Using Agile Story Points as an Estimation Technique in CMMI Organizations

Alaa El-deen Hamouda
Systems and Computers Engineering, Al-Azhar University
Cairo, Egypt
Dr.alaa.hamouda@gmail.com

***Abstract*— Story Point is a relative measure heavily used for agile estimation of size. The team decides how big a point is, and based on that size, determines how many points each work item is. In many organizations, the use of story points for similar features can vary from team to another, and successfully, based on the teams' sizes, skillset and relative use of this tool. But in a CMMI organization, this technique demands a degree of consistency across teams for a more streamlined approach to solution delivery. This generates a challenge for CMMI organizations to adopt Agile in software estimation and planning. In this paper, a process and methodology that guarantees relativity in software sizing while using agile story points is introduced. The proposed process and methodology are applied in a CMMI company level three on different projects. By that, the story point is used on the level of the organization, not the project. Then, the performance of sizing process is measured to show a significant improvement in sizing accuracy after adopting the agile story point in CMMI organizations. To complete the estimation cycle, an improvement in effort estimation dependent on story point is also introduced, and its performance effect is measured.**

Keywords—CMMI; Agile; Story Point; Software Estimation

I. INTRODUCTION

Size is the primary input to many models used to estimate effort, cost, and schedule. The estimates should be consistent with project requirements to determine the project's effort, cost, and schedule. A relative level of difficulty or complexity should be assigned for each size attribute. It is important to use the appropriate methods to determine the attributes of the work products and tasks that will be used to estimate the resource requirements. Methods for determining size and complexity should be based on validated models or historical data [1, 2].

Story Points is a relative measure heavily used for agile estimation of size. The team decides how big a point is, and based on that size, determines how many points each work item is. Simply, look at representative work items, give the smallest the size of one point, and then go through all other work items and give them a relative point estimate based on that point. One project team may say that a work item of a certain size is worth 1 point. Another project team would estimate the same sized work item to be worth 5 points. That is fine, as long as they are consistent within the same project. In CMMI organizations, such difference is not acceptable. The projects sizes must be relative. If one project team says that a work item of a certain size is worth 1 point. Another project team should estimate the same sized work item to be worth 1 point. This generates a challenge for CMMI organizations to adopt Agile in software estimation and planning.

Although a lot of researches were introduced to adopt Agile in CMMI organizations [3, 4, 5, 6, 7], very little work addresses this problem of estimation. Considering that agile story points –in its original form- does not satisfy the CMMI requirements, some of researches proposed using estimation techniques different from agile story points.

For example, [8] proposed using Common Software Measurement International Consortium; COSMIC as functional size measurement instead of agile story point. [9] proposed using use case point as estimation technique, breaking the use cases into a set of user stories and then estimating the user stories in story point. However, with such techniques, the benefits of story points [10] widely used in agile development will be lost.

Through this paper, a methodology that adopts agile story point in CMMI organizations is proposed. It includes multiple effective techniques and allows a CMMI organization to start seeing that consistency right away and receive maximum benefits in their streamlines agile approach to estimation and story point evaluation.

This paper is organized as follows: Section II defines the problem to be addressed, then comes Section III, describing the proposed methodology. After that, Section IV shows the implementation and evaluation of applying the proposed methodology. Finally, Section V concludes with the results.

II. PROBLEM DEFINITION

Story point is a relative measurement of how difficult a task is. This is because humans are actually better at relative estimates than precise measurements. This is the philosophy of story points [10, 11].

In agile story point method, the estimator –usually developer- takes about two tasks on the project and assigns them two

978-0-7695-5222-4/14 $31.00 © 2014 IEEE
DOI 10.1109/AGILE.2014.11

different story point values. Then he estimates the other tasks using those two story point approximations as a basis for the next estimate. For example, task C is not much harder than task A but incredibly easier than task B. So, it is only about two story points more work than task A.

Agile story point overcomes the problems of other model-based techniques. A fundamental problem in estimation with such techniques like use case point and function point is that the estimate cannot be arrived at until all of the use cases / logical files are written. Writing user goal use cases / logical files is a significant effort that can represent 10–20% of the overall effort of the project. This investment delays the point at which the team can create a release plan. More important, if all the use cases / logical files are all written up front, there is no learning based on working software during this period [12].

As well, use cases are large units of work to be used in planning a system. While use case points may work for creating a rough initial estimate of overall project size, they are much less useful in driving the iteration-to-iteration work of a team, which is a crucial aspect in agile development [12]. For function point, gaining proficiency is not easy; the learning curve is quite long. The method is also quite time-consuming thus might result costly.

For that, the agile story points is en efficient method to measure the software size by analogy [13, 14]. It is highly recommended, due to its features in terms of applicability over software domains and project phases, in agile development, iterative approaches, and accurate and standardized measurement results [9, 15].

However, the unit of measurement that story points possibly represent is considered arbitrary, and their values are subjective. They are not derived from a model. Therefore they cannot be transferred outside the business environment in which they are produced, which is the single project team. In other words, story points' validity as a measurement method is not ensured, and their accuracy, precision, and comparability can vary significantly [2].

From a process improvement perspective, e.g. within the "CMMI for Dev. 1.3" model [1], there must be a consistent estimation technique all over the organization. This is required to institutionalize at Capability Level 3 (Organization Level) several Process Areas such as Project Planning, Project Monitoring and Control, and Measurement and Analysis, which would otherwise remain at Capability Level 2 (Project Level) [3].

The agile story point method, in conjunction with the CMMI-DEV, will allow for a governance of software development processes and projects suitable for any organizational context and size, including medium to large ones, when "rapid" methodologies are applied as well. This results in benefits for both project management and process improvement practices.

The advantages of applying agile story point method are not only related to CMMI organizations, but are also related to most medium to large organizations that need to have one scale for all projects. This is so important for measurement data of the project to be compared with (and contribute to) benchmarking data of the other projects. As well, the management will benefit from a uniform measurement model across different projects [8]. All of that encourage the development of agile story points technique suitable for such organizations.

III. PROPOSED METHODOLOGY

We introduce a methodology that achieves the following objectives:

1- To have a reference library of requirements. The new requirements will be estimated according to this reference library
2- To consider the technical and environmental factors that affect the different sizes between projects
3- To have a measure of productivity factor to have a uniform scale for all projects

Thus, the proposed methodology has the following steps:

A. Building the reference library
B. Sizing using the reference Library
C. Making Effort Estimation using productivity factor (PF)
D. Updating the requirements size if needed during the project
E. Updating the PF and the reference library at the end of the project

More details about each step are illustrated below.

A. Building the reference library

To build the reference library, the following steps are applied.

1. Select some software projects developed previously in the organization. The selected projects should cover the different business domains of the developed software inside the organization.
2. Collect historical data (requirements, spent efforts) from these projects.
3. Assume the productivity factor of the organization (average of all teams' velocities) if there is no PF was calculated before.
4. Categorize the requirements into seven groups according to the spent efforts. The spent efforts include the phases of analysis, design, implementation, and testing. The groups have efforts equal to the assumed productivity

factor (team velocity) multiplied by the Fibonacci values (1, 2, 3, 5, 8, 13, and 21). For example, assuming the productivity factor is 6 hours/ story point, the groups' efforts are 6, 12, 18, 30, 48, 78, and 126 hours. Three to four requirements in each category (group) is recommended.

5. A variance of +10% to -10% in efforts is accepted. For example, a requirement of 52 hours can be added to the fifth group (48 story hours)
6. Write down the story point for the selected requirements in each category

B. Sizing using the reference Library

All new projects use this library as a frame of reference when estimating new stories. This can be done at the project start and/or at the sprint/iteration start. The following procedure of sizing is followed.

1. For each requirement in the new project, the author (developer responsible for implementing the requirement), estimate the size in story point
2. For large requirements, they should be broken into smaller requirements (the greatest should not need more than three weeks, which is about 126 hours)
3. Identify team to do the estimation. It includes:
    a. Expert estimators (2-4 Field Experts recommended provided that the number of estimators are even. By that, the total number of estimators and author will be odd)
    b. Author
    c. Estimation coordinator
4. Author presents the requirement to be estimated
5. Author and experts jointly identify tasks & assumptions
6. Independently, each estimator writes a value (1, 2, 3, 5, 8, 13, or 21) in a piece of paper according to its analogy to the historical requirements data base. For example, this story is like that story in the database (maybe a little more difficult, maybe a little less). By that, estimation is performed against multiple stories of the same effort and each story is given a comparable estimated value. This size is called the Unadjusted Story Point (USP).
7. Each estimator writes values for the Technical Complexity Factors; TCF. These are the same factors used in Use Case Point (UCP) estimation technique [16]. Technical factors describe the expectations of the users for the delivered software. Generally, it is an assessment of non-functional requirements, table 1. All factors do not have the same potential impact on a project size estimate, so each factor has a multiplier, representing the relative weights of the factors. For each of the thirteen technical factors, a relative magnitude of 0 to 5 is assigned. They represent a continuum of effort / difficulty. Those (0-5) values are then multiplied by the multiplier for each factor. For example, a relative magnitude of 3 for cross-platform support would result in 6 points – because cross-platform support has twice the impact on work size as a focus on response time. The TCF is calculated first by summing up the relative magnitudes (multiplied by the multipliers for each factor) using the equation:

$$TCF = 0.6 + (0.01 * \sum \text{relative magnitudes}) \quad (1)$$

For example, if the relative magnitude of every technical factor were 2, the adjusted sum would be 28. The TCF would then be 0.6 + 0.28 = 0.88.

Table 1: Technical Complexity Factors

Metric	Description	Weight	Value
T1	Distributed System	2	
T2	Response or throughput performance objectives	1	
T3	End user efficiency (online)	1	
T4	Complex internal processing	1	
T5	Code must be re-usable	1	
T6	Easy to install	0.5	
T7	Easy to use	0.5	
T8	Portable	2	
T9	Easy to change	1	
T10	Concurrent	1	
T11	Include special security features	1	
T12	Provide direct access for third parties	1	
T13	Special user training facilities are required	1	

8. Each estimator writes values for the *Environmental Complexity Factors (ECF),* the same factors used in Use Case Point (UCP) estimation technique [16]. *They are* representation of the capability of the team and the environment in which the software is being developed. There are eight factors, table 2. Factors 7 and 8 are both *negative numbers* – so higher numbers have the reverse impact as with other factors. The estimators determine the value that characterizes each factor, from 0 to 5. Then multiply by the multipliers and add them up. The environmental complexity factor is calculated by the equation:

$$ECF = 1.4 – (0.03 * total) \quad (2)$$

The higher the identified sum is, the lower the environmental factor. This is different than calculating the technical factor, but it makes sense – the more experienced the developers are, the less effort they will spend on a given project.

9. Each estimator hand over his estimation to the coordinator. This confidentiality is important to avoid following a certain estimator.
10. The coordinator distributes the list to all experts
11. The different estimator (higher or lower) is asked to explain why he gave his value

Table 2: Environmental Complexity Factors

Metric	*Description*	*Weight*	*Value*
F1	Familiar with Rational Unified Process	1.5	
F2	Application experience	0.5	
F3	Object-oriented experience	1	
F4	Lead analyst capability	0.5	
F5	Motivation	1	
F6	Stable requirements	2	
F7	Part-time workers	-1	
F8	Difficult programming language	-1	

12. All values are discussed:
    a. If all agree on a certain value, it is ok
    b. If someone insists on his different value, the value that most agreed on should be selected
13. Back to step 4; continue till estimating all requirements
14. The project manager calculates the Adjusted Story Point (ASP) using the following formula:

ASP = USP * TCF * ECF (3)

It is important to note that when many experts independently come to the same estimate based on the same assumptions, the estimate is likely to be correct. It is the coordinator's responsibility to make sure that the experts are working on the same and correct assumptions.

C. *Making Effort Estimation using PF*

The estimated size is delivered to the project manager who calculates the efforts. The estimated efforts are calculated by multiplying the estimated size (ASP) by the PF. The team does not know about the details of the effort calculation. The estimated efforts can then be used for project schedule planning.

D. *Updating the requirements size if needed during the project*

For example, after implementing some requirements in the first iteration/sprint, the requirements of the second iteration/sprint may become clearer and then the size may need to be updated.

E. *Updating the PF and reference library*

At the project end**,**

- Repeat step 2 (sizing using the reference library) to be sure of the right requirements sizes after developing them and hence understanding them clearly.
- Update the PF using the following formula

PF = Actual effort for all requirements/ the total of all updated requirements sizes

- The project team suggests the new requirements that are with new types like (e.g. data validation...) to the reference library categorized by the sizes 1, 2, 3, 5…

Every six months, form a committee from all projects/business domains to update the reference library by reviewing the requirements of each type to be sure the library is up to date with new emerging stories which may not have similar one in the library.

IV. Implementation and Evaluation

The proposed methodology was applied on a CMMI Level 3 Company. The company has got CMMI Level 3 since 2006 and was reappraised successfully in 2009 and 2012. It has several projects for developing On-The-Shelf products and others for developing customized projects. It used Use Case Point (UCP) estimation technique [17, 18] for two years. Then, they found significant deviation between the estimated size at the start of the project/iteration and the actual size (the reviewed size at the project/iteration end). This motivated the Engineering Process Group to look for another technique of estimation. They selected the Quick Function Point (QFP) [19 and 20]. They made it optional for each project manager to decide to use UCP or QFP according to some criteria. After four years of using the QFP and UCP on about 20 projects, they found the deviation in sizing was about 28%. This deviation was determined by the resizing percentage; actual size / estimated size. Also more than 40% of requirements were mistakenly sized. There was a difference between the estimated size at the project start and the corresponding reviewed size at the project end. So, they faced late project delivery issues.

To fix these problems, they decided to take rapid development approach through Agile adoption. As well, they looked for another estimation technique that depends on analogy. They thought of agile story point but they faced the problem of working on the organization level, as required by CMMI, not the project level as in agile story point. We proposed our methodology. They applied it on different projects as follows:

1. First, from the historical data, the project manager (PM) selected different requirements from different old projects with different teams. The selected requirements had an estimated effort equal to the actual effort performed. By

that, he guaranteed there were no special circumstances in the selected requirements.

2. The PM assumed the productivity factor (team velocity) with 6 hours / story point. That was due to the fact that the greatest requirement should not take more than three weeks (21 SP *6 hour /SP =126 hours). Otherwise, it should be divided into smaller requirements. The PM didn't inform his team of this assumption. Using the Fibonacci values 1, 2, 3, 5, 8, 13, and 21, he categorized the requirements according to their efforts to groups of 6, 12, 18, 30, 48, 78, and 126 hours.
3. He asked the developer of each selected requirement to write the story board if it was not written clearly before. By that, he developed the reference library.
4. He started a new project and formed his team. Then, the team made size estimation of the new requirements using the reference library. Some team developers faced some challenges in finding reference requirements similar to their requirements. This was solved by giving more explanation of the story boards of the requirements in the reference requirements database. No one –except the project manager- knew how the sizes were transferred to efforts. This was to avoid thinking in efforts and time during sizing process.
5. At the end of the project, the requirement became clearer. The sizing process was done again to determine the real size, called actual size.
6. The PM updated the PF and the team suggested the requirements to be added to the library.
7. The same methodology was applied on four projects.

The team was five developers and three testers in the first project, four developers and two testers in the second, three developers and one tester in the third and fourth. The organization had a lot of projects running concurrently and the engineers were distributed over them. The duration of the four projects was four months, four months, three months, and two months respectively. The estimated sizes at the project start and at the project end for all requirements in the four projects are illustrated in table 3, 4, 5, and 6. The illustrated sizes are in unadjusted story points. This is due to the fact that there was no change in the TCF and ECF at the project start and at the project end.

From the tables of requirements and sizes, it is apparent that three out of eleven requirements in the first project, one out of twelve in the second, no one in the third, and four out of eighteen in the fourth had differences in sizing at the project start and project end. By averaging these values, table 7, we have a difference percentage of 14% which is a significant improvement over the previous one (using other estimation techniques) of 40%. Also, the actual difference between the estimated size at the project start and project end were calculated for all projects, table 8. The error percentage was calculated and averaged regardless the sizes at the project end was higher or lower than that of the project start. It is clear that the error percentage became 5.9%, which was a great improvement over the previous one of 28% when other techniques, e.g. UCP or QFP, were used.

Table 3: Project 1 requirements and sizes

Req. #	Estim. at the start	At the end
1	21	21
2	21	13
3	21	13
4	8	8
5	3	3
6	3	3
7	5	8
8	5	5
9	21	21
10	21	21
11	21	21

Table 4: Project 2 requirements and sizes

Req. #	Estim. at the start	At the end
1	21	21
2	21	21
3	13	13
4	8	21
5	13	13
6	21	21
7	3	3
8	1	1
9	1	1
10	2	2
11	21	21
12	21	21

Table 5: Project 3 requirements and sizes

Req. #	Estim. at the start	At the end
1	8	8
2	3	3
3	21	21
4	21	21
5	13	13
6	8	8
7	8	8

Table 6: Project 4 requirements and sizes

Req. #	Estim. at the start	At the end
1	5	5
2	1	1
3	21	13
4	1	1
5	2	2
6	3	3
7	3	3
8	1	1
10	3	3
11	1	1
12	1	1
13	5	5
14	1	1
15	5	5
16	1	5
17	1	5
18	8	5

Table 7: The requirements with difference in sizing

	No. of Req.	No. of Req. with size difference	Error %
Project 1	11	3	27%
Project 2	12	1	8%
Project 3	8	0	0%
Project 4	18	4	22%
		Average	14%

Table 8: Sizes at project start and end

	Size at start	Size at the end	Error %
Project 1	150	137	8.7%
Project 2	146	159	8.9%
Project 3	95	95	0.0%
Project 4	66	63	4.5%
		Average	5.9%

The actual efforts in each project were used to update the productivity factor with the following equation:

PF = Actual effort for all requirements/ total of all updated requirements sizes (sizes estimated at the project end)

The PF was updated to be 8.7 instead of the assumed value of six hours per one story point. This affects the estimation of new projects.

It is clear that the proposed methodology used the agile story points and adjusted the values using the environmental factors and the system complexity factors that cannot be considered in the estimation of relativeness. It also incorporated the concept of productivity factors. Both of that made it a reliable estimation technique in CMMI organizations with Agile adoption.

V. Improving the Effort Estimation Stage

In Step (C) 'making effort estimation', it is assumed that the relationship between the effort and size is linear. It is compliant with Karner's assumption [21]. However, this assumption does not reflect the actual situation in the practical software industry. For instance, if the effort required to build a software project of size 500 SP is 4,000 person-hours, the effort needed to build the same project type of size 2000 SP would be more than 16,000 person-hours. The second issue is that in equation 3, the effort is a function of the adjusted story point (ASP) size. The ASP encompasses the non-functional requirements of the system and it may increase the original unadjusted story point by 30%. This is due to the fact that the maximum and minimum value of the TCF is 1.3 and 0.6 respectively as deduced from equation 1 [21]. By that, the value of the adjusted story points (ASP) will be 30% more or less than the unadjusted story points. However, IBM states that non-functional requirements might represent more than 50% of the total effort [22]. This means that the non-functional requirements may virtually increase the unadjusted size by 100%.

To tackle these shortcomings, we present a regression model to adjust the productivity factor in the proposed methodology, and thus, enhance the estimation. We use multi-layer neural networks. The inputs are the story points, technical complexity factors and the environmental complexity factors, figure 1. The same projects and requirements in the previous section are used after removing the outliers to learn the neural network. The technical complexity factors are calculated using the equation:

$$Technical\ Complexity\ Factors = \sum_{i=1}^{13}(CF_i * W_i)$$

Where CF_i is the complexity factor and W_i is its weight as in table 1. Similarly, the environmental complexity factors are calculated using the equation:

$$Envienomental\ Complexity\ Factors = \sum_{i=1}^{8}(EF_i * W_i)$$

Where EF_i is the environmental factor and W_i is its weight as in table 2.

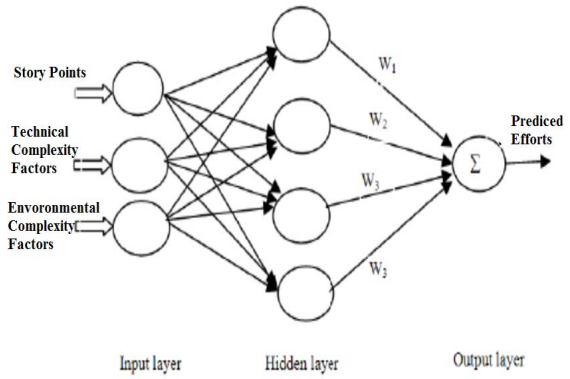

Figure 1: Effort Estimation using story points, complexity and environmental factors

Magnitude of Error Relative to the Estimate (MER) has been used as an evaluation method for the accuracy of the effort estimation. MER has been calculated using the equation:

$$MER = \frac{|\text{Actual Effort} - \text{Predicted Effort}|}{\text{Predicted Effort}}$$

Results show that the proposed machine learning model using neural network outperforms the effort estimation using linear equation of step (C). The MER decreases from 34% to 12%, which is considered a significant improvement.

By that, our complete proposed methodology decreases the size estimation error from 28% to 5.9%, and the effort estimation from 34% to 12%.

VI. Conclusion

Analogous estimation is successful due to our inherent ability to better measure relative size than absolute size. Agile story point is en efficient analogy based software estimation. However, it is dependent on the team' sizes, skillset and relative use of story point. So, it was used on the project level. This caused a considerable challenge to CMMI organization that needs organization level estimation technique. For that, a comprehensive methodology is proposed to use story point in CMMI organizations. This methodology makes it possible to establish an organizational level estimation and add the relativity concept of the story points. It also adjusts the agile story point values using the environmental factors and the system complexity factors that cannot be considered in the estimation of relativeness. Then, the concept of productivity factors is incorporated.

After applying the proposed methodology in a CMMI organization with several projects in different domains, the estimation accuracy improves significantly and the size estimation error decreases from 28% to 5.9%.

As well, the Karner's model of the relation between effort and size has some limitations since it assumes that the relationship between software effort and size is linear. Furthermore, it underestimates the influence of nonfunctional requirements on software effort. To tackle these limitations, a machine learning model has been introduced. It decreases the effort estimation error from 34% to 12%.

Acknowledgment

I would like to thank the Horizons Software Company that provided some automated tools to improve the effectiveness of this work.

References

[1] SEI, CMMI for Development, Version 1.3, CMMI Product Team, November 2010.

[2] C. F. Kemerer, An empirical validation of software cost estimation models, Communications of the ACM, Vol. 30, No. 5, pp. 416-429, 1987.

[3] SEI, CMMI or Agile: Why Not Embrace Both!, Technical Note CMU/SEI-2008-TN-003, Software Engineering Institute, November 2008.

[4] Neil Potter and Mary Sakry, Implementing Scrum (Agile) and CMMI Together, The process group, Vol. 16, No 2. March 2009.

[5] Fritzsche and Patrick Keil, Agile Methods and CMMI: Compatibility or Conflict?, e-Informatica Software Engineering Journal, Vol. 1, No. 1, 2007.

[6] Jessica Diaz and et al, Mapping CMMI Level 2 to Scrum Practices: An Experience Report, EuroSPI, CCIS 42, pp. 93–104, 2009.

[7] Carsten Jakobsen and Kent Johnson, Mature Agile with a twist of CMMI, IEEE Agile Conference, 2008.

[8] COSMIC-based Project Management in Agile Software Development and Mapping onto related CMMI-DEV Process Areas, Proceeding of the joined international conferences on Software Measurement, Novemebr 2010, Germany.

[9] Mike Cohn, Agile Estimating and Planning, Addison-Wesley, 2005.

[10] Evita Coelho and Anirban Basu, Effort Estimation in Agile Software Development using Story Points, International Journal of Applied Information Systems 3(7):7-10, August 2012. Published by Foundation of Computer Science, New York, USA.

[11] Charles R. Symons, Function Point Analysis: Difficulties and Improvements, IEEE Transactions on Software Engineering, VOL. 14, NO. I , JANUARY 1988.

[12] Mike Cohn, User Stories Applied for Agile Software, Prentice Hall, 2006.

[13] Eduardo Miranda and et al, Sizing user stories using paired comparisons, Journal of Information and Software Technology Vol. 51, No. 9, September, 2009 Butterworth-Heinemann Newton, MA, USA.

[14] E. Kocaguneli, T. Menzies, A. B. Bener and J. W. Keung, "Exploiting the Essential Assumptions of Analogy-Based Effort Estimation," IEEE Transactions on Software Engineering, vol. 38, pp. 425-438, 2012.

[15] M. Cohn, Tutorial on Agile Estimating and Planning, Agile 2006 Conference, 2008, Minneapolis.

[16] G. Karner, "Resource Estimation for Objectory Projects" *Objective Systems,* 1993.

[17] Ribu, Kirsten, Estimating Object-Oriented Software Projects with Use Cases, Master of Science Thesis, University of Oslo, Department of Informatics, 2001.

[18] Schneider, Geri and Jason P. Winters, "Applying Use Cases: A Practical Guide", Addison Wesley, 1998.

[19] IFPUG: Function Point Counting Practices Manual, Release 4.3.1, International Function Point Users Group, www.ifpug.org, 2010.

[20] C.A. Behrens, 'Measuring the Productivity of Computer Systems Development Activities with Function Points', IEEE Transactions on Software Engineering, pp. 648-652, November 1983.

[21] Ali Bou Nassif, Luiz Fernando Capretz and Danny Ho, Estimating Software Effort Using an ANN Model Based on Use Case Points, 11th International Conference on Machine Learning and Applications, 2012.

[22] Y. Ossia, IBM haifa research lab. *IBM Haifa Research Lab* [Online], 2011. Available: https://www.research.ibm.com/haifa/projects/software/nfr/index.html.