critical points.

**Table 1.** A comparison between user stories and goal oriented analysis

| User Stories | Goal Oriented Analysis |
|---|---|
| Customer on site | Low customer commitment |
| Customer participates in the elicitation | Customer confirms the analysis |
| Requirements are uncertain because they are not modelled and use only plain words | Requirements are well-defined in a model |
| They collect functional requirements | It collects functional and non-functional requirements |

[1] Goal Oriented Requirements Engineering
[2] Goal Oriented Analysis

Furthermore, GOA helps to specify non-functional requirements (typically system operation taxonomy). In this way, designers are facilitated in their work: "they receive functional requirements organized according to a useful taxonomy; for each requirements category, they can assess the quality of the design solutions (deriving from the functional goal-analysis) by respecting the non-functional requirements" [23]. Thus, GOA and user stories show different characteristics that we summarize in Table 1.

To explain our technique we illustrate a case study on a small company (10 workers) that manages job proposals for graduated students across Europe. Their aim was a virtual work environment in order to have one job scouter in each European country and to manage from any place the information about proposals, students and companies. To achieve this, a KMS[3] solution was planned to organize and facilitate collaborative creation of documents. We have identified 4 stakeholders: directors, marketing employees, host companies, applicants. Here are the main steps performed:

1. Definition of goals-requirements diagrams: goals represent the long-term needs and expectations of the stakeholders of the system and they can be decomposed in sub-goals in order to specify more accurate needs. Each goal is detailed and refined in requirements. Finally, requirements are classified by a label that indicates the design dimensions they have implications on. Fig. 1 shows the diagram for directors.

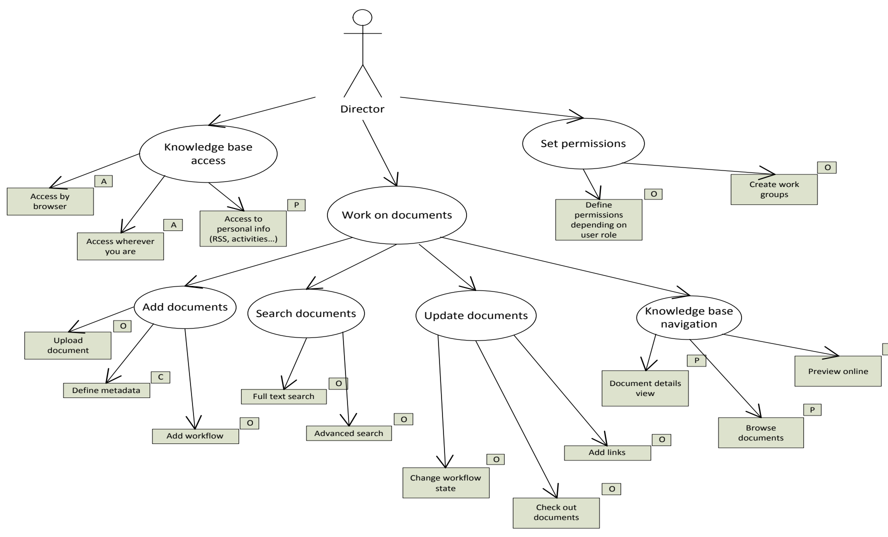
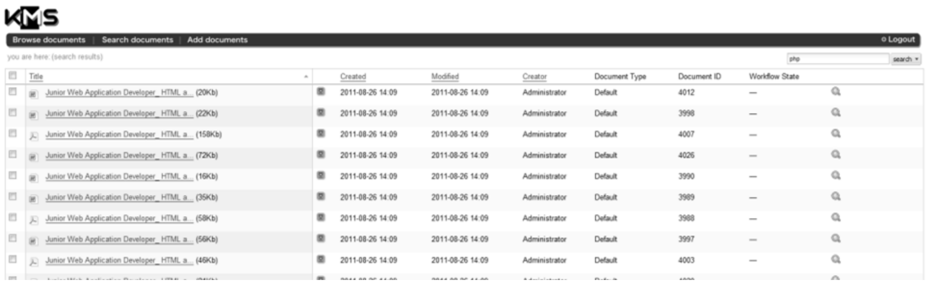


**Fig. 1.** An example of goals-requirements diagram

2. Definition of mock-ups: graphical user interfaces provide an idea of the new system by illustrating features and layout. Fig. 2 shows an example of mock-up for search feature, by illustrating the fields that must be shown in the search results.

[3] Knowledge Management System



| | Title | | Created | Modified | Creator | Document Type | Document ID | Workflow State |
|---|---|---|---|---|---|---|---|---|
| | Junior Web Application Developer_ HTML a... (20Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 4012 | — |
| | Junior Web Application Developer_ HTML a... (22Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 3998 | — |
| | Junior Web Application Developer_ HTML a... (158Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 4007 | — |
| | Junior Web Application Developer_ HTML a... (72Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 4026 | — |
| | Junior Web Application Developer_ HTML a... (16Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 3990 | — |
| | Junior Web Application Developer_ HTML a... (35Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 3989 | — |
| | Junior Web Application Developer_ HTML a... (58Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 3988 | — |
| | Junior Web Application Developer_ HTML a... (56Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 3997 | — |
| | Junior Web Application Developer_ HTML a... (46Kb) | | 2011-08-26 14:09 | 2011-08-26 14:09 | Administrator | Default | 4003 | — |

**Fig. 2.** An example of mock-up

3. Interviews definition and submission: after designing the proposals to be presented to the stakeholders, it is necessary to write an interview for each class of stakeholders. By this interview they validated and confirmed mock-ups and goals-requirements diagrams, suggesting changes depending on their needs.

These steps conclude the participation of stakeholders; afterwards they will be involved in acceptance tests at the end of each development iteration. Identified requirements are classified in functional or non-functional requirements according to their taxonomy. User stories extend functional requirements that we have identified; they describe in plain words the steps that compose each requirement. Since Agile process is mainly user-centric, we also introduced cards that describe non-functional requirements: we named them technical occurrences. Their functionality is to provide constraints during the development iterations; in this way developers consider every detail of the feature to develop. Fig. 3 shows some examples.

| | |
|---|---|
| **Name** | Upload document |
| **Taxonomy** | User story |
| **Description** | 1. Access to the knowledge base<br>2. Select the folder where the user wants to upload a document<br>3. Press «upload document» button<br>4. Choose the document to be uploaded<br>5. Complete metadata<br>6. Define document type<br>7. Upload document |
| **Priority** | High |
| **Exploration factor** | High |

| | |
|---|---|
| **Name** | Full text search |
| **Taxonomy** | Technical occurrence |
| **Description** | - Measuring concept: find keywords within a document<br>- Measuring method: time to complete the task<br>- Current level: $T_{manual}$ = The user must open each document and look for the keywords within the document<br>- Planned level: $T_{automatic}$ = 'full text search' shall provide the documents in few ms<br>- Worst case: $T_{worst} = T_{automatic} + T_{manual}$<br>- Best case: $T_{best} = T_{automatic}$<br>Worst case is tolerable because the probability of no indexing of a document is very low |
| **Priority** | High |
| **Exploration factor** | Medium |

**Fig. 3.** Examples of user story card and technical occurrence card

A feature is composed by user stories and is constrained by specific technical occurrences; moreover a feature shall satisfy mock-ups identified in the requirements elicitation phase. The output of this phase is a backlog that lists all features and stories that the production team has identified. This backlog data is mainly used for next release planning in order to identify priorities, risks and estimates. Class diagram in Fig. 4 summarizes relationships among all these entities.

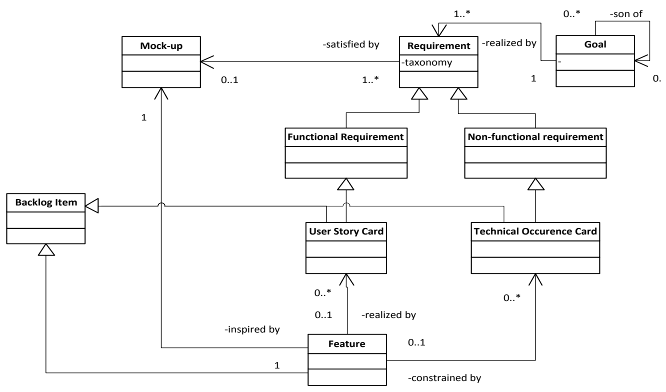


**Fig. 4.** Class diagram supporting our Agile extension

## 4 Conclusions

We have discussed an approach to enhance the agile development approach for small businesses. The enhancement consists in replacing the easy but loose user stories by a more structured technique that captures both functional and non-functional requirements while keeping an effective user involvement. A field validation on an actual project shows many benefits and we gave an example in section 3.

The roadmap in Fig. 5 illustrates the core of the methodology we have identified. The main target are small projects: customers commitment is not heavy and their satisfaction is high. Moreover, our requirement elicitation method demonstrated to be particularly effective on functional domains (i.e. those application domains where the customer focuses on the goals to be reached by the functions of the system). The quadrant in the right corner of Fig. 5 summarizes the results: our reference model is conceived to describe function-oriented solutions that respond to real needs of agility in small companies. Future works include a complete proof of concept and the improvement of the reference model by providing new extensions.

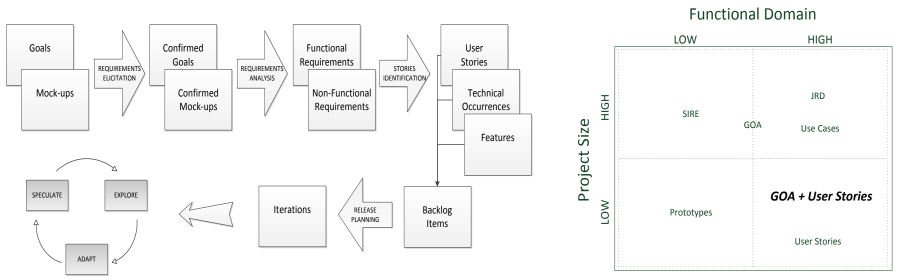


**Fig. 5.** Methodology and positioning

## References

1. McDonagh, P., Prothero, A.: Euroclicking and the Irish SME: Prepared for e-commerce and the single currency?. Irish Marketing Review, 13(1), 21-33 (2000)
2. Telstra Corporation and the National Office for the Information Economy. Small Business Index: Survey of Computer Technology and E-Commerce in Australian Small and Medium Businesses. Pacific Access Pty. (2000)
3. Duxbury, L., Decady, Y., Tse, A.: Adoption and Use of Computer Technology in Canadian Small Businesses: A Comparative Study. In S. Burgess (Ed.), Managing IT in small business: challenges and solutions, pp. 19-47 (2002)
4. Madrid-Guijarro, A., Garcia, D., Van Auken, H.: Barriers to innovation among Spanish manufacturing SMEs. Journal of Small Business Management, 47(4), 465-488 (2009)
5. Dibrell, C., Davis, P. S., Craig, J.: Fueling innovation through information technology in SMEs. Journal of Small Business Management, 46(2), 203-218 (2008)
6. Thong, J. Y. L., Yap, C. S., Raman, K. S.: Environments for information systems implementation in small businesses. Journal of organizational computing and electronic commerce, 7(4), 253-278 (1997)
7. Pollard, C. E., Hayne, S. C.: The changing faces of information systems issues in small firms. International Small Business Journal, April-June, 16(3), 70-87 (1998)
8. Agile Alliance. Manifesto for Agile Software Development. http://agilemanifesto.org/
9. Truex, D. P., Baskerville, R., and Klein, H. K.: Growing Systems in an Emergent Organization. Communications of The ACM, 42 (8), 117-123 (1999)
10. Highsmith, J.: Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House. (2000)
11. Hass, K. B. (2007). The Blending of Traditional and Agile Project Management. PM World Today, Vol. IX, Issue V
12. Richards, K.: Early mainstream: Agile develops in the enterprise. Application Development Trends (2006)
13. Forrester Research. Dr. Dobb's Global Developer Technographics Survey (2009)
14. West, D., Grant, T.: Agile Development: mainstream adoption has changed agility (2010)
15. VersionOne Research. State of Agile development survey (2010)
16. Vijayasarathy, L. R., Turk, D.: Agile Software Development: A Survey Of Early Adopters. Journal of Information Technology Management, 19(2), (2008)
17. Leffingwell, D.: Agile Software Requirements: lean requirements practices for teams, programs, and the enterprise, First Edition. Addison-Wesley. (2010)
18. Motta, G., Pignatelli, G.: Designing business processes for business performance: a framework, BAI, Seoul, 7-9 (2008)
19. Goguen, J.A., Linde, C.: Techniques for requirements elicitation, Requirements Engineering, 93, 152-164 (2003)
20. Davenport, T. H.: Process innovation: reengineering work through information technology. Harvard Business School Press, (1993)
21. Pohl, K., Haumer, P.: Modelling contextual information about scenarios, REFSQ, Barcelona, 187-204 (1997)
22. Van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour, RE, Toronto, 249-262 (2001)
23. Bolchini, D., Paolini, P.: Capturing web application requirements through goal-oriented analysis. WER, 16-28 (2002)
24. Bolchini, D., Mylopoulos, J.: From task-oriented to goal-oriented web requirements analysis. WISE, (2003)