GlobalLogic®

Using Story Points to Estimate Software Development Projects in the Commercial Phase

Marcelo Schenone
Delivery Director, GlobalLogic Latin America

Accurately estimating a software development project's total effort is an essential step to providing your customer with a competitive proposal in the commercial phase of a new opportunity. However, both traditional and Agile-based techniques have drawbacks that can negatively impact the process. In this white paper, we will demonstrate a new approach to project estimation that integrates the best features of both traditional and Agile-based techniques.

www.globallogic.com

Marcelo Schenone
Delivery Director, GlobalLogic Latin America

Table of Contents

Marcelo Schenone
Delivery Director, GlobalLogic Latin America

Introduction

As a software development services provider, GlobalLogic is familiar with both the commercial and delivery aspects of starting a new project. Being able to accurately estimate how much work a project will require during the commercial stage is crucial to both correctly pricing out the project and ensuring that you meet the customer's requirements (and our own guarantees) during the delivery phase.

There are currently several techniques for estimating software development projects, both traditional and Agile-focused. Traditional techniques typically estimate all the requirements up-front using various algorithms. They identify functional requirements, measure technical complexity, and adjust technical factors in order to estimate a project's total development effort.

On the other end of the spectrum, Agile methodologies estimate projects on an incremental basis using Planning Poker and Story Points [1]. The development team analyzes the tasks that are required to implement the User Stories and then leverages its team velocity to translate the estimation into hours required to build a release plan. Once the project begins, the team uses the same mechanism to estimate how much work must be done in each Sprint. However, Story Points are team-specific; another team may estimate different Story Points for the same project. Therefore this approach is collaborative and relies on a team's experience.

While both traditional and Agile estimation techniques ultimately provide results regarding the total effort required to execute a software project, they each have their drawbacks. Traditional techniques were not designed with a collaborative, team-centric approach in mind, which is one of the key values of Agile methodologies. However, the estimation techniques associated with Agile methodologies were not designed to standardize effort estimation (i.e., being able to duplicate results independent of the team doing the estimation) or set a baseline of metrics to improve the estimations.

In this white paper, we will examine how GlobalLogic took the best features of both techniques to create a better approach to estimating software projects. Starting with the estimation technique used in Agile methodologies (i.e., based on Planning Poker and Story Points), we integrated features of the Use Case Points technique [2] and then adapted this new technique to estimate full projects in their early phases. We have defined and tested this estimation technique for mobile technology software development projects leveraging Android, BlackBerry, iOS, and HTML5 technologies.

Definition of Shared Pivots

GlobalLogic's first goal in creating a new estimation technique was to standardize the measurement of complexity in Story Points. We therefore developed the concept of a "Shared Pivot," which is a unit of estimation composed by a name, a description, and a weight in Story Points. This will be the basis for technical estimation.

Our next step was to build an inventory of Shared Pivots to standardize the estimation of component complexity in Story Points. We produced a number of Shared Pivots that can be used to estimate any User Story on a project, and we identified four Pivots as the basis for the estimation technique.

Standard Text: a screen or display component that contains standard textual information without business logic behind it

Remote Image View: a screen or component for displaying information extracted via an API

List: a screen or component that displays a list of structured information obtained through an API

Form: a screen or component that displays a form that gets information by calling an API with client-side validations.

Name	Weight in Story Points
Standard Text	1
Remote Imagine View	3
List	5
Form	8

Estimating the Product Backlog

As part of the estimation technique, one of the first tasks is to define the project scope. In Agile terms, this means generating the Product Backlog and estimating it. The Product Backlog is composed of a list of User Stories defined in the standard way [3]: *As a (role), I want (goal/desire) { so that (benefit) }.*

Examples of User Stories include:

- As a user, I want to see the terms of service.
- As a user, I want to login as a member so that I access my favorite features.
- As a member, I want to logout so that I can end my session with the application.
- As a member, I want the system to remember my session information.

Each User Story identified in the Product Backlog will be estimated using the previously described Shared Pivots. The User Story will be estimated by selecting the Pivot that is more similar to the functionality being described. The Pivot has a specific weight in Story Points, which can be adjusted if there are any technical differences that add or reduce complexity. Finally, a table is created with the complete size of the project in Story Points (example below).

This table summarizes the whole functional complexity of the project in Story Points. The first goal of achieving a standardized estimation in Story Points is completed, and the technique continues with adjustment factors.

ID	User Story	Selected Pivot	Pivot Story Points	Adjusted Story Points for Story	Story Points Adjustment Reasons
1	As a user, I want to login as a member.	Form	8	13	Landscape
2	As a user, I want to see the conditions of service.	Standard Text	1	3	Rich Text (HTML)
			Total Adjusted Story Points	16	

Technical Adjustments for Estimation Based on UCP

In order to estimate a development project, we have to consider several concepts related to nonfunctional requirements or team constraints. As such, we borrowed some concepts from the Use Case Points (UCP) approach (e.g., technical and environmental factors) in order to adjust our effort estimation technique.

UCP is based on counting the actors and use cases from a Use Case Model. From this total count, a metric called Unadjusted Use Case Points (UUCP) is obtained. The UUCP is then adjusted by Technical Complexity Factors (TCFs) and Environmental Factors (EFs), which were mostly inherited from the Function Points technique. The adjusted UCP is then used to measure the final functional complexity of the project. Finally, the adjusted UCP is multiplied by a productivity factor that results in project estimation in man hours. The productivity factor is calculated by estimating the average effort it takes for a standard developer to implement one Use Case Point (i.e., similar to the team velocity in the Agile estimation technique).

The TCFs are used to adjust for various non-functional requirements that are not reflected in the functional requirements or Product Backlog of a project. TCFs are related to all non-functional requirements that typically exist in a software development project such as performance, concurrency, usability, maintainability, etc.

The table below lists all TCFs in the UCP technique:

Technical Factors	0 to 5 Scale	Weight
T1 Distributed System	0 = Not Important 5 = Essential	2
T2 Performance	0 = Not Important 5 = Essential	1
T3 End User Efficiency	0 = Not Important 5 = Essential	1
T4 Complex Internal Processing	0 = Not Important 5 = Essential	1
T5 Reusability	0 = Not Important 5 = Essential	1
T6 Easy to Install	0 = Not Important 5 = Essential	0,5
T7 Easy to Use	0 = Not Important 5 = Essential	0,5
T8 Portable	0 = Not Important 5 = Essential	2
T9 Easy to Change	0 = Not Important 5 = Essential	1
T10 Concurrent	0 = Not Important 5 = Essential	1
T11 Special Security Features	0 = Not Important 5 = Essential	1
T12 Provides Direct Access for Third Parties	0 = Not Important 5 = Essential	1
T13 Special User Training Facilities Required	0 = Not Important 5 = Essential	1

The Environmental Factors (EFs) are used to adjust project management complexities, primarily those around team constraints. They are related to the experience of the development team, the degree of stability in the requirements, the difficulty of the programming language, etc. The table below lists all EFs in the UCP technique.

These adjustment factors are applied to the total project estimation and given a final number of adjusted UCP. Since the Agile estimation technique of Story Points does not include these factors for technical and environmental adjustment, they will be added and expanded in the estimation technique proposed.

Environmental and Team Factors	0 to 5 Scale	Weight
F1 Familiar with Unified Process	0 = No Experience 3 = Medium 5 = Expert	1,5
F2 Application Experience	0 = No Experience 3 = Medium 5 = Expert	0,5
F3 Object Oriented Experience	0 = No Experience 3 = Medium 5 = Expert	1
F4 Lead Analyst Capability	0 = No Experience 3 = Medium 5 = Expert	0,5
F5 Motivation	0 = No Experience 3 = Medium 5 = Expert	1
F6 Stable Requirements	0 = No Experience 3 = Medium 5 = Expert	2
F7 Part-Time Workers	0 = No Experience 3 = Medium 5 = Expert	-1
F8 Difficult Programming Language	0 = No Experience 3 = Medium 5 = Expert	-1

Adapting Technical and Environmental Factors

While developing this new estimation technique, we decided to unify the technical and environmental factors into a single category. Our reasoning is that this approach will simplify estimation and reduce the size of the estimation template sheet.

Each of the TCFs and EFs within the UCP technique were analyzed so as to decide if they applied under a context of Agile estimation. Moreover, we leveraged the experiences of our own architects and technical leaders to add other factors that were deemed important to consider in adjusting the total effort. Based on this analysis, we identified 27 technical and environmental factors that will be used to adjust the estimated effort in Story Points. Below is a detailed description of each factor:

Performance: establishing whether the application has performance requirements that must be taken into account when developing it, or directly dependent on the same performance that comes by default on the mobile platform

UI Quality: if the application needs a very advanced user interface, enabling rich usability of the application

Algorithmic Complexity: if the business requires the use of complex algorithms such as mathematical formulas, complex business rules, large calculations, statistics, etc.

Reuse: if the application requires the development of certain components that may be reused for other applications

Deployment: if deployment will occur in a market or app store (i.e., an app store's approval times are a key consideration)

Ease of Use: if the application has specific accessibility requirements or complex levels of user experience for different audiences

Portability: if the application must be ported to other platforms (i.e., using cross-platform frameworks could potentially solve this problem by developing a unique code base that is then automatically ported to other platforms)

Stable Requirements: if the functional requirements are well-defined and specified or are volatile in their definition (i.e., this can impact the development effort due to rework)

Concurrency: if the application has multi-threading requirements or distributed concurrency

Security: if the application requires specific testing and security development, including HTTPS encryption on DB/FS, testing of vulnerabilities, etc.

Integration with Other Systems: used if the application has integration with external applications or APIs (e.g., Facebook, Twitter, Google Services, etc.)

Experience with the Development Methodology (Scrum / Kanban): the experience level of the team who is working with the organization's development methodology

Expertise in the Domain: if the team has experience in the application domain (e.g., banking and finance, consumer, telecommunications, etc.)

Full-Time Allocation: if the team members are fully allocated to the project or simultaneously working part-time on other projects (i.e., multi-tasking impacts productivity of the team)

Expertise in technology: if the team already has experience in the technology stack in which the application is being developed

Portrait and Landscape Support: if the screens need to be oriented in both portrait and landscape mode on the devices (i.e., requires developing different versions of the user experience)
Different Resolutions: if the application requires support for different screen resolutions (i.e., may impact redesign or user experience)

Several Devices: if the application needs to be developed for a finite number of devices or for a large number (i.e., can increase the project's testing effort)

Different Versions of OS: if the application must run on different versions of an OS (i.e., may impact the development efforts due to background compatibility)

Persistence / Caching: if an app must persist data in order to work offline (i.e., impacts development efforts)

Battery Consumption: whether to develop the app so that it minimizes battery consumption

Languages: whether to develop in multiple languages (i.e., can increase the testing effort)

Distributed Team: if the team is distributed geographically (with the challenges of a distributed team) versus a team located in a single place

Engagement Type: if the engagement with the customer is Time and Materials, Outcome-Based, or Fixed Time Fixed Price (i.e., in Fixed Time Fixed Price, you need requirements management overhead in order to monitor scope creep; the other two engagement types are more Agile-friendly and scope is negotiated dynamically)

Customer Language: if the team can speak fluently in the same language as the customer (i.e., language barriers can impact communication and team productivity)

Customer Time Zone: if the customer is in the same time zone and can communicate with the team throughout the day, or if the customer is in another time zone and has less overlap (i.e., this can impact the speed of development when interacting with the product owner or other stakeholders)

Application Exposure: if the application will be published to the world in the app store or is only for internal usage (i.e., this can impact decisions regarding user experience and expected quality levels)

For each of these factors, a weight must be defined. The range will increase or decrease the weight of the project's total number of Story Points. This adjustment can be executed two different ways:

(1) Adjustment in Functional Complexity (ACF): variability of +5 / -5 in Story Points for the whole estimation effort

(2) Adjustment in Team Velocity (ATV): variability of velocity of +5 / -5 Story Points per week for the whole estimation effort (see the next section on Productivity for the usage of this adjustment)

Finally, the Adjusted Story Points are calculated by adding or subtracting the total number of Story Points that result from the sum of the ACF to the Unadjusted Story Points. In this way, the technique introduces technical and environmental complexity adjustments that result from both nonfunctional requirements and project management scenarios. The result is a final number in Adjusted Story Points that is the total weight of the Product Backlog.

Productivity

Our final step in creating a new estimation technique is mapping the total number of Adjusted Story Points to the total estimation effort using the team velocity. This will provide the basis to build a release plan and set up a development team. This step is executed through the Productivity Factor.

The Productivity Factor is used to estimate the number of Story Points that will be implemented in a certain timeframe (i.e., the velocity concept in Agile methodologies). This factor represents how many man-hours are required to build one Adjusted Story Point. The Productivity Factor depends on each organization, development team, and customized methodology. At GlobalLogic, all tasks related to the implementation of a User Story (which need to be taken into account when estimating team velocity) are described below:

Front-End
Mobile front-end
Automated testing
Unit testing
Integration

Back-End
HTTP client to communicate with service
Exception handler
REST resource
Automated testing
Unit testing
Integration

Analysis
Define acceptance criteria for back-end
Define acceptance criteria for front-end
User testing
Manual testing

UX
Wireframe(s)
Mockup(s)

The correct way to estimate the Productivity Factor is through past project experiences. Karner, in his original paper [2], comes to a total of 20 MH x AUCP (Man Hours per Adjusted Use Case Points), which was later taken as one of the most utilized industry standards.

Our own mobile practice leaders analyzed the effort of previous projects and proposed a value of 2 MH x ASP, which is the standardized velocity as measured by the technical experts for GlobalLogic mobile teams. This velocity size will also be adjusted by the Adjustment in Team Velocity (ATV), decreasing or adding to productivity.

Finally, our new estimation technique multiplies the total number of Adjusted Story Points by the Productivity Factor to arrive at a total number of man-hours, which is the input for defining team composition and building the release plan.

Conclusions

GlobalLogic has been using this new estimation technique for over a year with good results, and we use each new project experience to adjust the different factors according to organizational and business realities. This approach has enabled us to provide our customers with more accurate estimates, better project management, and overall increased satisfaction.

We are now refining the technique to provide more accurate productivity estimates, and we are also expanding the initiative to other core technologies such as Java and .NET development.

References

[1] Mike Cohn, Agile Estimating and Planning, Prentice Hall, November 11, 2005

[2] Gustav Karner, Resource Estimation for Objectory Projects, September 17, 1993

[3] Mike Cohn, User Stories Applied, Addison Wesley, 2004

About the Author

Marcelo Schenone is the Delivery Director and Head of PMO at GlobalLogic Latin America, where he has worked for over 13 years. Marcelo also teaches requirements management and process improvement at the University of Buenos Aires (UBA), and he is currently conducting research projects on Agile methodologies (XP, Scrum, Lean), software improvement frameworks (CMMI, ISO 9001, ISO 90003), and project management (PMI, Agile PMI). Marcelo graduated from the Engineering College at UBA, and he earned his MBA in Strategic Management from the University of Barcelona.

About GlobalLogic Inc.
GlobalLogic is a full-lifecycle product development services leader that combines deep domain expertise and cross-industry experience to connect makers with markets worldwide.Using insight gained from working on innovative products and disruptive technologies, we collaborate with customers to show them how strategic research and development can become a tool for managing their future. We build partnerships with market-defining business and technology leaders who want to make amazing products, discover new revenue opportunities, and accelerate time to market.

For more information, visit www.globallogic.com

GlobalLogic®

Contact

Emily Gunn
+1.512.394.7745
emily.gunn@globallogic.com