
Agile Estimation Using Functional Metrics

Agility and Discipline

Thomas Cagley,
Vice President
David Consulting Group
440.933.8768 Office
440.668.5717 Cell
t.cagley@davidconsultinggroup.com

Contents

Introduction 3

Budgeting, Estimation and Planning 3

Estimation 4

 Uncertainty: 5

 Self knowledge: 5

 Consistency of Method 5

 Estimation Types 5

Case One: 8

 Case Two: 8

Summary 9

Introduction

The term agile has come to mean many things to many people. The definitions and connotations range from how work is organized within a project to a description of the speed at which work is completed or alternately a radical rethinking of organizational culture. Regardless of how you define agile I would suggest that we all would agree that agile methods are now maturing. Part of the process of maturing is the incorporation of best practices from other methods and frameworks creating a hybrid. The fringe is influencing the center and the center is influencing the fringe. The hybrid is at once better than any of the absolutes and threatening to those who believe in absolutes.

Estimation has been a lightning rod for the discussion all methods (agile, waterfall, iterative or water fountain) with the issues of predictability and standardization radiating outward. Because of the controversy this is an area where a wide range of hybridization has always occurred. Organizations adjust techniques to fit governance structures, culture and risk profiles. There is no one size fits all solution. This paper provides a path for incorporating the use of function points into agile estimation techniques. The process will yield an estimation process that combines one part functional metrics, one part parametric estimation techniques with two parts agile estimation (heavily influenced by Mike Cohn). I would suggest that functional metrics provide a path for incorporating the best practices of robust software sizing with the collaborative techniques championed by the agile community in a manner that increase standardization without ignoring the principals of the Agile Manifesto.

Budgeting, Estimation and Planning

I'd like to begin this discussion by challenging your preconceived notion of estimation as compared to the activities of budgeting and planningⁱ. These three concepts are sometimes thought of as being synonymous however I believe it is important to understand just how different these concepts are. Each has different inputs and outputs, uses different tools and techniques and is generally used by different groups within the organization. A quick overview of the macro differences are:

- Budgeting
 - Defines how much we have to spend based on the influence of scope
 - Tends to ignore the cone of uncertainty
- Estimation
 - Presents an approximation of effort and duration based on size and project nature
 - Focused by the cone of uncertainty (a range based on knowledge)
- Planning
 - Defines tasks and allocates resources
 - Focused on the narrow part of the cone of uncertainty (a much smaller range)

Estimation, planning and budgeting might be related but they are certainly not the same. The use of functional metrics in agile estimation is targeted at the estimation layer of this three layer cake but provides support for planning. Developing a basic understanding of the components of estimation (we are going to ignore budgeting as bastion of guesses) and its relationship to sizing is critical to using these techniques.

Estimation

Estimation is several parts science and a least one part magic. This strange confluence of science and magic defines the transformation of requirements size, skills, people and equipment into how much the project will cost and how much effort it will takeⁱⁱ. The whole process of transformation is bound by a cone of uncertainty. Uncertainty builds boundaries around the false precision of the estimate providing a range around the estimate based on what is known and unknown. Collaborative estimation techniques are good at increasing team knowledge while reducing the amount of self-deceit that can occur when knowledge is discussed.

The amount of art increases as the estimation discipline is replaced by the planning discipline. The art of planning matches specific tasks with people thru a process of assignment. In a perfect world estimates and planning would be able to be done together in seamless workflow but estimates happen generally earlier in the project lifecycle before you can decompose work into tasks which is required for planning.

The simplest form of any estimation model, human or tool based is a mathematical mash-up of size (implied or counted), team and organizational behavioral attributes and degree of difficulty (technical complexity) applied to a productivity signature. As the level of sophistication in the mathematics increases tools SEER, SLIM or KnowledgePlan make sense. Other methods raise level of collaboration and do any of the required math in the heads of the participants. These techniques include Delphi, analogy or planning poker. The process in this paper splits the difference leveraging collaboration to increase participation and self knowledge while suggesting the use of a simple spreadsheet based parametric models to increase consistency and standardization.

Sounds simple, right? Estimation has been a nagging pain in every IT manager's backside since a user asked how much a project would cost and when it would be done. We have gotten pretty good at budgeting using techniques like "x number of people times 20 hours in a day and you'll get something next year" methods. It's when we try to figure out how much functionality will be delivered in real life that things start to break down or least get very, very complicated.

There are three main categories of problems that cause estimation to be problematic in the real world.

1. Uncertainty: how much do you know about what you're building?
2. Self knowledge: what you do really know about yourself and your team?
3. Consistency of method: do you have a process for estimating?

Uncertainty:

A lot has been written about uncertainty, mostly from the point of view of requirements, however the impact of uncertainty extends further than requirements into factors that can be purely technical (whether specific coding languages can do the job) to the complexities of the real world (cue the changing economy as an example). If we change our perspective to completion of the project, I propose that we will all admit that level of project uncertainty is substantially reduced the closer you are to completion of the project. Moving back toward the beginning of the project where most estimation exercises occur one simple truth becomes apparent, knowledge dispels uncertainty. We need to gather better knowledge whether by leveraging history, mathematical algorithms and/or project specific information to make better estimates. Integrating agile techniques for knowledge capture in projects are tools for reducing uncertainty. Techniques to use include incorporating a user or user proxy on the team, focusing on short pre-defined time horizons, implementing processes that foster communication and periodic re-planning.

Self knowledge:

Two psychologists, Joseph Luft and Harry Ingham developed a construct to understand personal awareness. The tool named Johari's Windowⁱⁱⁱ divides personal awareness into four different categories, as represented by its four quadrants: open, hidden, blind and unknown. The lines dividing the four panes are like window shades, which can move as an interaction progresses. The concept is adaptable to teams. Team level blind spots complicate estimation, planning and ultimately performance. Techniques to improve a team's self knowledge include forming stable teams, fostering intimate communications and ensuring retrospectives actually happen often. These tools minimize what isn't known by the team and surface misunderstandings quickly.

Consistency of Method

There are several types of estimation that can be leveraged during any project.

Estimation Types

Let's quickly review the four most popular estimation techniques in very broad terms. The four are:

- Analogy
- Bottom-up
- Parametric
- Delphi

Estimation by analogy starts with the selection of a similar project (building a new bathroom based on the results of building a bathroom last week) which acts as a central metaphor for the new project. The estimator will then decide how closely the two projects resemble each other and whether there are any mitigating circumstances that will affect the effort required to finish the project, how much the project might cost and how long it will take. Based on these differences the estimator will apply a correct factor and voila, an estimate is created. The second general category of estimation techniques is the bottom-up estimate. An estimate of this type typically starts by identifying a set of technical deliverables (a shower stall, sink and pipes for bathroom with a shower if building a house) or work

breakdown structure (the tasks needed to build a bathroom for our house). The identified low level deliverables are estimated based on some form of history and then rolled up to higher and higher levels until the cost or effort for the entire project is known. The third category estimation is called parametric estimation. This form of estimation builds and leverages statistical relationships between historical data and one or more variables that define scope such as functional size (the number of square feet in the bathroom times the productivity of the builders). The fourth type of estimation techniques can be grouped broadly in to the category of Delphi. The central theme of Delphi techniques is the use of collaborative techniques to leverage group think to decide upon an estimate (the plumbers, electricians and carpenters get together and use a process to come to consensus on how much time is required to build the bathroom). These techniques work best when the requirements being estimated can be stated a level of granularity that can be understood by those participating in the estimation session.

Mike Cohn has described the planning continuum using an onion analogy^{iv}. Where strategy is the outer layer followed by layers for portfolio, product, release and iteration segments as you approach the onion core. I suggest that there is no one tool or technique perfectly suited for each level in the onion.

Integrating Cohn's planning onion into our earlier conversation of budgeting, estimation and planning, I would suggest that strategy and portfolio levels are budgeting tasks. The product and release layers are estimation tasks where as tools like white parametric estimation make sense. Iteration and day-to-day organization are planning tasks where planning tools like schedules, kanban boards and standup meeting make sense to direct activity. The method we are introducing in this paper combines the use of functional metrics and tools in the estimation step in a manner that fosters usage in the planning layers of the onion. This set of techniques provides a consistent strategy and answers the changing information needs as the project evolves. All this, while providing a collaborative environment for both the team and the client.

Agile estimation using functional metrics is designed to cover the product and release rings of Cohn's planning onion using a synthesis of parametric and Delphi estimation techniques with the emphasis shifting from parametric to Delphi as events dictate. The technique leverages the ability to size requirements to develop parametric estimates and then dovetails collaborative techniques to refine those estimates based on memories and self-knowledge.

The process flow is as follows

Stage One

1. Identification of functional requirements (or stories)
2. Sizing using Quick and Early Function Points
3. Simple Parametric Estimation

Stage Two – Sprint or Iteration Planning

1. Break Requirements into More Granular Pieces (if needed) and Refine Size
2. Team Level Re-estimation of Requirements Using Delphi Techniques

3. Team Level Commitment

Size is a critical component for developing an estimate and for planning however size and estimates are not synonymous. Size is merely a step along the path from point A to point B. As we move along the path size will be revisited twice.

The sizing process begins by segregating the functional requirements from the non-functional requirements. The functional requirements are then sized using Quick and Early Function Points (QEFPP). Function points for all their warts are the easiest way to consistently size software requirements. This is accomplished by focusing on the basic building blocks of functionality found in all software projects. The QEFPP method leverages the relationship between action verbs and transactions to identify the transaction functions found in function points and the subject of the requirement to identify the data functions found in function points. The application of this technique is similar to sentence diagramming that you learned during grade school or high school. This relationship between words and size has been observed and investigated over the past few years by a number of different people within the functional sizing community including myself (see "Turning Perfect Good Words Into Numbers" originally presented at the IFPUG Functional Sizing Summit at www.davidconsultinggroup.com). Function points or any functional metric has at its heart the goal of converting requirements or stories into a number in a consistent manner. The number then must be interpreted based on the abilities of the team or organization. At the level of the overall project, the technique described in this paper leverages parametric estimation. A simple parametric estimation equation could be:

$$Y = -(7^X - 6 * (X^2)) - Z * X + 26.587$$

Where:

X = Size in function points

Y = Productivity rate for the type of project

Z = Behavior or Process Index

The result is a productivity rate for the project. Collection of historical data on a selection of projects will be required to build an equation. I would further suggest that you will need to augment internal data with external data to increase the validity of the estimation equation. Note the factor can be applied to a disaggregated requirement, epic or story. This estimate is created for organizational planning purposes.

Stage two begins as sprints or iterations are kicked off. The sprint teams (we will use SCRUM terminology) breakdown the stories into pieces that can be accomplished during the sprint, then re-size the pieces using the QEFPP method. The goal of using QEFPP at this point is to take one source of variance out of the estimation discussion that will be had when the Delphi method is applied. This focuses the group on expanding the team level self-knowledge needed to coalesce on an appropriate level of effort needed to complete the story. For example the QEFPP technique has been combined with planning poker into a process that was quickly learned by the sprint teams. The results were a marked reduction

in stories that were not completed during the sprint they were committed to in. By removing size as a variable the team that initially piloted this method indicated they were better able to focus on discussing team capabilities and technical considerations when doing their initial sprint planning.

Real life examples will help drive how organizations synthesized what could be considered competing methodologies into something greater than the sum of the two parts.

Case One:

- Firm: Small custom technology organization
- Project Types: Internal and external projects
- Culture: Highly collaborative
- Current Methodology: Mixed waterfall and SCRUM

Other notes: All external projects are bid with many using a fixed fee structure. Internal projects were continually re-scoped to fit the internal development budget which changes as the economy waxes and wanes. Prior to rewriting the estimation process and leveraging functional metrics approximately 30% of bids were successful and budgets tended to be a suggestion. The lack of estimation success meant that there was a significant risk of losing money if the business was won and if the business was not won going out of business in the long run.

The firm adopted Quick and Early Function Points for sizing the backlogs for all projects, both internal and external. Where backlogs were not being used to manage requirements they were developed. A quick baseline was developed to determine a productivity factor. The productivity factor was then used to translate individual stories into effort. Each team spent a day reviewing how they worked together to generate a baseline of self-knowledge and trust. Collaborative story level estimation was redone using planning poker. It became apparent quickly that a lot of disaggregation was needed to actually estimate the backlogs. After applying QEFPP and the productivity factor to the in-flight projects the firm progressed to applying QEFPP to all bids.

The results were that won bids increased 20% and negative misses were nearly eradicated. A negative miss was defined as underbidding on a fixed bid contract.

Case Two:

- Firm: Large software development firm
- Project Types: Internal projects (software for resale)
- Culture: Hierarchy, classic command and control
- Current Methodology: Mixed waterfall (but SCRUM recently introduced)

Other notes: The methodology in the environment was predominantly classic waterfall with central PMO. Just before we readdressed the estimation process a team had implemented SCRUM and some components of extreme programming (XP) this was done in sort of a guerilla fashion. One very large

project was consuming the majority of the organizations resources. Significant requirements were still being discovered after construction had begun. Estimates had been developed based on a bottom up process very early in the project and they were of questionable validity. The top managers just returned from begging for more money from the board of directors. The project was being capitalized.

The solution in this case was for the company to more firmly embrace the SCRUM framework for project management at the team level. A product backlog was developed and Quick and Early Function Points were adopted to size the backlog. The sizing process exposed a number of functional blind spots (function points can be leveraged as form of analysis). Team members were trained in using QEFP which allowed them to size new stories or re-size changed stories at an individual sprint planning level. The impact of these changes was to allow the PMO to size and preplan the backlog with development leaders and the primary product owner. The full product owner team selected stories to formulate sprints during the sprint planning meetings. Teams re-evaluated (sized and estimated) the selected stories and then committed to the stories they could do.

The initial result was improved product owner satisfaction, involvement allowed them to be part of the solution. After the first few sprint there was an increased perception of estimate consistency both at the product backlog and sprint team level. It was also noted that the teams that had been using the using SCRUM before adopting the new estimation methods had a reduced number of stories that had not completed at the end of the sprint. The teams attributed this improvement during retrospectives to a better capacity to size and commit to work that they're actually able to accomplish during the sprint.

Summary

Agile methods have matured and are now being integrated into many different approaches to the development of software. Estimation has been problematic for all methods; agile to plan based therefore it tends to be a lightning rod for experimentation and synthesis such as is being described in this paper. Agile Estimation Using Functional Metrics has presented a path for integrating the discipline found in functional metrics with the collaborative approaches found in agile estimation.

ⁱ Phil Amour, The Inaccurate Conception, COMMUNICATIONS OF THE ACM March 2008/Vol. 51, No. 3

ⁱⁱ One current thread of thought as voiced by Tim Lister on the Software Process and Measurement Cast 51, www.spamcast.net, is that estimation tends to provide a false sense of accuracy.

ⁱⁱⁱ http://en.wikipedia.org/wiki/Johari_window, June 14, 2009

^{iv} <http://www.mountangoatsoftware.com/presentations/51>, June 14 2009