Six Sigma for Agile Teams

Dr. Thomas M. Fehlmann
Euro Project Office, Zurich
thomas.fehlmann@e-p-o.com

Abstract

Purpose *– This paper addresses the most common pain points in agile software development.*

Methodology/Approach *– Many experts agree that agile software development methodologies make good use of best practices – but how can they be measured quantitatively? Agile literature provides surprises for software measurement professionals: for instance, time-consuming size measurement approaches are proposed that produce unreliable and non-reproducible results. However, measurements for agile software development are readily available and less painful than in traditional environments.*

In this paper a combination of techniques used in Six Sigma for Software – Functional Size Measurement, Six Steps to Completion, and Quality Function Deployment – is presented making agile development processes lean and measurable. We use Six Sigma transfer functions to map development project controls onto process responses.

Findings *– Agile software development methodologies treat software development as knowledge acquisition by embracing change. They better map reality than traditional approaches do. This facilitates measuring software processes for performance and quality.*

The findings are valid for all kind of lean software development.

Originality *– The work originates from workshops at the PROFES 2010 conference in Limerick and the IWSM / MetriKon / Mensura conference in Stuttgart 2010 and has not yet been published in an international journal.*

Keywords *– Agile Software Paradigm, Scrum, Quality Function Deployment; Six Sigma for Software; Combinatory Metrics*

Paper type *– This paper had been presented at the 3rd European Research Conference on Continuous Improvement and Lean Six Sigma, Glasgow, 2011*

1 *Introduction*

Six Sigma orientates products and services towards measurable customer values. It provides tools and methods to eliminate and prevent defects whereby anything which diminishes customer values is termed a defect.

Contrary to common belief, a Six Sigma approach for Software has not much to do with bugs found by engineers: non-adherence to specification for instance does not automatically imply that customers see this as a defect. Specifications may be wrong, or may not be of any value to a customer, and hence non-adherence to specification won't be seen as a defect. This makes defect-counting difficult, especially in software, as only the customer value counts. Weaknesses that engineers classify as minor may end up as major defects in customer's perception, and vice versa. Customers might ignore major technical weaknesses of software if it does not hurt business. However, in traditional software development environments, the customer viewpoint is not readily available to the development team.

To make processes measurable for Six Sigma, various views of stakeholders such as engineers, customers, users, etc. have to be taken into account. In Six Sigma this is accomplished via *Transfer Functions,* mapping process controls onto the process response. In software development, transfer functions consist of measurable software engineering practices.

2 Agile Software Development

We assume the reader familiar with the agile software development paradigm; see for instance *(Schwaber and Beedle, 2008).* In this paper, we look at four pain points in agile:

1. Sizing user stories
2. Measuring Progress and define conditions when work is finished
3. Translating user stories into Story Items
4. Prioritising user stories

For all three problem areas, Six Sigma provides proven techniques fitting well into an agile development approach. Current techniques used in Agile Teams rely on Delphi techniques *(Cohn, 2005)* for sizing "*Story Points*". However, story points are non-reproducible agreements in development teams and do not refer to anything like a sizing measurement unit.

While the most popular project management method for Agile seems to be Scrum *(Roriz, 2009),* Six Sigma for Software fits best to *Lean* and *Test-Driven Development,* as stated in *(Poppendieck, 2007)* or *(Beck, 2000).*

3 Sizing User Stories with ISO 19761

3.1 User Story Example

Agile methodologies rely on *User Stories* that constitute the elements of the *Product Backlog.* They represent what the sponsor wants to get out of the software development process.

User Stories follow the wording scheme "As a … [functional user] … I want to … [respond to an event / retrieve some data] … [With some frequency and/or quality characteristic] … So that … [description of value or benefit is achieved]" *(Buglione and Trudel, 2010).*

A sample User Story reads as "As a Library User, I want to search for books by title, with speed and ease-of-use, so that I can find all books with similar titles" *(Rule, 2010).* This '*Library User Story*' will be used in the sequel for explaining use of Six Sigma in agile development.

3.2 The COSMIC Measurement Method

The COSMIC measurement method (ISO/IEC 19761) *(Abran, 2009)* identifies four subtypes of data movement for counting functional size: <u>E</u>ntry, e<u>X</u>it, <u>R</u>ead and <u>W</u>rite, each of which includes specific associated data manipulation. Data movements can be derived from the components identified in sequence diagrams according the RUP Methodology *(OMG UML, 2009).* Sequence diagrams create a common understanding among developers and sponsors, and allow identifying technical risks at an early stage. Hence they fit well into agile paradigm *(Fuqua, 2003).*

Functional sizing is simple: all identified data movements are summed up. Agile teams can visualise functional sizing by drawing sequence diagrams. When applying Six Sigma to software, size measurement is key precondition. Otherwise, Six Sigma indicators such as *Defects per Million Opportunities* (DPMO), defect density, or variation in standard distribution would make no sense.

The COSMIC measurement method works best together with *Structure Diagrams* as suggested by *(Rule, 2010).* Structure diagrams, when used and visualized, are focussing and structuring discussions among the team, as proposed by *(Ambler, 2004).*

3.3 Estimate Effort Needed for Implementing User Stories

User stories can be sized using the ISO/IEC 19761 functional size, and cost drivers can be identified from a standardized list of known influence factors *(Santillo and Moretto, 2010).* This saves valuable time by replacing the story point sizing sessions by structured discussion among the team. Contrary to story points, sizing with ISO/IEC 19761 is reliable, perfectly repeatable and thus, together with the cost drivers identified, supports effort estimations if benchmarks among comparable projects are available *(ISBSG, 2010).*

4 Measuring Non-Functional Requirements

4.1 Identifying Story Items

Note that in the library user story, three elementary data movements "Read Book Title", "Search in Book Data", "Present List of Books Found" are immediately identifiable using ISO/IEC 18761. These data movements constitute three functional Story Items needed to implement the user story.

However, there are four more Story Items hidden in the library user story: 1) the need for an extended title text search mechanism, 2) search speed optimisation, 3) an entry scan that forgives grammar glitches, and 4) a pattern matching search algorithm. These additional four Story Items are needed to implement the user story to the customer's full satisfaction, and are easily identified by agile teams; however, they are not independent user stories, and they don't constitute elementary data movements according ISO/IEC 19761. The three functional and the four quality-related Story Items implement the library user story. The user story paradigm stimulates both developers and sponsors in detecting additional Story Items compared to ISO/IEC 19761 alone. So when did they find all Story Items for implementing the user story, and how shall the team identify superfluous Story Items?

Functional sizing does not count non-functional requirements. However, developers implementing functionality or data movements have to know how to design a solution, use services and many other things, following non-functional requirements such as meeting performance or other quality expectations. The Design for Six Sigma technique identifies design controls that deliver the response as requested for the expected profile of business drivers, using transfer functions.

4.2 *Quality Function Deployment – A Six Sigma Tool*

Quality Function Deployment (QFD) has come to life in Japan during the 1980ies and is applied in various industries for determining customer's needs. Today, QFD is widely used in the automotive industry, but also in software design, requirements engineering, marketing services, decision making, and QFD is part of the Design for Six Sigma toolkit.

4.3 *Transfer Functions between User Stories and Business Drivers*

The basic idea in QFD is to deploy the desired profile $\underline{\mathbf{y}}$ of response topic – e.g. *Voice of the Customer*, or business drivers profile – into observable and measurable controls $\underline{\mathbf{x}}$ such that the transfer function between controls and response map these controls onto the expected response within defined variations. The transfer function is a linear mapping $\mathcal{T}$ between the profile vector $\underline{\mathbf{x}}$ and the response $\underline{\mathbf{y}}$. It is described as a matrix whose cells describe the impact of coupling between controls and response. Controls in software development are technical requirements; e.g., user stories, in QFD called *Voice of the Engineer*. The transfer function $\mathcal{T}$ has two aspects:

- It maps the control profile $\underline{\mathbf{x}}$ onto the business driver's space of the original response profile $\underline{\mathbf{y}}$. If $\mathcal{T}(\underline{\mathbf{x}}) \cong \underline{\mathbf{y}}$, the controls $\underline{\mathbf{x}}$ are the requirements for a valid solution with regard to the desired response;
- The transfer function describes what the process does – i.e., in software, it describes of the Story Items or the work instructions for the developers.

Transfer functions are important to agile teams since their matrix cells contain the Story Items needed to implement a user story such that the response profile meets customer's business drivers. However, contrary to traditional Six Sigma in manufacturing, the transfer function is adaptable in software development. It becomes 'agile' by itself.

Transfer functions in software development describe the combined effect of all design decisions and work instructions pertaining to that particular software development project. Thus, transfer functions allow agile teams to measure the consistency of their approach – while it's happening.

4.4 The Business Value of QFD

QFD workshop technique allows identifying suitable controls such that the response of the development process meets customer's needs. In agile, these controls are the user stories, and QFD yields their priorities based on customer's business driver profile. QFD uses the transpose $\underline{\mathbf{x}} = \mathcal{T}^{T}(\underline{\mathbf{y}})$ to identify suitable controls, and validates the result using the *Convergence Gap* $\|\underline{\mathbf{y}} - \mathcal{T}(\underline{\mathbf{x}})\|$, defined as the vector difference between the original business driver profile $\underline{\mathbf{y}}$, and the effectively achieved development process response $\mathcal{T}(\underline{\mathbf{x}})$. When the convergence gap closes upon all completed user stories, the project is finished and the goals are met.

For linear algebra fundamentals, see *(Kressner, 2005)*; how it works for Six Sigma for Software, see *(Fehlmann, 2005)*.

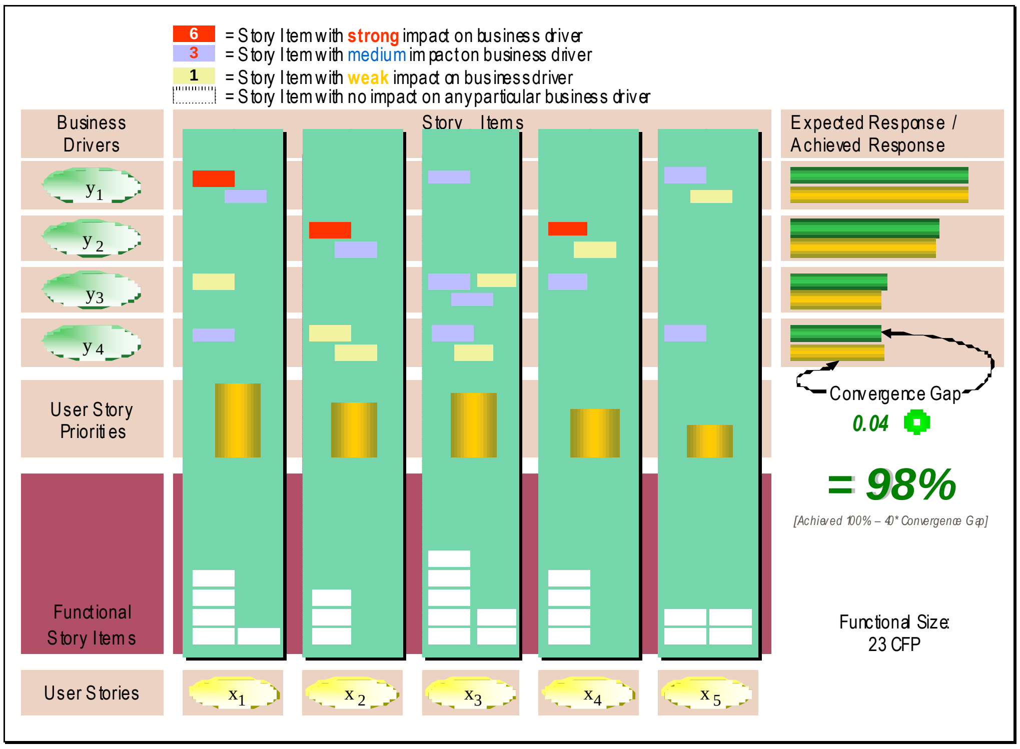
The convergence gap is computed in the response topic area, not in the control area of the QFD matrix. As a consequence, two solution approaches that yield the same responses are equivalent for Six Sigma metrics.

5 Blending Six Sigma with Agile

When working with agile development teams, we use an extended form of the QFD matrix which records all planned and completed user stories as controls, tracks sprint backlogs, and calculates the convergence gap against the business driver's profile of project goals. It is named the *Buglione-Trudel Matrix* following discussions at IWSM / MetriKon / Mensura 2010 in Stuttgart *(Buglione and Trudel, 2010)*.

The $\underline{\mathbf{y}}$-axis is labelled with the business drivers. They describe which qualities matter for the customer's business: customer loyalty, relevance, speed, performance, safety, security, privacy, correctness, completeness are typical for values that derive from the customer's business. They can be specific like 'personal health data of employees is accessible to Human Resource People only', or describe soft values like 'Create Trust among Users'. Business drivers should not prescribe technical solutions (like 'must be implemented with <some specific tool>'); business drivers must be something where software design choices have impact upon.

The **x**-axis is labeled with the user stories that the developers are expected to implement. The matrix cells contain the Story Items needed to implement a user story, identifying the respective business driver where it contributes.



6 = Story Item with strong impact on business driver
3 = Story Item with medium impact on business driver
1 = Story Item with weak impact on business driver
= Story Item with no impact on any particular business driver
Business Drivers
Story Items
Expected Response / Achieved Response
y1
y2
y3
y4
User Story Priorities
Convergence Gap
0.04
= 98%
[Achieved 100% – 40* Convergence Gap]
Functional Story Items
Functional Size: 23 CFP
User Stories
x1
x2
x3
x4
x5

Fig. 1. The Buglione-Trudel Matrix

If the project sponsor has set the business response profile **y**, the software design defines the transfer function $\mathcal{T}$. When the agile team creates user stories, it implicitly finds the solution **x** and optimises the transfer function $\mathcal{T}$ in order to meet the customer's business drivers **y** by 'balancing' the Buglione-Trudel matrix, until the convergence gap closes, indicating that the transfer function $\mathcal{T}$, when applied to **x**, will deliver what the customer expects. This eliminates work that adds no value, and makes agile software development effectively lean, following *(Poppendieck, 2007)*, or *(George, 2010)*.

In each sprint, the developers identify the Story Items needed for user stories selected according priority. Developers enter each Story Item into one or more matrix cells if it contributes to the respective business driver. Story Items are not restricted to functional features; they can refer to quality enhancements, such as creating a database schema that is extendible in future version of the software, feature re-factoring for better performance, improving ease-of-use. The creativity of developers proposing Story Items is restricted by the argument whether they contribute to some business driver, thus identifying Story Items that add value, and blocking those that don't.

Story Items that don't contribute specifically to any of the business drivers are recorded in the bottom row of the Buglione-Trudel matrix. Most functional

Story Items – e.g., basic functionality – will be placed in the bottom row, while quality tasks appear in the upper part of the Buglione-Trudel matrix. Typically, quality tasks don't add to functional size. Upper and lower part cells together serve as sprint backlog; summing up each column for functional size calculates the size per user story, and similar for planned and actual effort, see Fig. 2.

The *Correlation Values* assigned to the matrix cells represent the impact these Story Items have on each business driver. Completed user stories remain in the matrix; marked as 'completed'.

5.1 Using QFD for User Story Prioritisation

First of all, the QFD matrix prioritises user stories. Knowing the functional size, weighting their contributions to business drivers and checking the convergence gap for the response profile $\underline{\mathbf{y}}$ is the method of choice for selecting user stories for the next sprint, identifying those user stories that contribute much to business drivers and therefore are probably needed first.

For this, before knowing Story Items in detail, the matrix is filled in the traditional QFD manner with impact relationships valuated as 1-3-9 *(Creveling et. al., 2003),* indicating the impact developers expect that the user story has.

5.2 Harvesting Developer's Intelligence

The evaluation of the matrix cell is done as follows: the developers distribute impact points to all Story Items recorded in the matrix cells, indicating how much they contribute to each business driver. This can be done initially, at planning poker time, after the daily scrum meetings, and revised whenever needed. The developers use colour marker buttons that they can distribute; red points represent value 6, blue points represent value 3, and light green points represent value 1. Story Items in the "functional" bottom row receive no impact points, as functional user requirements are not specific on any of the business drivers, except in some rather rare special cases.

The total impact of a correlation matrix cell is the sum of all points for Story Items recorded in the matrix cells contributing to the respective user story. The profile $\mathcal{T}(\underline{\mathbf{x}})$ is calculated by weighting total impact with the business driver profile y, applying the matrix $\mathcal{T}$ to the vector $\underline{\mathbf{x}}$.

5.3 Using QFD as a Communication Tool

The result is a visualisation that is immediately understood by developers and customers alike. Areas with strong correlation in the matrix contain much red, indicating important control topics for reaching the stated goal. Areas that contribute less remain lighter colour or even white.

From the QFD practice it is known that people perceive a matrix with bad convergence gap as unbalanced; and well-balanced matrices in turn represent optimum convergence gap. Because of this, QFD has been very successfully applied as a transfer function visualisation workshop technique in many industries. QFD is a communication tool. The completed QFD matrix is balanced in the sense that all requirements are addressed but none is overachieved, as seen with the horizontal bars in Fig. 1. The team allocates or de-allocated Story Items according importance for the customer, to make the matrix look balanced. The convergence gap metric is a quality indicator for the quality of the software product. A small convergence gap indicates that all of the business drivers are addressed, and none over-achieved.

Note that one Story Item may appear several times in different cells of the column, in which case it contributes to more than one business driver. Quality impact is not always correlated to total effort, and effort can be optimised in preferring "cheap" Story Items that have high impact on business drivers.

6 Tracking Progress and Defining Completion Criteria

6.1 Six Steps to Completion

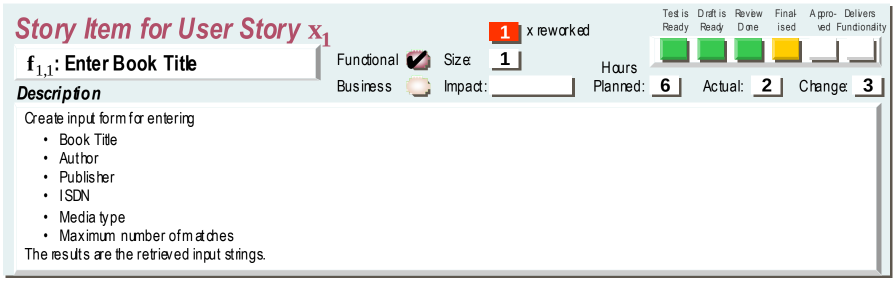
The Six Steps to Completion method is used in Six Sigma for Software to measure progress on Story Items. Progress on Story Item is classified in six steps, each having a completion rate assigned to it:

- *Test is Ready* stage: Is problem understood and does a unit test exist?
- *Draft is Ready* stage: does a full draft exist? i.e., is code completed?
- *Review Done* stage: has the code been quality checked?
- *Finalized* stage: usually, something is left for improvement.
- *Approved* stage: team and sponsor agree that the work is done.
- *Ready for Use* stage: other programmers or users rely on the Story Item.

The Six Steps to Completion measurement method allows for an unbiased assessment of the progress done in a sprint, and early identification of obstacles. It is a proven way to structure daily scrums *(Fehlmann, 2004)*.

6.2 Detailed Story Items

Fig. 2 is a detail view of one Story Item in the Buglione-Trudel Matrix:



Story Item for User Story x_1

1 x reworked

Test is Ready	Draft is Ready	Review Done	Final-ised	Appro-ved	Delivers Functionality
🟩	🟩	🟩	🟨		

$f_{1,1}$: Enter Book Title

Functional ✓ Size: 1

Business Impact:

Hours Planned: 6 Actual: 2 Change: 3

Description

Create input form for entering
- Book Title
- Author
- Publisher
- ISDN
- Media type
- Maximum number of matches

The results are the retrieved input strings.

Fig. 2. Sample Story Item in the Buglione-Trudel matrix while being executed

Story Items can be recorded on cards, or, if needed, in a tool. The author uses PowerPoint slides for both the matrix and their Story Items, as actually shown in this paper, backed by Excel spreadsheets for calculation.

The Story Items contain all information needed for self-organised agile teams, including references to the matrix cell, to functional size, to effort estimation and actual effort spent. Change effort needed because of new insights, new needs, or refactoring, is also recorded; it adds to actual effort. Progress indicators according Six Steps to Completion are marked green when completed, yellow when currently at work, and red if blocked by obstacles. Since all information is measurable, together they constitute a set of simple, visually communicated metrics that all stakeholders easily understand.

7 Lessons Learned

7.1 Benefits

The key lesson is that – notwithstanding what kind of software development approach is used – the structure of the Buglione-Trudel matrix measures software development processes with more ease and less pain, because it measures what developers actually do. Agile developers communicate more openly; software developers in traditional environments better keep such information for themselves. Lean approaches rely on open communication, making important information available for the benefit of all.

The matrix can be used for effort estimation in all kind of projects, even in early planning stages, and before Story Items are actually known. Matrix-based estimations compare side-by-side with parametric estimations supporting contractual agreements; see *(ISBSG, 2010)*.

7.2 Limitations

The obvious limitation of the approach is: if the business driver profile is unknown, the team cannot decide which Story Items to choose for meeting customer's needs. Although that shouldn't happen, in reality it does.

7.3 Managerial Impact

Using the structured approach of the Buglione-Trudel matrix is expected to reduce management efforts rather than increase them. Agile projects have a high need for communication and measurement, and this can become very ineffective if methods are used such as presented in agile literature *(Cohn, 2005)*. The concept of the Convergence Gap is well understood by financial officers, thus helps keeping agile projects alive in times of crisis.

7.4 Future Steps

For small teams and small projects, the approach is feasible based on paper card tools alone; as often suggested in agile literature; however, Fig. 2 calls for a database solution. While it is possible to create such a database tool based on manually managed PowerPoint and Excel tools, a professional solution would allow using the Buglione-Trudel matrix for managing large and distributed agile software development projects.

8 Conclusion

Lean Six Sigma approaches ease the task and lower the pain when developing software using agile approaches. The development team, the sponsor, and management, can equally well assess at each point in time how far the development project is from meeting business requirements.

Although the approach originates from agile, lean concepts can be adapted to every kind of software development, including traditionally planned software development approaches and to software project estimation.

Agile software development offers significant advantages over traditional approaches. Relying and harvesting on developers intelligence, and embracing change, creates opportunities for becoming lean otherwise lost.

References

(Abran, 2009)	Abran, A. et. al. (2009), *The COSMIC Functional Size Measurement Method – V3.0.1 Measurement Manual*, COSMIC Consortium, Quebec, Canada
(Ambler, 2004)	Ambler, S.W. (2004), *The Object Primer*, 3rd Edition – Agile Model–Driven Development With UML 2.0, Cambridge University Press, New York, NY
(Beck, 2000)	Beck, K. (2000), *extreme programming explained*, Addison-Wesley, Boston, MA
(Buglione and Trudel, 2010)	Buglione, L., Trudel, S. (2010), "Guideline for sizing Agile projects with COSMIC". In: *Proceedings of the IWSM / MetriKon / Mensura 2010*, Stuttgart, Germany
(Creveling et. al., 2003)	Creveling, C.M., Slutsky, J.L., Antis, D. (2003), *Design for Six Sigma*, Prentice Hall, Princeton, NJ
(Cohn, 2005)	Cohn, M. (2005), *Agile estimating and planning*, Prentice Hall, Princeton, NJ
(Fehlmann, 2004)	Fehlmann, Th. (2004), "Six Sigma for Software", in: *Proceedings of the 1st SMEF Conference*, Rome, Italy
(Fehlmann, 2005)	Fehlmann, Th. (2005), "The Impact of Linear Algebra on QFD", in: *International Journal of Quality & Reliability Mgmt*, Vol. 21 No. 9, pp. 83-96, Emerald Group Publishing Ltd., Bradford, UK
(Roriz, 2009)	Roriz, H. (2009), "Can Scrum Support Six Sigma?", in: *Scrum Alliance*, http://www.scrumalliance.org/ articles/161-can-scrum-support-six-sigma [seen 26-Feb-11]
(Fuqua, 2003)	Fuqua, A.M. (2003), "Using Function Points in XP – Considerations", in: *Extreme Programming and Agile Processes in Software Engineering*, Springer, Lecture Notes in Computer Science, Vol. 2675 /2003, pp. 1013ff, Berlin, Germany
(George, 2010)	George, M. ed. (2010), *The Lean Six Sigma Guide to Doing More with Less*, John Wiley & Sons, Hoboken, NJ
(ISBSG, 2010)	Hill, P. ed. (2010), *Practical Software Project Estimation 3rd Edition*, McGraw-Hill, New York, NY
(Kressner, 2005)	Kressner, D. (2005), *Numerical Methods for General and Structured Eigenvalue Problems*, Springer, Lecture Notes in Computational Science and Engineering, Vol. 46, Berlin, Germany

(OMG UML, 2009)	OMG Unified Modeling Language™ (2009) – OMG UML, *Superstructure V2.2*, pp. 506--524, http://www.omg.org/spec/UML/2.2/Superstructure/PDF/ [seen 26-Feb-11]
(Poppendieck, 2007)	Poppendieck, M. & T. (2007), *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley Professional, Boston, MA
(Rule, 2010)	Rule G. (2010), *Sizing User Stories with the COSMIC FSM method*, http://www.smsexemplar.com/wp-content/uploads/ 20100408-COSMICstories-article-v0c1.pdf [seen 26-Feb-11]
(Santillo and Moretto, 2010)	Santillo, L., Moretto, G., (2010) on behalf of SBC (GUFPI-ISMA): *A general Taxonomy of Productivity Impact Factors*, Software Benchmarking Committee, Gruppo Utenti Function Point Italia – Italian Software Metrics Association, Roma, Italy
(Schwaber and Beedle, 2008)	Schwaber, K., Beedle, M. (2008), *Agile Software Development with Scrum*, Prentice Hall, Princeton, NJ