Improving Estimations in Agile Projects:
Issues and Avenues

Luigi Buglione, Alain Abran

Abstract

From the mid '90s on, a number of Agile Methodologies have been proposed, most of them based on the basic values and principles summarized in the 2001 "Agile Manifesto". These agile methodologies were aimed at small teams with severe project constraints (i.e. small project teams in the same location, the customer as a member of the project team, informal communication, test-driven approach, etc.). Compared to more traditional project methodologies, Agile (or Lightweight) Methodologies are more detailed on Construction and Testing practices, but much less specific about other topics, such as Estimation. Currently, in most Agile Methodologies the experience of the team represents the basis for estimating from the high-level requirements.

The application of a Functional Size Measurement Method (FSMM) for estimation purposes raises a number of technical problems in Agile projects (i.e. unstable requirements, iterative SLC, non-functional requirements). A candidate solution is to combine an early sizing method for an agile project with a full FSMM method to be applied later in the SLC, when User Stories (the way XP labels high-level functional requirements) become available and are more stable. The goal of the paper is to identify estimation issues in the most known and adopted agile methodologies, looking at possible improvements at the organizational level.

1. Introduction

Starting from the mid '90s on, "*agile*" or "*lightweight*" methodologies have been proposed as a new paradigm for overcoming some problems in managing ICT projects such as the instability of user requirements. The focus of these agile methodologies (AM) has been mostly on short-term projects. Several agile methodologies (AM) have been developed, such as: XP (eXtreme Programming) [1], SCRUM [2], the Crystal family (Clear, Yellow, Orange and Red), FDD (Feature Driven Development) and DSDM (Dynamic System Development Method). The commonalities and basic values and principles of these AM have been documented in the 2001 the "Agile Manifesto". The focus of this paper is on understanding how estimation and measurement-related issues are managed in AM and on identifying areas for improvements at the organizational level.

Section 2 presents the rationale behind Agile Methodologies and a brief outline of the main ones. Section 3 discusses how the estimation issue is tackled in such methodologies. Section 4 provides suggestions for improving the estimation process in an agile context. Finally, Section 5 proposes conclusions and prospects for future work.

2. Agile Methodologies

Turbulent short-term projects, with unstable requirements and customers asking every day different things to their software providers represent the typical situation where an Agile Methodology can help project managers. "Agile" does not refer only to a "quick" or a

"shortcut" to achieve a certain goal; it refers also to a different way to organize the project work, taking into account multiple constraints that challenge project managers in their day-to-day work. Figure 1 uses a musical analogy to compare a project adopting a "*heavyweight*" software life cycle (SLC) approach (e.g. RUP, Spiral) and a "*lightweight*" one (e.g. XP, SCRUM etc.).

	SYMPHONIC ORCHESTRA Heavyweight	*JAZZ GROUP* Lightweight-Agile
PROJECT	• Each music session is a replay of a "*production series*"	• Each music session is a sort of "*prototype*"
PROCESS	• Structured process in a rigid manner (*following the music partition 'as is' in the music book*)	• Dynamic and adaptive process (*no music book*)
TEAM	• Many people well synchronized with actions	• Tight-knit team with few (*and skillful*) musicians
OUTPUT	• Stable and predictable outcomes	• Improvised and variable, but staying with established beats

Figure 1: Agility – A Musical Analogy [3]

While some organizations are attempting to scale up the usage of Agile methods to large software projects[1], their typical practices and values are profitably used by small teams under well-recognized constraints: small project teams in the same location, customer part of the project team, informal communication, test-driven approach, towards a Zero-defects approach for releasing the product[2], and so on.

2.1. The Agile Manifesto

In 2001, the community behind these AM developed and documented a consensus on underlying common principles for such methods, named the "Agile Manifesto" [4]:

> "*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
> **Individuals and interactions** *over processes and tools*
> **Working software** *over comprehensive documentation*
> **Customer collaboration** *over contract negotiation*
> **Responding to change** *over following a plan*
> *That is, while there is value in the items on the right, we value the items on the left more*"

A number of AM has been proposed based on these principles. Abrahamsson *et al.* [5] has presented a survey of the coverage among the best known Agile methods against the SLC

---

[1] The main reason is in the coordination effort of small teams, producing high-value returns at low costs. Nonetheless, the scalability of AM to large projects is of interest to a number of large ICT companies [6][7]. Refer to [1] Chapter 15 about XP and [2] Chapter 9 about Scrum.
[2] One of the main XP Testing practices is "no release till all unit test pass"
(http://www.extremeprogramming.org/rules.html)

phases and also in terms of "*what*" was covered (project management, process, practices/activities/work products). Fig.2 presents a high-level view of this analysis.
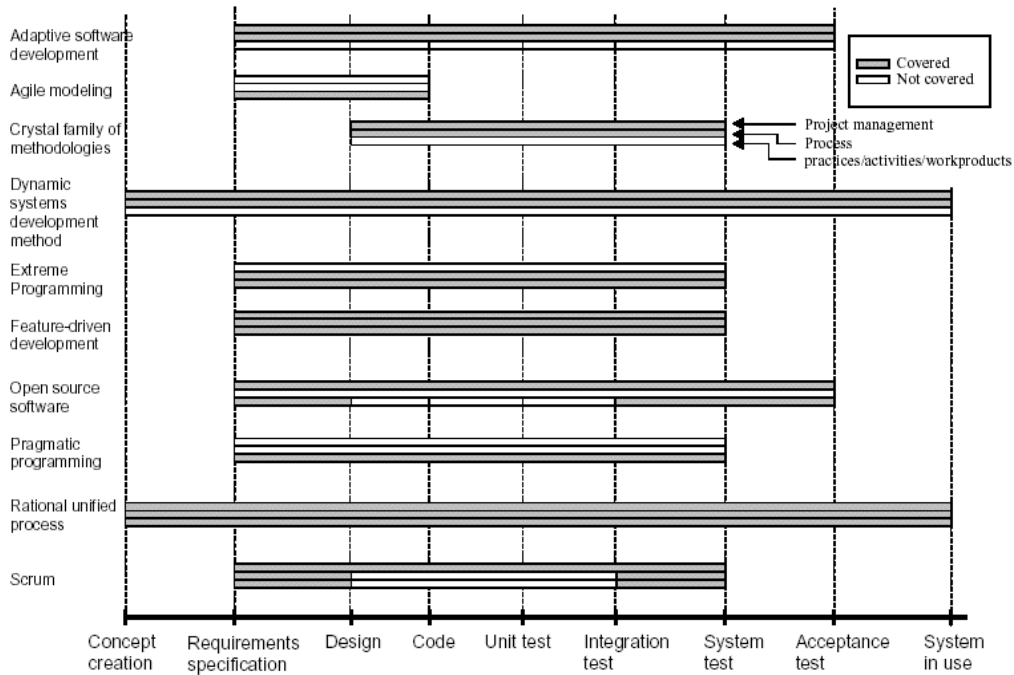

Figure 2: Phase-Activity Coverage of Agile Methods [5]

2.2. Agile and SPI methods

A few authors have looked into Agile methodologies for their relevance to software process improvement (SPI) models. For instance, Mark Paulk analyzed the coverage level of XP practices on Sw-CMM and CMMI [8], specifically at maturity levels 2 and 3. A similar type of analysis was proposed later by Ken Schwaber for SCRUM [2;10].

Table 1: XP and SCRUM compliance with Sw-CMM v1.1 KPAs

		RATING/COMPLIANCE	
ML	Sw-CMM KPA	XP [9]	SCRUM [2;10]
2	**RM** - Requirements management	++	++
2	**SPP** - Software project planning	++	++
2	**SPTO** - Software project tracking and oversight	++	++
2	**SSM** - Software subcontract management	--	--
2	**SQA** - Software quality assurance	+	++
2	**SCM** - Software configuration management	+	+
3	**OPF** - Organization process focus	+	+
3	**OPD** - Organization process definition	+	+
3	**TP** - Training Program	--	--
3	**ISM** - Integrated software management	--	--
3	**SPE** - Software product engineering	++	++
3	**IC** - Intergroup co-ordination	++	++
3	**PR** - Peer reviews	++	--
4	**QPM** – Quantitative Project Management	--	--
4	**SQM** – Software Quality Management	--	--
5	**DP** – Defect Prevention	+	--
5	**TCM** – Technology Change Management	--	--
5	**PCM** – Process Change Management	--	--
Legend:	+ Partially addressed; ++ Largely addressed; -- Not addressed		

From Table 1, some KPA would be fully covered at levels 2 and 3, some partially and some not at all. Readers will note some discrepancies betweens these various studies when these results from Table 1 are compared with Fig.1: for instance, the "Project Management" layer is empty in [5] while it is rated "++" in [8].

3. Estimation issues in an Agile software development project

It has been recognized over the past years that improving the estimation process requires an organization to continuously gather its projects' data, whatever the estimation technique (experience, analogy, regression analysis,…) used in the organization. Models such as MDB^{MM} [11] as well as the international standard ISO 15939 [12] provides support for measurement implementation, stressing the relevance and benefits of using its own historical data for estimation purposes.

The next sub-sections present a survey of estimation approaches used in various agile methodologies.

3.1. XP

In XP, the high-level requirements of the Customer, called User Stories (US), are expressed in a non-technical language and focus only on the functional side. Figure 3 shows an example of the main fields to be filled out in XP estimation (title, the code of the related acceptance test, the implementation priority, the number of Story Points and the functionality required, with a short text).

Title: *Waiting State*		
AccTest: *checkOptions0*	**Priority**: 1	**Story Points**: 2
When the Coffee Maker is not in use it waits for user input. There are six different options of user input: 1) add recipe, 2) delete a recipe, 3) edit a recipe, 4) add inventory, 5) check inventory, and 6) purchase beverage		

Figure 3: An example of User Story (Coffee Maker)[3]

Story Points (SP) represent one of the possible ways XP people estimate the time needed to produce and release certain functionality to the Customer: these estimates are "relative" numbers since each project team or author provides a different definition of what a SP measures[4].

Other estimation units and related concepts are: **Ideal Time** ("*the time without interruption where you can concentrate on your work and feel fully productive*" [13]), **Velocity** ("*to measure the project velocity you simply count up how many user stories or how many programming tasks were finished during the iteration. Total up the estimates that these stories or tasks received*" [14]) and **Load Factor** ("*The load factor equals actual calendar days to complete a task divided by the developer's estimated "ideal" days to do it*" [15]).

---

[3] URL: http://open.ncsu.edu/se/tutorials/coffee_maker/
[4] Stressed also by [16], chapter .6. For instance, Wake [17] associates one SP to one week, Shore [18] suggests one SP as an Ideal Day, Bossi [19] defines a SP as one total working hour (30' for a couple of programmers, in *pair programming)* and so on.

Those units are used in the "Planning Game"[5] when planning each project iteration, based on the evaluations of each US. If a US is "too big", XP rules suggest to split it into several US. At the end of the estimation process, US must be prioritized for scheduling into the iteration plan.
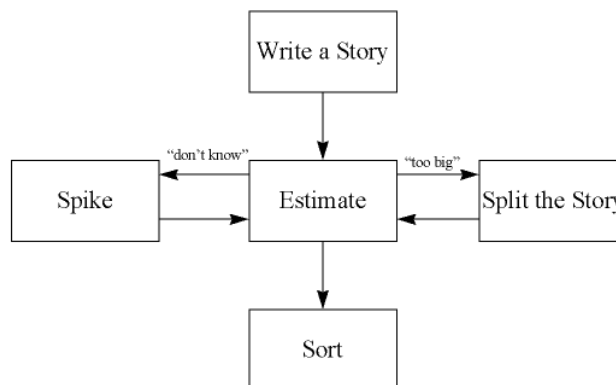

Figure 4: XP Planning Game[6]

Summarizing, XP practices suggest to estimate by experience/analogy[7], estimating the effort needed mainly based on the request of series of functionalities.

3.2. SCRUM

In Scrum, the basic concept for planning is the "*backlog*" and there are two possible levels of backlog: *product* (that coincides with the project) and *Sprint* (in Scrum, this is referred to as an iteration, usually a 30-calendar days period).
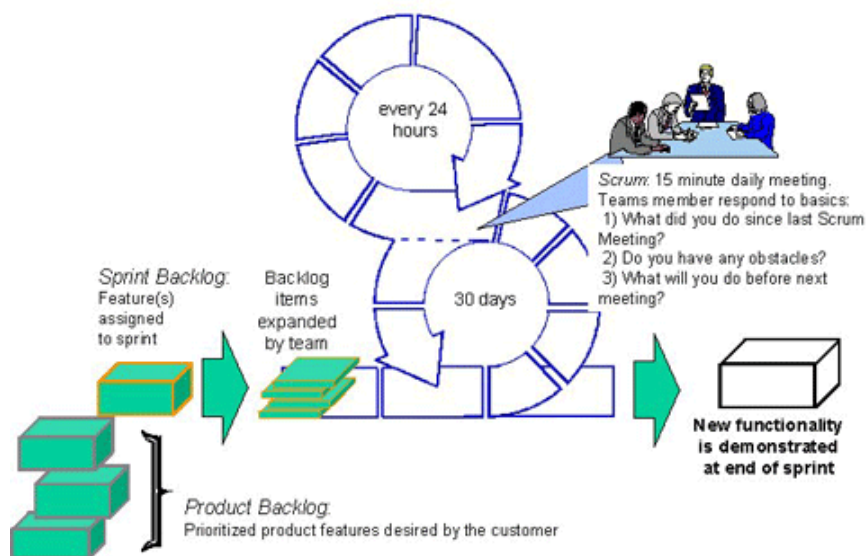

Figure 5: Scrum – The process[8]

The product backlog must be filled with the *features* requested by the Customer, to be evaluated in terms of the effort needed and then split into Sprints, according to the number of people working on the project and the available schedule time, with a micro-planning

---

reporting single *tasks* (as in work-breakdown structure, or WBS), refined daily after each *daily meeting*.

Sprint	ID		Backlog Item	Owner	Estimate (days)	Remaining (days)
			Product Backlog			
			Estimating System Upgrade			
1	1	Minor	Remove user kludge in .dpr file	BC	1	1
1	2	Minor	Remove cMap/cMenu/cMenuSize from disciplines.pas	BC	1	1
1	3	Minor	Create "Legacy" discipline node with old civils and E&I content	BC	1	1
1	4	Major	Augment each tbl operation to support network operation	BC	10	10
1	5	Major	Extend Engineering Design estimate items to include summaries	BC	2	2
1	6	Super	Supervision/Guidance	CAM	4	4
	7	Minor	Remove Custodian property from AppConfig class in globals.pas	BC	1	
	8	Minor	Remove LOC_ constants in globals.pas and main.pas	BC	1	
	9	Minor	New E&I section doesn't have lblCaption set	BC	1	
	10	Minor	Delay in main.releaseform doesn't appear to be required	BC	1	
	11	Minor	Undo modifications to Other Major Equipment in formExcel.pas	BC	1	
	12	Minor	AJACS form to be centred on the screen	BC	1	
	13	Major	Extend DUnit tests to all 40 disciplines	BC	6	

Figure 6: Scrum – Product Backlog[9]

Scrum does not define any unit of measure for the items included in the backlogs. Instead, you estimate directly the effort (in man-days or man-hours) for tracking both product and Sprints using the *burndown chart* that is, a graph showing how much time remains to complete the feature/tasks planned in such iteration, describing the project "velocity" the team is running.
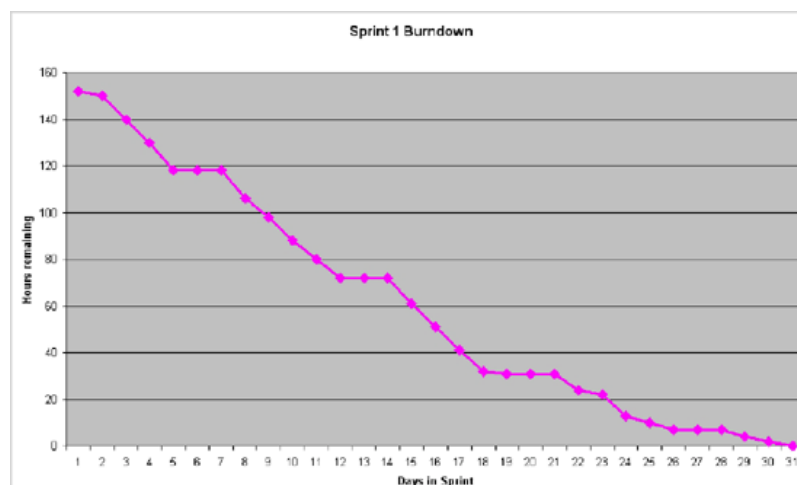

Figure 7: Scrum – Burndown Chart[10]

3.3. Crystal Family

Crystal Clear [22] is an agile methodology for a co-located team of 8 or fewer people. For estimation purposes[11] in the initial project interviews (technique #1) within its iteration cycle a 2-page long interview template must be filled in for each next team member, including some project's data and lessons learned from past projects. It is to be noted however that in the project's data records, there is no history of sizing data available.

---

[9] Figure from http://www.methodsandtools.com/archive/archive.php?id=18p2
[10] Figure from http://www.methodsandtools.com/archive/archive.php?id=18p2
[11] Cockburn suggests iteration of 1-week to 2-months long.

In the Blitz planning (technique #3) the Crystal equivalent of the User Stories, called Blitz Planning Cards, are estimated in Ideal days (to be refined into Estimated Elapsed Days) using Delphi analysis (technique #4) by some experts in the domain on the basis of counting content elements of such specific task (i.e. number of screens, business classes, …). The project progress is tracked, as in Scrum, with "Burn" Charts (technique #9, both burn-up and burn-down).

3.4. Feature Driven Development (FDD)

In Feature Driven Development (FDD v1.3 processes) [23], the estimation process consists in Process #2 to build the (functional-business) feature list, with a level of granularity for which each feature will not take more than two weeks to be completed; otherwise a feature must be decomposed into smaller features. The exit criteria for Process #3 (Plan by Feature) include the completion dates for each planned feature (month/year), properly assigned to team members.
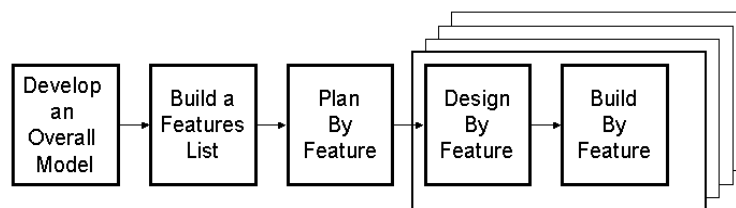
Figure 8: FDD – The process [23]

3.5. DSDM

DSDM [24] considers three main stages (pre-project; project; post-project), and a series of iterations within the project lifecycle, as shown in Figure 9.
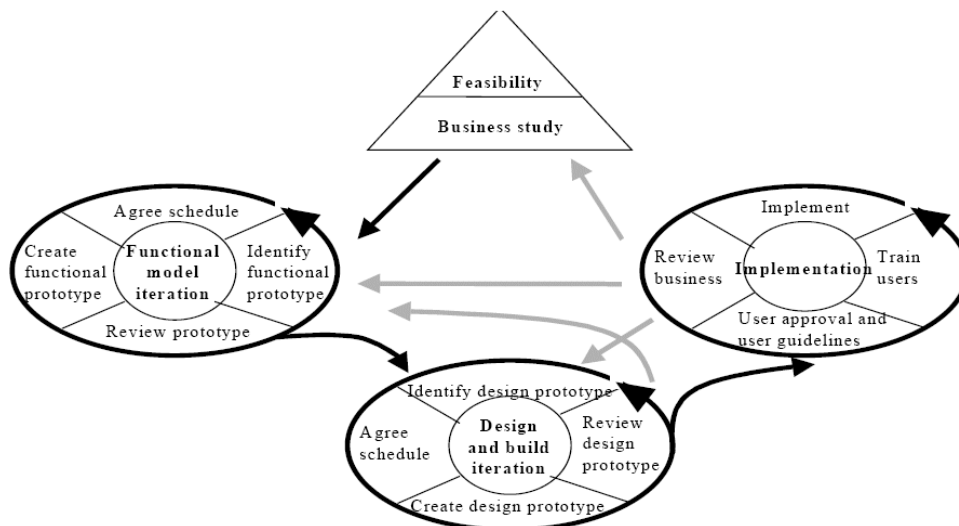
Figure 9: DSDM Lifecycle [24]

After the feasibility study, a prototype is built in the "Functional model iteration", taking care also of non-functional requirements (NFR). For planning (and estimation purposes), DSDM has four major phases: a pre-project planning, planning at the project start, planning during the project and finally, planning at the increment end. In its "Management tool and

techniques in DSDM project" section, the "Estimating" sub-section[12] also refers to "*the handling of contingency*".

Two approaches to estimation techniques are proposed:
- Top-down: based on business requirements, an early estimate is derived from high-level requirements, from a very high-level view on the project (i.e. estimation by analogy).
- Bottom-up: based on tangible systems, figuring out for low-level components, take time to be prepared, need sufficiently detailed information, provide a good basis for planning the project (i.e. Function Point Analysis, Estimating from System Components, Collaborative Estimating).

In contrast to other AM, DSDM explicitly says that "*it is essential to collect metrics from projects so that they can be validated and refined for use in estimating future projects, i.e. to provide continuous improvement of the estimating process*".

3.6. Other studies/proposals
In his book "*Radical Project Management*" [25], Thomsett analyzes candidate causes for wrongly estimating a project, and their priority, in the *funnel of increasing accuracy*.
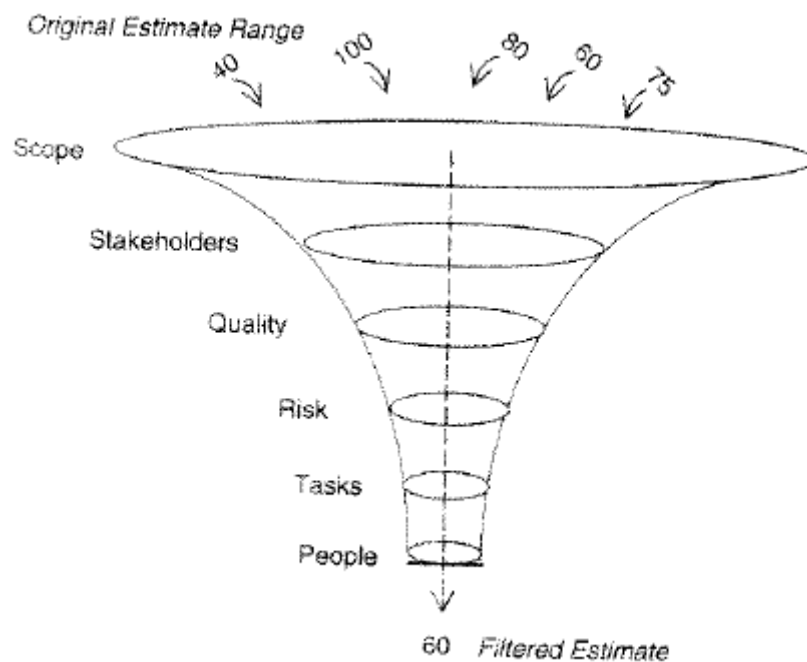
Figure 10: The Funnel of Increasing Accuracy [25]

The estimation process is based on a detailed analysis of the project WBS through wide-band Delphi estimates and sensitivity analysis (best, likely, worst cases), to be adjusted according to project's quality requirements.

4. Discussion on Estimation practices in AM
With respect to proposed estimation-related practices in AM, some common flaws and missing elements can be identified:

---

[12] URL: http://www.dsdm.org/version4/2/members/Estimating.asp

a) <u>No estimates for non-functional requirements of the projects</u> → the focus of AM on evaluating features, user stories, or more general high-level user requirements mainly from a functional viewpoint does not include the non-functional perspective of the projects. Explicitly, only few methods such as DSDM propose in their practices to consider them for estimation and planning purposes.

b) <u>No sizing units</u> → the general approach to estimation in AM is based on the experience of the team; the team then estimates effort (and sometimes only schedule data) based mainly on functional analysis. Such estimation by experience is not supported by historical data. For example: in AM, effort and schedule are the time-related expressions corresponding to "how much" work will be performed; they do not mention the functional requirements measurement with a recognized quantitative measurement method. In this AM approach to estimation, the size is only considered as "implicit" in the estimators' mind. Story Points and Velocity concepts represent a "delivery rate" based only on the experience and knowledge of such particular team, applicable with success in small and stable teams. But it is much more challenging in larger organizations where the composition of a project team is not necessarily as stable during time.

c) <u>No practices for gathering and using historical data</u> → the practice to collect data from projects is not required by most AM.

d) No standards applied<u>: an overall observation is that each methodology and each team applying a certain AM uses its own definitions</u>; adequate measurements require to be based on standards and input with a validated, quality level.

5. Conclusions & Prospects

Agile Methods (AM) such as XP, FDD and DSDM represent interesting solutions for projects with unstable requirements, iterative SLC, short-term milestones and small teams.

Only in the last five years, the attention was also paid to Project Management practices in agile methodologies. Through this interest, planning and scheduling practices have been tailored to agile methodologies, but with much less attention to the estimation process.

AM still being a young approach to software development, much work remains to be done to improve the way AM manage estimates, including tailoring relevant practices from well-established and proven "heavyweight" methodologies.

Next steps should include analyses of the impact on size, effort and productivity of non-functional requirements in agile projects to improve estimates right from the Requirement Elicitation phase using, for example, a new User Story structure called US^2 (2nd generation of User Stories).

6. References

[1] BECK K. & ANDRES C., *Extreme Programming Explained. Embrace Change*, 2/e, Addison-Wesley, 2005, ISBN 0-321-27865-8

[2] SCHWABER K., *Agile Project Management with SCRUM*, Microsoft Press, 2003, ISBN 0-7356-1993-X

[3] BUGLIONE L., *Agile Methodologies*, SEMQ website, URL: http://www.geocities.com/lbu_measure/agile/agile.htm; last access: Feb 1, 2007

[4] ---, *Agile Manifesto*, 2001, URL: http://www.agilemanifesto.org

[5] ABRAHAMSSON P., SALO O., RONKAINEN J. & WARSTA J., *Agile Software Development Methods. Review and Analysis*, VTT Publication # 478, 2002

[6] MC MAHON P.E., *Lessons Learned Using Agile Methods on Large Defense Contracts*, Crosstalk, Software Technology Support Center (STSC), May 2006, pp. 25-30, URL: http://www.stsc.hill.af.mil/crosstalk/2006/05/0605mcmahon.pdf

[7] VAIHANSKY P., SUTHERLAND J. & VICTOROV A., *Hyperproductivity In Large Projects Though Distributed Scrum*, Agile Journal, December 8 2006, URL: http://www.agilejournal.com/content/view/190/

[8] CMMI Product Team, *CMMI for Development, Version 1.2*, CMMI-DEV v1.2, CMU/SEI-2006-TR-008, Technical Report, Software Engineering Institute, August 2006, URL: http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr008.pdf

[9] PAULK M.C., *Extreme Programming from a CMM Perspective*, IEEE Software, Nov-Dec 2001, Vol. 18 No.6 , pp.19-26

[10] SCHWABER K., *It depends on Common Sense,* 2004, Presentation, URL: http://xpday2.xpday.org/KeynoteSlides.ppt

[11] BRAUGARTEN R, KUNZ M., FAROOQ A. & DUMKE R., T*owards Meaningful Metrics Data Bases*, Proceedings of the 15th International Workshop on Software Measurement (IWSM2005), Montréal (Canada), September 2005

[12] ISO, IS 15939:2002, *Information Technology – Software Engineering – Software Measurement process*, International Organization for Standardization, Geneva, 2002.

[13] BECK K. & FOWLER M., *Planning Extreme Programming*, 2000, Addison-Wesley, ISBN 0201710919

[14] --, *Project Velocity*, ExtremeProgramming.org, URL: http://www.extremeprogramming.org/rules/velocity.html

[15] --, *Load Factor*, ExtremeProgramming.org, URL: http://www.extremeprogramming.org/rules/loadfactor.html

[16] COHN M., *Agile Estimating and Planning,* Prentice Hall, 2005, ISBN 0131479415

[17] WAKE W., *Customer in XP*, XP123 website, June 2000, URL: http://xp123.com/xplor/xp0006c/index.shtml#PlanningGame

[18] SHORE J., *The Art of Agile Development*, 2007, URL: http://www.jamesshore.com/Agile-Book/estimating.html

[19] BOSSI P., *ExtremeProgramming Applied: a case study in the private banking domain*, January 2003, OOP2003 Conference, Munich (Germany), URL: http://www.quinary.com/pagine/downloads/files/Resources/OOP2003Paper.pdf

[20] BERGIN J., *Learning the Planning Game An Extreme Exercise*, July 27 2001, URL: http://csis.pace.edu/~bergin/xp/planninggame.html

[21] PMI, *A Guide to the Project Management Body of Knowledge* (PMBOK), 3/e, Project Management Institute, ISBN 1-930699-45-X, 2004, URL: http://www.pmi.org

[22] COCKBURN A., *Crystal Clear. A Human-Powered Methodology for Small Teams*, Addison-Wesley, 2005, ISBN 0-201-69947-8

[23] DE LUCA J., Feature Driven Development (FDD) processes v1.3, Nebulon Pty. Ltd., 2004, URL: http://www.nebulon.com/articles/fdd/download/fddprocessesA4.pdf

[24] DSDM CONSORTIUM, *DSDM Public Version 4.2*, 2007, URL: http://www.dsdm.org/version4/2/public/

[25] THOMSETT R., *Radical Project Management*, Yourdon Press, 2002, ISBN 0-13-009486-2