Estimating Agile Iterations by Extending Function Point Analysis

A. Udayan Banerjee, B. Kanakalata Narayanan, and C. Mahadevan P
NIIT Technologies Ltd, No.31/2, Rupena Agarhara, Hosur Main Road, Bangalore-560068, India

Abstract - *Estimation is critical to software development irrespective of the development methods being used. Waterfall methods work on the concept of signed off requirements, while agile methods are designed to handle changing requirements through increased customer participation and frequent releases. Statistical estimation methods like Function Point Analysis (FPA) are more appropriate in scenarios where requirements are explicitly documented, while agile projects are typically estimated using analogous non standard sizing methods like Story Points.*

Organizations outsourcing software development want vendors to adopt agile methods but estimate using standard techniques like FPA and upfront commit to schedules, features, effort. The key characteristic of agile projects which impacts estimation is its iterative incremental lifecycle which includes evolutionary design, requirement refinement, increasing code base and constant code refactoring. In this paper we propose a mechanism for estimating the size of agile iterations in Function Points by extending FPA [11] techniques along with Caper Jones activity scope for software projects [17, 20]. We have applied this technique on three agile projects and observed that there is a linear correlation between effort consumed and the estimated iteration size.

Keywords: Agile, Function Points, Estimation, Iterative Development, Outsourcing.

1 Introduction

As an answer to the challenges of modern software development, different lightweight approaches have been established since the mid 1990s that can be subsumed under the brand Agile Methods [1-2]. They "allow for creativity and responsiveness to changing conditions" [3]. They also emphasize on customer participation, quick reaction to requirements' changes and continuous releases. These methodologies are gaining in popularity as preferred means for developing software as they allow organizations to deliver software effectively in a changing environment [4].

Agile methods specify that working code should be delivered in small pieces iteratively catering to a sub set of the functionality asked for by the user. With every iteration, users are encouraged to provide feedback, add, remove, change requirements based on which the subsequent code is refined and incremented. The main idea behind this approach is that through emphasis on working code delivered frequently there is a greater chance of delivering usable software which provides business value. [4]

Software estimation is a critical component of software development, irrespective of the development method being adopted. Estimation defines the transformation of requirements, skills, people and equipment into cost and effort [5]. The main software estimation techniques are the following:

- Analogy based: where a new project is estimated based on its resemblance to an existing project,
- Expert opinion: where a group of experts gather together to come to a consensus on how much time is required to build a piece of software,
- Lines of code based: where the estimate is arrived at based on the expected lines of code,
- Bottom up methods like work breakdown structure where each task required for the project is estimated and the sum of it is the total effort for the project,
- Statistical methods like Function Point Analysis which quantify the size of the software rather than estimate the effort directly. These methods use metrics collected from past projects along with mathematical formulae to estimate project costs.

Each method has its advantages and disadvantages which are well researched and documented. Statistical methods offer a scientific approach to software estimation, as compared to the other methods, which are more subjective in nature, with the exception of lines of code based sizing. These methods are more preferred when software development is outsourced by organizations to vendors mainly because they help in quantifying software in a standard way irrespective of the technology being used and such estimations can be independently verified. The main disadvantage of statistical methods is that they require the specifications to be articulated in a detailed manner to provide accurate estimates [6]. Agile development projects on the other hand are characterized by fuzzy or evolving requirements. They are typically estimated using a combination of analogous methods along with expert opinion. Agile processes recommend that estimations are best done by the team executing the project, and revisited each iteration, using a sizing metric evolved by the team. Story Points or Ideal days [7] are most popularly used in this respect. The team looks to past projects or iterations, and draws on its own experiences to produce estimates [7,8]. Caper Jones [9] states that one of the agile weaknesses is a widespread failure to measure projects using standard metrics such as function points.

The purpose of this paper is to propose a mechanism for calculating the size of agile iterations as Function Points accommodating for iterative incremental development by extending Function Point Analysis [10] techniques along with Caper Jones activity scope for software projects [11,12]. While there have been papers published trying to establish a theoretical relationship between Function Points and Story Points [13, 14], we have gone one step further and tried to establish a working model for sizing agile iterations using FPA, in the outsourcing context. Through this we present a standard and consistent approach to sizing agile iterations.

2 Estimating Agile Projects with Story Points

In agile projects the features to be developed are expressed in the form of user stories [15] and one of the popular methods of sizing stories is using Story Points, a subjective unit of estimate derived by agile teams. In this method a team compares a user story to one or more similar stories and gives the story a size in terms of 'Story Points' or 'Ideal Person days'. The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story [13]. Each team defines story points as they see fit. One team may decide to define a story point as an ideal day of work and another team may define a story point as a measure of the complexity of the story [13]. Story points have emerged as industry best practice for measuring an agile development team's velocity i.e. the number of user stories delivered in an iteration [7].

The stories that may be taken up by a team in an iteration is dependent on the experience of the team, the cohesiveness of the team, the knowledge they have on the product, the technology complexity involved etc. These numbers vary from team to team. Michael Cohn [16] says that it is very difficult to establish a direct correlation between story points and hours and further says that the relationship between story points and hours is typically a distribution centered on a mean (Figure 1). Even for a given team, same story point sized stories may take different times during different points in the release life cycle. A team which has been working for a long time on a specific product may be able to deliver more stories than a newly formed team [13]. Thus while story points provide a way for agile teams to flexibly estimate user stories, it is not always possible to extend these metrics across teams or at an organization level [17].

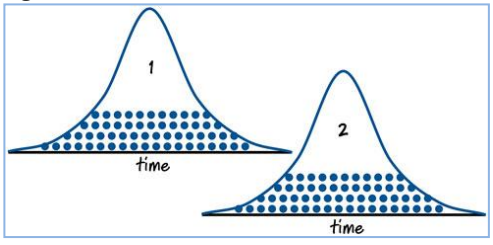

Figure 1: Hours to develop a 1 and 2 point story [16]

3 Challenges of Estimating Iteration Size Using Statistical Methods

In this section we attempt to show how the very nature of iterative development makes it difficult to apply traditional statistical methods like FPA to estimate projects. Each agile iteration is a mix of new stories, refactoring of existing stories, testing and bug fixes to existing stories. It is not executed like a mini waterfall based project where a requirement is executed, completed and signed off with a fixed schedule in a single iteration.

One of the initial activities done at the start of a project or release is Release Planning, where the development team along with the customer or Product owner get together to understand the requirements (product backlog) and estimate roughly the size of the requirements and the number of iterations it will take to fulfill them. Team commits to a probable release date and the best and worst case list of features that may be released by the release date. The team may use Story Points to size the release or use statistical methods like Function Points depending on the extent of clarity they have regarding the requirements.

Iteration Planning and the Definition of Done: Every iteration, the team commits on the number of stories that can be accommodated. While agreeing to develop stories, the team formulates a 'Definition of Done' (DoD) which is a list of activities that will be performed by the team in that iteration towards the selected stories. DoD is a simple list of activities (writing code, coding comments, unit testing, integration testing, release notes, design documents, etc.) that add verifiable/demonstrable value to the product and can be undertaken in an iteration [18]. The team may formulate a definition of done for the release which is a super set of the DoD for an iteration. For example a team may decide that they would leave integration and stress testing (which is in scope for the release) for later iterations and only take up unit testing and functional testing on each story in the current iteration. Based on the activities pending, teams may later visit user stories which where were developed in initial iterations to complete and polish them the extent needed to make a formal release.

Hence, the key problem in estimating the size and the effort required in an iteration using FPA is that all activities required to be completed towards a function or user story in a project lifecycle may not necessarily be taken up in a single iteration. The same story may be visited several times over subsequent iterations either to refine it or to add more complexity or undertake additional activities like end to end testing.

Change Management in agile Projects: In order to accommodate change, agile methods recommend that **t**he customer or product owner is able to re-prioritize stories, introduce additional complexity or add new stories into the product backlog. During the iteration planning meeting the team is expected to understand the new prioritized backlog

and decide which items they can take up in the iteration as per the DoD. Hence in an agile project life cycle estimation is something that frequently occurs and is continuously revised and updated.

Evolutionary Design: While agile teams deliver working code in each sprint, they do not generally have a well defined design phase as in waterfall based projects and instead work around evolutionary design practices. Most teams start by having a basic working design which is refined and re-factored as the project progresses. Sometimes more than one design alternative may be tried out, which could cause significant code changes to user stories which are already developed in early iterations [4]. Another reason for design or code refactoring may be driven by business or regulatory needs which may demand adjustments in delivered code. Again this is an aspect affecting estimation of an iteration.

Testing and Bug Fixes: In agile projects the code delivered in a iteration is tested by the product owner and bugs may be recorded which are typically taken up by project teams in subsequent iterations. Most agile teams have dedicated iterations where they do not accommodate any new user stories, but only work on refining and bug fixing on existing user stories in order to make them release worthy. This again adds another dimension to the estimation process.

Complexity is handled by most agile teams iteratively. So a complex user story may be broken up to show a simple working code which is iteratively enhanced with each iteration. An example of this is providing a multi language capable website. In early iterations a single language page may be developed, which is subsequently enhanced and tested for multiple languages..

Hence, upfront committing to a function point count to be delivered in a iteration is very difficult for an team as all these varied dimensions are also to be considered.

4 Estimation in the OutSourcing Context

Outsourcing with off-shoring software projects is a popular trend in organizations whose core activity is not developing software with the chief motive being cost reduction [19]. Most companies outsourcing software use competitive methods to request quotes from vendors and chose the vendor who most closely meets their expectations in terms of cost, quality, skill levels etc. The pre-dominant estimation method preferred in an outsourced scenario is Function Point Analysis (FPA) with the development process being waterfall based and vendors working on the concept of formal signed off requirements.

FPA is very popular when development work is outsourced because it is a standard technique [10] and is considered a scientific approach to sizing software and an absolute metric which can be computed, irrespective of the team executing the project. The organizational productivity benchmark typically computed in hours per function point (technology specific) is applied to the Function Point count to arrive at the effort and the schedule for a project. It enables companies to verify if the vendor estimates are realistic and within accepted range.

4.1 Agile Estimation in an Outsourcing Context

With agile development methods gaining popularity organizations want to realize the benefits of such methods while continuing with the trend of outsourcing. Organizations have started expecting their vendors to execute projects using agile development techniques.

Given their variable nature it is difficult to fix scope, budget and schedule in agile projects. It is recommended to work with a fixed budget or schedule keeping the scope variable or within a range of possible features that could be delivered [20]. However, these techniques work well only if the projects are in-house developments of the company concerned or if there is a high degree of trust between a customer and a software vendor. In a competitive situation, organizations expect vendor companies to provide an upfront effort estimate and commit to schedules, features and resources even while developing projects the agile way. These estimates may have to be done during contracting and much before a team is assembled to execute the project. Since story points are not counted by scientific methods, they are not accepted as a credible estimation technique in this context and customers typically want function point based estimates.

Offshore vendors assemble bid teams consisting of representative members to help in creating estimates for agile projects during contracting stage either using Function Point Analysis or work breakdown structure or other suitable methods.. Such estimates are at a macro level and may vary significantly once the project execution commences and micro level details are obtained.

A common problem during project execution is the expectation on the team to produce iteration level estimates. It is possible that members of a team assembled for a project and may not have worked with one another before or may not be ones with similar experience or may not have worked in the problem domain or technology [4]. This problem is further complicated when there are multiple agile teams executing a large project. So teams may not be skilled enough to produce estimates and with the absence of organizational metrics or a standard way to estimate the iterative development process, it is very difficult to size agile iterations with team inputs.

Another aspect that agile methods implicitly assume is an atmosphere of trust. They assume that developers truthfully estimate for stories and as iterations progress, take up more and more stories and increase their pace of working and become better at estimating [21, 22]. Secondly customers or product owners are expected to believe in the subjective developer estimates and give the team enough time to achieve a predictable pace of working and delivering user stories.

This is again a difficult situation in a competitive vendor customer relationship.

5 Proposed Solution

Since function point based counts are standard and accepted we propose in this paper to extend the techniques of function point analysis to help in estimating agile iterations. The key considerations being

- Accommodate for increasing complexity and iterative design
- Accommodate for the same story to be worked upon in multiple iterations based on the Definition of Done

We have used FPA method of calculating project complexity using a value adjustment factor [10,23] and the Caper Jones suggested activity scope percentage for software projects [11, 12] to quantify the size of the stories taken up in agile project iteration in the form of Function Points. Through this we propose to provide a scientific basis for computing the size of an iteration, such that the estimates can be verified, validated and defended.

5.1 Function Points and General System Characteristics

Function Points are counted in a two step process. The first step is to classify each feature in the system against one of the major functions i.e. External Inputs / External Outputs / External Queries / Internal Logical Files / External Interfaces and arrive at the function point count, which is called the raw or unadjusted function point [10], [23]. The raw function points are adjusted by computing a value adjustment factor based on the possible impact of a set of fourteen general system characteristics (GSC) of the system to be developed. These factors are listed in for reference in Table 1 [10, 23, 24]

GSC	Description
Data Communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
Distributed Data Processing	How are distributed data and processing functions handled?
Performance	Was response time or throughput required by the user?
Heavily Used Configuration	How heavily used is the current hardware platform where the application will be executed?
Transaction Rate	How frequently are transactions executed daily, weekly, monthly, etc.?
On-line Data Entry	What percentage of the information is entered On-Line?
End -User Efficiency	Was the application designed for end-user efficiency?
On-line Update	How many ILF's are updated by On-Line transaction?
Complex Processing	Does the application have extensive logical or mathematical processing?
Reusability	Was the application developed to meet one or many user's needs?
Installation Ease	How difficult is conversion and installation?
Operational Ease	How effective and/or automated are start-up, back-up, and recovery procedures?
Multiple Sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
Facilitate Change	Was the application specifically designed, developed, and supported to facilitate change?

Table 1: General System Characteristics

Each characteristic has associated descriptions that help determine the degrees of influence of the characteristics. The degree of influence ranges on a scale of zero to five, from no influence to strong influence. The IFPUG Manual provides detailed evaluation criteria for each GSC [10]. The GSC is scored based on their influence on the system being counted and this provides the value adjustment factor. The unadjusted Function Point count is multiplied by the value adjustment factor to arrive at the Adjusted Function Point count. The resulting score can increase or decrease the Raw Function Point count by up to 35% [10, 23, and 24].

The GSC can be adjusted to size code complexity by varying the degree of influence of the relevant parameters. For example the FP size of a feature's adherence to performance guidelines may be estimated by varying the degree of influence of the 'Performance' GSC. A re-usable application having the requirement to be configurable or having the ability to be installed in multiple sites (or tested on multiple devices for a mobile application) may be sized by varying the

influence of 'Multiple Sites' and/or the 'Facilitate Change' characteristic.

5.2 Activity Scope as described by Caper Jones

Caper Jones in a paper [11, 12, 25] has described that software projects include many more activities than just coding or programming and has published a list of activity patterns for different kinds of projects. This is a list of around 25 typical activities that are undertaken in software projects and the percentage of effort associated with each activity. He recommends that teams understand which of the most likely activities would be performed in a project and use the activity effort percentage as a guide to estimating software projects. (Ref: Table 2)

Table 2: Caper Jones list of the 25 most applied activities in Software Projects with their % contribution to the estimate

Activity	% Weightage
Requirements	3.84
Architecture	2.25
Project Plan	1.33
Project Management	6.75
Initial Design	3.84
Prototype	4.5
Detail design	4.5
Design Reviews	3.02
Coding	13.5
Unit testing	4.5
Configuration management	0.41
Code inspection	4.5
Formal integration	2.71
Functional testing	4.5
Integration testing	3.84
System testing	3.38
QA	4.5
Field testing	3.02
Independent verification & validation	5.42
Independent third party test	3.38
Acceptance testing	1.94
Installation & training	1.94
User documentation	9.67
Reuse acquisition	1.13
Package purchase	1.63
Total	**100%**

5.3 Extension of 5.1 and 5.2 to accommodate Agile Iterative Development

Our proposition is to use the concepts in section 5.1 and 5.2 to express the size of an agile iteration in Function Points.

Release Planning: Compute the size of all the stories in a release using Function Points based on the information available and estimate the effort / schedule using the organizational productivity baseline creating the macro level estimate.

Definition of Done for a release: Discuss and come to an agreement on the definition for done (DoD) for a release and map it to the activities as per Caper Jones activity scope. The weights given by Caper Jones are an indicator and they may be adjusted as each team sees fit. Table 3 shows definition of done for one of our reference projects. The percent of the applicable activities in our sample project was 87.57% and we normalized the same to100%.

Table 3 : Activities as applicable to a reference project for a release

Sl No	Activity Group	Activity	% Weightage	% Applicable	Normalized %
1	Requirements	Requirements	3.84	3.84	4.4
2	Architecture	Architecture	2.25	2.25	2.6
3	Planning	Project Plan	1.33	1.33	1.5
4	Planning	Project Management	6.75	6.75	7.7
5	Design	Initial Design	3.84	3.84	4.4
6	Design	Prototype	4.5	4.5	5.1
7	Design	Detail design	4.5	4.5	5.1
8	Design	Design Reviews	3.02	3.02	3.4
9	Coding	Coding	13.5	13.5	15.4
10	Coding	Unit testing	4.5	4.5	5.1
11	Coding	Configuration management	0.41	0.41	0.5
12	Code Review	Code inspection	4.5	4.5	5.1
13	Integration	Formal integration	2.71	2.71	3.1
14	Testing and QA	Functional testing	4.5	4.5	5.1
15	Testing and QA	Integration testing	3.84	3.84	4.4
16	Testing and QA	System testing	3.38	3.38	3.9
17	Testing and QA	QA	4.5	4.5	5.1
18	Testing and QA	Field testing	3.02	3.02	3.4
19	Independent Testing	Independent verification & validation	5.42	5.42	6.2
20	Independent Testing	Independent third party test	3.38	3.38	3.9
21	User Acceptance	Acceptance testing	1.94	1.94	2.2
22	User Acceptance	Installation & training	1.94	1.94	2.2
23	Documentation	User documentation	9.67	0	0.0
24		Reuse acquisition	1.13		
25		Package purchase	1.63		
		Total	**100**	**87.57**	**100.0**

Sizing an Iteration: In every iteration, an agile team works on new stories and existing stories. The size of an iteration is the size of the quantum of work done in the iteration. We propose to size an iteration as follows:

- Compute the total size of the stories in an iteration in Function Points
- Identify the percentage of activities to be undertaken towards new and existing stories, i.e. the Definition of Done (DoD) for the New stories and DoD for existing stories in an iteration as mapped to Caper Jones applicable activity scope (DoD for the release)
- Apply this percentage to the total size of the stories to arrive at the iteration size.

DoD for New Stories:. As explained in section 3, the team may not necessarily undertake all the activities related to a

story in the same iteration. New story refers to the first time a story is worked upon in an iteration.

Table 4: Activities as applicable to new stories in a iteration

Activity Group	% Weight	As Applicable for New Stories (%)
Requirements	4.4	4.4
Base	2.6	
Planning	9.2	9.2
Design	18.1	12.7
Coding	21.0	14.7
Code Review	5.1	
Integration	3.1	1.5
Testing and QA	22.0	11.0
Independent	10.0	
User Acceptance	4.4	
Total		**53.5**

For example Table 4 depicts the activities that were undertaken towards new stories in an iteration for a reference project. Since the applicable activities are 53.5% of the total activities to be undertaken for the project, the size of the new stories has been measured as 53.5% of the final size of the same stories that were to be delivered as part of the release. The rationale for only undertaking 53.5% of the total work in the specific iterations is as follows.

With reference to Table 4: Only a percentage of the design as delivered at the end of the project was undertaken in the iteration and this was refined in subsequent iterations. This also meant that code towards realizing the design was also spread across multiple iterations with bulk of the initial coding being done in the current iteration. Similarly code review for new stories was formally done in subsequent iterations hence this activity was not sized in the current iteration. Testing for the stories was carried out across iterations with about 50% of the testing activity being undertaken in the current iteration and Independent testing being taken up in the subsequent iteration. The remaining 46.5% of function points remaining towards realizing the same set of stories for the release were spread across the remaining iterations. This number is not a fixed percent but an example to depict the iterative development cycle.

DoD for existing stories:E very iteration team would also be working on stories delivered in earlier iterations, either for refactoring code on account of design evolutions or an account of testing and bug fixes. Again using the Caper Jones activity scope identify the relevant activities applicable for the iteration towards existing stories (DoD of existing stories) and use the percentage to revise the size estimate of the stories.

Table 5: Activity break up for a set of stories across multiple iterations

		Iteration N	Iteration N+1	Iteraion N+2	Iteration N+3	UAT
Size using FPA		**118.81**	**118.81**	**118.81**	**118.81**	**118.81**
Applicable Size in FP		**115.60**	**115.60**	**115.60**	**115.60**	**115.60**
Requirements	4.40%	4.40%				
Design	18.10%	12.67%	2.72%	2.72%		
Coding	21.00%	14.70%	3.15%	3.15%		
Code Review	5.10%		2.55%	2.55%		
Integration	3.10%	1.55%	0.78%	0.78%		
Testing and QA	22.00%	11.00%	3.67%	3.67%	3.67%	
Independent Testing	10.00%		5.00%	1.67%	1.67%	1.67%
Planning	9.20%	9.20%				
UAT	4.40%				2.20%	2.20%
Total Applicable activity	**97.30%**	**53.52%**	**17.86%**	**14.52%**	**7.53%**	**3.87%**
Applicable Size in FP		**63.59**	**21.22**	**17.26**	**8.95**	**4.59**

Table 5 shows as an example the activity break up for a set of new stories of total size 118.81 FP spread across 4 iterations and user acceptance testing (UAT). Since only 97.3% of the activities as per Caper Jones scope was applicable, the total applicable size is 115.6 FP. The new stories were taken up in 'Iteration N' and the code was reworked / re-factored across the next 3 iterations before it was released for user acceptance testing. Code Review for the stories were taken up in Iteration N+1 and Iteration N+2 (to accommodate for the review process and rework on account of review comments), while the design evolution took place across 3 iterations with about 70% of design being undertaken in Iteration N. Similarly testing was spread across 4 iterations, with about 50% of the testing happening in Iteration N and the remaining % spread across the other 3 iterations. Table 5 thus gives the size of a set of stories as spread across the multiple iterations in the project. This table is an example and in this manner development teams could calculate the size of new and existing stories in each iteration.

Sizing Code Complexity: As described in Section 5.2: GSC can be adjusted to size the impact of varying code complexity. We propose to size the impact of increasing code complexity as follows:

- In the initial iterations size stories with minimal complexity based on the system general characteristics as applicable for the iteration.
- In later iterations when the same stories have to be enhanced for complexity like for example tuning an application to meet performance criteria, the same story may be sized by varying the appropriate GSC.
- The size impact on account of the enhanced complexity would be a difference between the two sizes.

Base Architecture: Activity towards creating a base architecture for the application will be spread across initial

iterations and the activity percentage associated needs to be accommodated in the sizing for an iteration. Since this affects the whole release and is not dependant on a specific set of stories, its size would be a percentage of the total FP size of the release.

Change Management: Changes can be accommodated again using the same techniques as mentioned above i.e.

- Estimate the total size of the change in FP
- Calculate the spread of the change in FP across iterations based on the definition of done
- Estimate the impact of the change on other user stories in FP.

We have applied these techniques on 3 reference projects that we implemented for a client, who is one of the world's leading provider of technology and services to hotels and hotel chains.

6 Reference Projects

Our reference project's objective was to enable our client to provide their customers with hotel booking capabilities on mobile devices. This product was to be white labeled and used by their customers namely various hotel chains and properties across the world. This application was developed for deployment on Android phones, iPhone and iPad and on mobile browsers.

The development was undertaken as three separate projects on account of the varying technologies and development skills required. The activity scope of these projects included Requirement Analysis, Architecture, Design, Development and Testing including Performance Testing and Usability Testing. The client wanted the application to be developed using agile-SCRUM practices. Each project had its own team for the complete project engagement and undertook all the required software development life cycle activities. The application was to be deployed and tested on multiple devices for each project and it also had to support 3 languages (English / Spanish and Japanese). The applications had to meet stringent performance requirements. The code developed was formally reviewed by the client's technical team. Each project had 6 iterations of 3 weeks each and a user acceptance testing phase for 6 weeks. The teams were a mix of experienced and junior developers and they were working together for the first time.

We first estimated the size of the projects as delivered to the customer in Function Points (Table 6).

Table 6: Size of each reference project with total effort in FP

Project	Size in FP	Effort (Person Months)
Project 1 Android	468.2	31.8
Project 2 iOS for iPhone	608.69	47.7
Project 3 Mobile Web	457.2	37.5

We applied the techniques outlined in Section 5 to size the iterations of each project. The initial iteration did not have any existing stories to be worked upon. For all the three projects, in the initial two iterations, the architecture and the reference framework for the entire application were put in place and the size of the iterations were adjusted accordingly. The size of the initial iteration was a percentage of the total size of the new stories along with the size of the percent of work undertaken towards creating the base framework. Subsequent iterations had a combination of new and existing stories. All the stories were worked upon in the first 5 iterations and the last iteration i.e. iteration 6 was dedicated towards testing and bug fixing , refining and polishing the code to make it release worthy.

Table 7: Productivity calculation for Project 1 and its variation

Project 1 Android				
	Iteration Size (FP)	**Effort**	**Productivity**	**Variation**
Iteration1	55.94	4.20	12.0	11%
Iteration 2	53.98	4.13	12.2	13%
iteration 3	80.86	5.7	11.2	4%
Iteration4	87.00	5.6	10.2	-5%
Iteration5	91.44	5.5	9.6	-11%
Iteration6	75.07	4.4	9.4	-13%
UAT	23.88	2.3	15.3	
	468.2	**31.8**	10.8	

We checked to see if the effort expended was comparable to the size computed using our methods. We calculated the productivity of each iteration and checked for the variation. Table 7 shows the size of each iteration (including UAT) and the productivity variation for Project 1 while Table 8 , Table 9 show the same metrics for Project 2 and Project 3.

Table 8: Productivity calculation for Project 2 and its variation

Project 2 iOs				
	Iteration Size (FP)	**Effort**	**Productivity**	**Variation**
Iteration1	55.94	4.6	13.2	8%
Iteration 2	74.4	5.9	12.7	4%
iteration 3	113.50	8.9	12.6	3%
Iteration4	117.30	8.7	11.9	-3%
Iteration5	118.90	8.7	11.7	-4%
Iteration6	97.6	6.9	11.3	-7%
UAT	31	3.9	20.4	
	608.60	**47.7**	**12.2**	

Table 9: Productivity calculation for Project 3 and its variation

Project 3 Mobile Web				
	Iteration Size (FP)	**Effort**	**Productivity**	**Variation**
Iteration1	55.94	5.0	14.3	11%
Iteration 2	54.0	4.9	14.4	12%
iteration 3	80.86	6.6	13.0	1%
Iteration4	87.00	6.5	11.9	-8%
Iteration5	85.44	6.5	12.1	-6%
Iteration6	70.07	5.1	11.6	-10%
UAT	23.88	3.1	20.6	
	457.17	**37.5**	**12.9**	

As we can see the productivity varies between +/- 15% of the mean showing that there is a direct correlation between effort expended in an iteration and size computed using this method. Another trend which can be clearly observed is that in the initial iterations the team was new and having lesser experience in the technologies and hence they worked with lower productivity as compared to later iterations. We have discarded the effort spent in user acceptance testing in our computation. The graphs below (Figure 2, Figure 3) also show that there was a direct correlation between the effort expended in iteration and the size of the iteration computed using this method in all the reference projects.

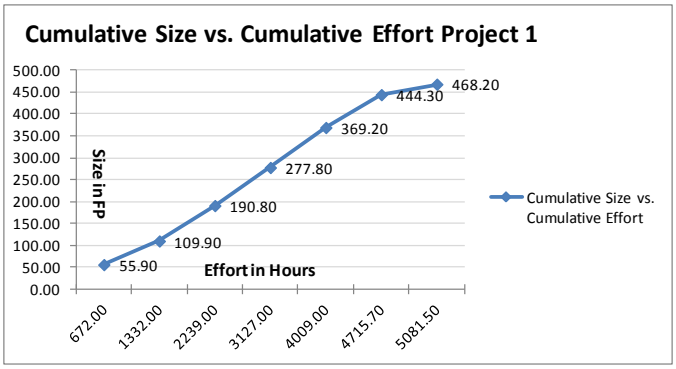

Figure 2: Cumulative Size versus Cumulative Effort for Project 1

There is a slight flattening of the curve at the top which is on account of user acceptance testing being conducted by the client and our role being limited to providing support and undertaking bug fixes.

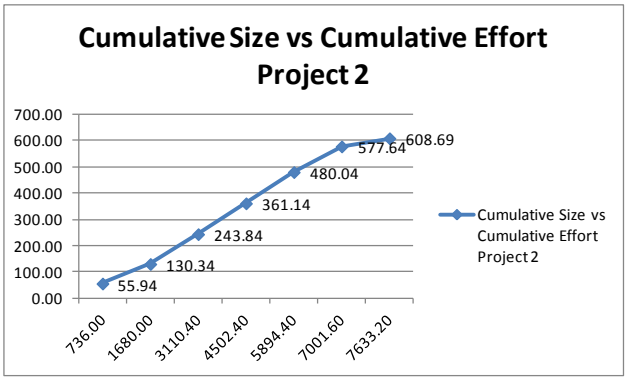

Figure 3: Cumulative Size versus Cumulative Effort for Project 2

7 Conclusion and Future Work

As we can see from the data above this approach to estimating the iteration size in agile projects has promise and needs further exploration. We have applied this method only on projects for mobile devices. We need to extend this idea to larger projects. In order to standardize this technique multiple projects will have to be sized using this method to see if organization productivity benchmarks can be computed reliably and used in new project estimations. The advantage of using this approach is that we can statistically arrive at a method of computing the number of stories which can be realistically taken up in an iteration based on the organization's productivity baseline. However this idea needs to be verified by applying it on a live project from the outset.

In cases where requirements are not reasonably well documented in early stages, it may be useful to initially size iterations using story point methods and subsequently apply the extended function point technique to validate if the effort to stories ratio is consistent and feasible.

We have not done research on how a team's morale may be affected due to an estimator making the estimates as opposed to the team as prescribed by the agile manifesto. We need to examine if such estimates will have the required buy in from execution teams, through independent studies.

Another area that needs exploration is the pre-game phase or iteration zero as called by some, where the initial work on a project happens like putting the team together, doing release planning, capturing basic requirements, setting up the infrastructure etc. We have not identified a method of estimating the size of this phase.

Variations of Function Point counting techniques like MK II Function Points [27] have to be further explored to see if they offer alternative methods to size iterative development.

8 References

[1] K. Beck and C. Andres. "*Extreme Programming Explained: Embrace Change*". Addison-Wesley,2nd edition, 2004

[2] A. Cockburn and J. Highsmith. "*Agile Software Development: The People Factor*". IEEE Computer, 34(11):131–133, 2001

[3] M. Doernhoefer. "*Surfing the Net for Software Engineering Notes*". SIGSOFT Software. Engineering. Notes, 31(1):5–13, 2006

[4] Experience of Executing Fixed Price Off-shored Agile Project , A. Udayan Banerjee*, B. Eswaran Narasimhan *, C. Kanakalata N *

[5] Agile Estimation Using Functional Metrics, Thomas Cagley

[6] Improving Estimations in Agile Projects: Issues and Avenues - Luigi Buglione, Alain Abran

[7] M. Cohn, Agile Estimation and Planning: Addison-Wesley, 2005.
[8] Ceschi, M., Sillitti, A., Succi, G. & De Panfilis, S. (2005) Project Management In Plan- Based And Agile Companies. Ieee Software, 22, 21-25.
[9] Jones, Capers; Applied Software Measurement; McGraw Hill, 2nd edition 1996; ISBN 0-07-032826-9; 618 pages
[10] http://www.ifpug.org/
[11] Software Cost Estimating Methods for Large Projects, Caper Jones
[12] Software Cost Estimation in 2002 by Capers Jones, Crosstalk magazine
[13] Using Function Points in Agile Projects - Célio Santana1,2, Fabiana Leoneo2, Alexandre Vasconcelos2, and Cristine Gusmão3
[14] Using function points in XP – considerations, Andrew M. Fuqua
[15] A User Story Primer – Dean Leffingwell With Pete Behrens
[16] Michael Cohn – Succeeding with Agile
[17] The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework - Jeff Sutherland and Ken Schwaber
[18] http://www.scrumalliance.org/articles/105-what-is-definition-of-done-dod
[19] J Sauer. "*Agile Practices in Offshore Outsourcing – An Analysis of Published Experiences*", IRIS 29, Helsingborg, Denmark, 2006
[20] Fowler, M. & Highsmith, J. (2001) The Agile Manifesto. Software Development, August
[21] Abrahamsson, P., Warsta, J., Siponen, M. T. & Ronkainen, J. (2003) New Directions On Agile Methods: A Comparative Analysis. Ieee, 244-254
[22] Levy, J. V. (2003) If Extreme Programming Is Good Management, What Were We Doing Before?
Edn, 48, 81-82, 84.
[23] http://www.softwaremetrics.com/fpafund.htm
[24] http://www.qpmg.com/fp-intro.htm
[25] Software Engineering An Introduction – Fakhar Lodhi
[26] Jones, C., Programming Productivity, McGraw-Hill, New York, (1986)
[27] MK II Function Point Analysis Counting Practices Manual - 1998