"How will the #NoEstimates movement impact software measurement?"

December 2016

Scope of this Report

Estimation and software measurements are interrelated concepts but they are not the same. This paper examines the potential impact of adopting a #NoEstimates approach for estimation on software measurement. In April 2015 we published a Trusted Advisor Report titled "So what exactly is this #NoEstimates movement?" The 2015 report discusses the impact of #NoEstimates on software estimation but not on other aspects of software development such as software measurement. Over the past few years, the concept of #NoEstimates has continued to gain steam and has become a movement within the Agile community. If we are trying to judge the impact of the #NoEstimates on software measurement, we need to understand what #NoEstimates actually means. There are several camps revolving around the basic central concept of not generating task level estimates. The central tenant of #NoEstimates has no impact on software measurement, however, some of the most extreme interpretation of the #NoEstimates may change how organizations approach measurement. To have a conversation we need to be begin by establishing a shared context and language across the gamut of estimating ideas whether Agile or not. Without a shared language that includes #NoEstimates we will not be able to understand if there are impacts to measurement concepts.

#NoEstimates Context

There are two macro camps in the #NoEstimate movement (the two camps probably reflect more of a continuum of ideas rather than absolutes). The first camp argues that a team should break work down into small chunks and then immediately begin completing those small chunks (doing the highest value first). The chunks would build up quickly to a minimum viable product (MVP) that can generate feedback, so the team can hone its ability to deliver value. This camp leverages continuous feedback and re-planning to guide work. Agile luminaries like Woody Zuill often champion this camp. A second camp begins in a similar manner – by breaking the work into small pieces, prioritizing on value (and perhaps risk) and delivering against an MVP to generate feedback – but they measure throughput.

Throughput is a measure of how many units of work (e.g. function points, stories or widgets) a team can deliver in a specific period of time. Continuously measuring the throughput of the team provides a tool to understand when work needs to start in order for it to be delivered within a period time. Average throughput is used to provide the team and other stakeholders with a short-term forecast of the future. This is very similar to the concept of throughput used in Kanban. People like Vasco Duarte champion the second camp who practice #NoEstimates from a lean or Kanban perspective. David Anderson, the Kanban visionary, expresses a similar position to #NoEstimates using throughput as a forecasting tool. Both camps in the #NoEstimates movement eschew developing the story- or task-level estimates. The major tactical difference between the two camps is the use of throughput as a forecasting tool. The less minimalist version of #NoEstimates to replace bottom-up estimating and planning found at the lowest level of the classic estimation continuum.

Classic Estimation Context

Estimation as a topic is often a conflation of three related, but different concepts. The three concepts are budgeting, estimation and planning. Because these three concepts are often conflated it is important to understand the differences and relationship between the three in order to understand the impact on measurement if any of the three is not done. It is worth noting that while these labels for the three concepts are typical in a normal commercial organization, they might be called different things depending on your business model. An estimate is a finite approximation of cost, effort and/or duration based on some basis of knowledge (this is known as a basis of estimation). A basis of estimation requires measuring something. Typical measures at this level include cost, effort, duration, and delivered size (e.g. IFPUG Function Points and SNAP). The flow of activity conflated as estimation often runs from budgeting, to project estimation to planning. In most organizations, the act of generating a finite approximation typically begins as a form of portfolio management in order to generate a budget for a department or group. The budgeting process helps make decisions about which pieces of work are to be done. Budgeting and software measurement are often loosely related based on macro-analogy concepts (an approximation technique based on historical performance). Most organizations have a portfolio of work that is larger than they can accomplish, therefore they need a mechanism to prioritize. Most portfolio managers, whether proponents of an Agile or a classic approach, would defend using business or organizational value as a key determinant of prioritization. Value requires having some type of forecast of cost and benefit of the project over some timeframe. Once a project enters a pipeline in a classic organization, an estimate is typically generated. The estimate is generally believed to be more accurate than the original budget due to the information gathered as the project is groomed to begin. The final step in the pyramid is planning. Plans break down stories into:

- tasks often with personnel assigned,
- an estimate of effort generated at the task level
- the sum of the task estimates for roll-up into higher-level estimates.

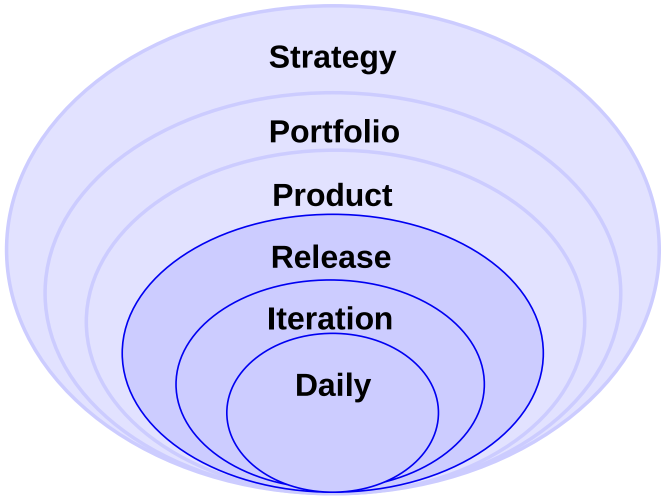
Any of these steps can (but should not) be called estimation. The three -level process described above, if misused, can cause several team and organizational issues. Proponents of the #NoEstimates movement often classify these issues as estimation pathologies. Jim Benson, the author of Personal Kanban, established a taxonomy of estimation pathologies that includes:

1. Guarantism – a belief that an estimate is actually correct.
2. Swami-itis – a belied that an estimate is a basis for sound decision making.
3. Craftosis – an assumption that estimates can be done better.
4. Reality Blindness – an insistence that estimates are prima facie implementable.
5. Promosoriality – a belief that estimates possible.

Estimates by definition are imprecise and can only be accurate within a range of confidence. However, these facts are often “forgotten” in lieu of the single number contract e.g. “This project will take 80-120 days,” becomes “This project will take 80 days” . Acting as if any of these pathologies are true has generated the anger and frustration needed to fuel the #NoEstimates movement which when taken to extreme can have a knock-on effect on software measurement.

Both #NoEstimates and classic estimation are tools to generate feedback (measurement) and create guidance for the organization. In its purest form #NoEstimates uses functionality to generate feedback and to provide guidance about what is possible. The less absolutist “Kanban’er” form of #NoEstimates uses both functional software and throughput measures as feedback and guidance tools. Classic estimation tools use plans and performance-to-the-plan to generate feedback and guidance. The goal is usually the same, it is just that the mechanisms are very different.

Budgeting, Estimation, Planning, #NoEstimates and the Agile Planning Onion



Source: Mike Cohn, www.mountaingoatsoftware.com

There are many levels of estimation budgeting, estimation and task planning. These three types of “estimates” can be mapped to the agile planning onion popularized by Mike Cohn. In the agile planning onion, strategic planning is on the outside of the onion and the planning that occurs in the daily sprint meetings at the core of the onion. Each onion layer closer to the core becomes more tactical. Software measurement plays a different role in each layer and as you get closer to the core of the planning onion the case for #NoEstimates becomes more compelling and the case for software measurement less compelling.

Budgeting is a strategic form of estimation that most corporate and governmental entities perform. In organizational budgeting, strategy and portfolio layers of the planning onion neither the continuous flow technique nor any other #NoEstimates technique answers the central questions most organizations need to answer which include:

1. How much money should I allocate for software development, enhancements, and maintenance?
2. Which projects or products should we fund?
3. Which projects will return the greatest amount of value?

These questions are not software measurement questions but rather macro-level resource allocation questions. Budgets are often scientific guesses that provide some approximation of the size and cost of the work on the overall backlog in the face of imperfect information. Budgeting is needed to secure resources in environments where demand outstrips capacity.

Consequently, budgeting reflects as a prioritization process to allocate scarce resources. Software measurement can and should play a role informing this prioritization and, in all but the most extreme #NoEstimates positions, the need for this level of software measurement isn't even controversial. Budgeting is almost always performed in corporate and governmental settings.

High-level estimation, performed in the product and release layers of the planning onion, is generally used to forecast when functionality will be available given certain assumptions about resource allocation and the availability of information. Release Plans and product road maps are types of forecasts that are used to convey when products and functions will be available. These types of estimates can easily be built if teams that have a track record of delivering similar types of value on a regular basis. The idea of #NoEstimates with throughput measures can be applied at this level of planning and estimation if the right conditions are met. Conditions include:

1. Stable teams
2. Adoption of an Agile mindset (both team and organizational levels)
3. Disciplined execution against the Agile mindset
4. A backlog of well-groomed stories

Organizations that meet these criteria can answer the classic questions of when What and how much based on the predictable delivery rates of #NoEstimate teams (assuming some level of maturity – newly formed teams are never predictable) in this type of environment.

In the standard corporate environment, task level estimation (typically performed at the iteration and daily planning layers of the onion) are an artifact of project management controls or partial adoptions of agile concepts. Estimating tasks is often mandated in organizations that allocate individual teams to multiple projects at the same time. The effort estimates are used to enable the organization to allocate slices of people's time to projects. Planning is the level of the estimation hierarchy where #NoEstimates potentially can have the largest impact on what is perceived as software measurement. Stable Agile teams, that are allowed to focus one project (or very controlled multi-projecting) and use #NoEstimate techniques, have no reason to estimate effort at a task level due to their ability to consistently say what they will do and then deliver on their commitments. Ceasing task level estimation and planning are the core change all proponents of #NoEstimates are suggesting – few of would argue with this approach under these conditions. Recognize that the benefit here comes not from avoiding measuring what is delivered, whether using functional measures or other value based measures, but rather from the avoidance of superfluous time accounting.

A special case that needs to be considered is that of commercial or contractual work. These arrangements often represent lower trust relationships or projects that are perceived to be high risk or both. The legal contracts agreed upon by both parties often stipulate the answers to the what, when and how much question before the project starts which then defines the level of software measurement required to satisfy the contract. Due to the risk the contract creates, both parties must do their best to predict/estimate the future before signing the agreement. Raja Bavani, Senior Director at Cognizant Technology Solutions stated that he thought that #NoEstimates was a non-starter in a contractual environment due the financial risk both parties accept when signing a contract.

It should be noted that while effort estimates in #NoEstimates environments are not done at the planning layers or generally at the estimation layer, most teams adopt rules to break work down into predictable units this is based on a rudimentary form of measurement. Rules or guidelines are often established that affect story and task size e.g. all tasks must be 8-16 hours in duration. However, no effort is made to develop work breakdown structures with effort and names attached. The use of rules to govern granularity is one of the reason flow measures can be used to forecast when work needs to begin in order to meet a date or dependency requirements. Johanna Rothman stated in her article “The Case for #NoEstimates,” that “when you deliver small, valuable chunks of work every day or more often” that you can avoid estimation as a form of planning. The critical words being small and every day which requires the team to understand how to groom stories to the desired granularity. Whether through the use of rules or feedback using these techniques to groom stories could easily be construed as a crude form of measurement.

Summary

#NoEstimates represents a wide range of belief ranging from an extreme of a bit of budgeting but nothing else to budget, high-level estimates but not task estimation. Which end of the spectrum your perception of #NoEstimates falls on will dictate the impact on the software measurement. An estimate is a form of planning that requires a knowledge of the past and therefore data. Planning is a considered an important competency in most business environments. Planning activities abound whether planning the corporate picnic to planning the acquisition and implementation of a new customer relationship management system. Most planning activities center on answering a few very basic questions. When will “it” be done? How much will “it” cost? What is “it” that I will actually get? Rarely does the question of how much effort “it” will take get asked unless as a proxy for how much will it cost. Answering those questions requires data derived from measuring what gets delivered. #NoEstimate changes very little in terms of software measurement (except in very extreme scenarios). Using #NoEstimates techniques still requires most organizations to budget; forecasting requires measurement data. Using #NoEstimates techniques requires breaking down stories into manageable, predictable chunks so that teams can predictably deliver value; breaking down stories requires measurement data. The ability to predictably deliver value provides organizations with the tool to forecast the delivery. #NoEstimates really isn’t not estimating and measurement . . . it is just estimating and measuring differently.

Sources:

http://herdingcats.typepad.com/my_weblog/2015/03/five-estimating-pathologies-and-their-corrective-actions.html 4/27/15

or http://moduscooperandi.com/blog/modus-list-3-our-five-estimate-pathologies/ 4/27/15