Function Points, Use Case Points, Story Points:
Observations From a Case Study

Joe Schofield, International Function Point Users Group
Alan W. Armentrout, Sandia National Laboratories
Regina M. Trujillo, Sandia National Laboratories

Abstract. Software development success, estimation, and measured value continue to challenge projects and organizations today. Schedule and budget seem predictable, perhaps unjustifiably in light of the inability to capture product size expectations early and often. Function Points (FP), Use Case Points (UCP), and Story Points (SP) are three approaches for establishing size (or size of effort). Each of these is applied in a case study to identify similarities and differences. The measure of preference is left for the discerning reader. Limited details about the project, its team size, duration, and goals are intentionally excluded from the research since the intended focus is on comparing the project's measurements.

Introduction

A measure of the size of a software application is required for many different kinds of analysis and decision making. This article looks at three popular methods used today by software project teams to estimate the size and effort required to develop software: SPs, UCPs, and FPs. Of the three, FP measurement is the only one based on an international standard: ISO/IEC 20926:2009. Use case points referred to in this article are based on work described by Clem [1], Cohn [2], and Schneider and Winters [3]. Story points emerge from the Scrum methodology and assign a level of complexity for completion of development tasks [4, 5].

Comparisons among these three measures are rather limited in the literature. Reasons abound as to why the measures should not be compared [6], yet it is natural if not necessary to compare and contrast projects, products, and organizations. The cautions are warranted given historic attempts, for instance, to use backfiring to convert lines of code to function points for comparison. However, because measurements are developed independently and for slightly different purposes, it does not imply that they might not have some discernible and useful association. Until the research has been performed, one cannot dismiss the possibility of meaningful relationships among the measures. During the International Software Measurement and Analysis Conference in Richmond, VA, in September 2011, an "agile working group" identified exploring the relationships among some of these measures as a "priority."

Within this article, these three measures were used to see if insightful comparisons could be made among them. It describes how measures for each of the three methods were calculated for an actual application, and compares them to each other with respect to results, effort to compute, and some, albeit, subjective characteristics. Selecting the one best suited for the current need is left to the discerning reader. A single study does not constitute a statistically valid sample but we are hopeful that it stimulates additional discussion and interest.

Each of the three measures was calculated independently of the other two, and each measure was computed by different individuals. The three measures were determined as follows:

- Story points were created and worked throughout the product development cycle by analysts working with the product owner for the application and were later refined by the developers during their sprint planning sessions. Only implemented story points were included in this study.
- Use case points were developed from use cases produced by a use-case modeler after the product was implemented and were based upon a review of the implemented application. The use cases were validated by the software project team's product verification manager to ensure that the scope was consistent with that used for the story points.
- Function points, like the UCPs, were determined based upon the completed product. A certified, trained, and experienced function point counter developed the measures, once again working with the team's product verification manager to maintain scope consistency with the other two measures.

Note: The traditional lines of code (LOC) are not considered in this article. Jones has called the use of LOC as a measure of software size "professional malpractice." Schofield has demonstrated the statistical unreliability of LOCs. In his worst documented case, the same program, written in the same language, developed by similarly educated developers (most with masters degrees and working as developers), accepted by the same instructor, with counted LOC identically, yielded variations as great as 22:1, and median variations around 9:1 [7].

Applied Story Points – a Closer Look

User story elicitation was the genesis of the SP allocations. It began with input from Subject Matter Experts (SMEs), management and end-users (stakeholders), given to the product owner, where high-level epics were transformed into functional user stories written in the language of business. This activity was followed by weekly grooming sessions where contributors prioritized the list of user stories, decided which are chosen for the next sprint, and then transformed to be sprint-ready, i.e., sprint-sizable stories. These user stories, in basic form, illustrated the functionality that was most significant to the stakeholders. When the bi-weekly Review / Retrospective / Planning (RRP) session commenced, the user stories were brought forth and story points allocated. During the session the development team proposed what it could commit to delivering by the end of the two-week sprint.

This case study's agile software development planning practices included the trendy Fibonacci [8] sequence; this metric is used to estimate the effort of each user story without assigning actual hours. Each number in the Fibonacci series (0, 1, (1), 2, 3, 5, 8, 13, 21, 34, 55 ...) measures relative effort. As an example, an 8 is eight times more effort than a 1. During the weekly grooming sessions, epics (groups of related user stories) were assigned a value using the Fibonacci sequence to provide an overall estimate of effort. Prior to the RRP session, the velocity was defined to suggest the total number of story points the development team could deliver in the sprint. The velocity is calculated based on the team's hours of availability using the following equation:

Projected Sprint Velocity = Average Velocity * (Total Hours this Sprint) / (Max Total Hours)

The resulting Projected Sprint Velocity was used during the RRP session as a basis of planning. Once the lead developer knew the level of commitment, "gut checks" were used by the lead developer and the product owner session to establish the expected velocity for the sprint. After each sprint-ready user story was selected from the repository, arbitrary values were assigned to each user story by each developer to deduce the size (story point count). The resulting value roughly equated to the original "gut check" size as confirmed by other team members through process observation. This behavior is apparently familiar in the agile world, given the following insight: "one characteristic of story points is … that they are a relative measure—which means that they compare the size of one story to another and there is no need for a standard meter value for 1 point. This approach provides flexibility and allows for the gut feeling, experience-based judgments to become statistically accurate." [9]

Note that the words "size" and "effort" are used almost interchangeably by practitioners and Scrum promoters and in fact share overlapping lexical semantics.

Table 1 exhibits the story point distribution over a 12-month development effort of 24 sprints. A total of 50 user stories were produced. Although not reflected in the Table, story point counts remained rather consistent for each two-week sprint.

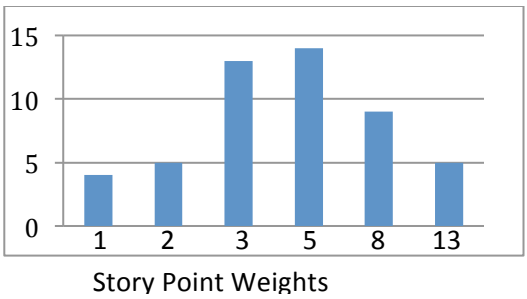

Table 1: Story Point Counts vs. Fibonacci Values

After story points were assigned, each developer decomposed their user story into several tasks, smaller units of measurable work. Following the agile approach [4], each task was assigned a value in terms of hours. These hours reflected the estimated length of time for the developer to complete the task during the two-week sprint. In addition, the features of Business Value and Developer Value were assigned H (high), M (medium), L (low). These two features were only applied during the user story estimation and not used as a means of measure or traceability.

Applied Use Case Points – a Closer Look

The following process was used to develop the use cases and use case points for the case study. The steps used to determine the individual use cases include:

1. Established user goals for the application.
2. Determined individual use cases needed to accomplish the user goals.
3. Outlined the basic flow, subflows, and alternative flows for each use case.

The steps used to calculate the UCPs for the system consisted of summing the unadjusted use case weight (UUCW) and the unadjusted actor weight (UAW) values as detailed by Clem, Cohn, Schneider and Winters, and Zawari [1, 2, 3, 10]. Technical Complexity Factors and Environmental Factors were assumed to evaluate to 1.0, roughly equating to assigning an average impact to each of the factors bearing on the development of the system. This assumption is not consistent with how UCPs are typically calculated. However, it is plausible if UCPs are used to produce a size estimate early in a system's life before the project team is fully assembled and all architectural decisions have been made. This treatment is consistent with excluding the General System Characteristics from the Function Point count below; that is, to assume a mid-value for their sum.

The steps used to determine the UUCW value were as follows:

1. Determined the number of use case transactions for each use case as reflected within its basic flow, subflows, and alternative flows. (See Collaris and Dekker [11] for a discussion on determining use case transactions.)
2. Associated each use case to a use case type using its use case transactions count according to Table 2.
3. Counted the number of use cases by use case type.
4. Multiplied the use case type count by its corresponding weight.
5. Summed the weight-count products across the three use case types to arrive at a UUCW.

Use Case Type	Number of Transactions	Weight
Simple	1 – 3	5
Average	4 – 7	10
Complex	> 7	15

Note: Cohn's, Schneider and Winters' determination of use case types differs from that of Clem's and Zawari's. Cohn's, Schneider and Winters' were used here.

Table 2: Use Case Types

Table 3 shows the UUCW for the case study is 65, as the system has 3 simple use cases, 2 average use cases, and 2 complex use cases.

Use Case Type	Number of Transactions	Weight	Use Case Count	Weight X Count
Simple	1 – 3	5	3	15
Average	4 – 7	10	2	20
Complex	> 7	15	2	30
			UUCW	65

Table 3: Unadjusted Use Case Weight

The steps used to determine the UAW were as follows:

1. Associated each actor to an actor type according to Table 4.
2. Counted the number of actors by actor type.
3. Multiplied the actor type count by its corresponding weight.
4. Summed the weight-count products across the three actor types to arrive at a UAW.

The UAW for the case study is 6, as the system has two actors who interact with the system using a graphical user interface.

The UCP was calculated by adding the UUCW to the UAW. The result of applying the above steps produced an UCP for the case study of 71 (65 + 6).

Applied Function Points – a Closer Look:

The scoping and counting were intended to include all of the functionality delivered to the customer, and only the functionality delivered to the customer. These two considerations are two of Albrecht's three principles of function point counting. The third aspect of Albrecht's [12] seminal characterization was to ensure that function points are developed independent of technology. Table 5 summarizes the data and transactional functions with the subject product.

The values associated with the two data functions and three transactional functions, and their three levels of complexity, as defined in IFPUG's Counting Practices Manual [13] follow.

Multiplying the appropriate values, the case study is measured at 161 unadjusted function points:

(7 * 12) + (5 * 1) + (10 * 3) + (1 * 4) + (2 * 4) + (10 * 3)

The distinction between unadjusted and adjusted is calculated by determining the 14 General System Characteristics (GSCs). Each of these 14 characteristics has a value ascribed to it between 0 and 5 based on a complexity scale that is unique for each characteristic. For the purposes of this example, the GSCs are assumed as the middlemost value, implying no increase or decrease in the UFP count. A similar median value was used for the use case point derivation above. The intention is to keep the counts simple and consistent, and not to introduce the opportunity for variation in assumed characteristics.

"Learning organizations" with historical data have an advantage in being able to review delivery numbers with past performance. Otherwise, organizations undermine their own measurement programs by lessening the value they could be realizing. (As a reminder, the CMMI®-DEV explicitly includes a measurement repository in the Organizational Process Definition process area as specific practice 1.4.) The following organizational values are not intended to represent a particular organization and are used as examples only. Assume for this case study, that the project estimated cost was based on a fixed schedule and whatever work was achievable within its Scrum-based release and sprint planning.

One might ask why size measures are being compared to effort measures. We proffer that because the measures are different it does not follow that useful comparisons cannot be made. Once a size estimate has been calculated one can derive an effort estimate if historical data is available.

Given the existence of related UCP organizational data, similar values could be projected for the cost of a UCP, or expected delivery for 71 UCPs, or defects per UCP, or number of UCPs developed per person month. And of course, similar values could be derived for story points. The presence of that data would likely provide useful performance metrics for monitoring and controlling of the project and for executive decision-making for prioritizing investments in selected development environments and methods.

Actor Type	Description	Weight
Simple	Another system through a defined API.	1
Average	Another system, interacting through a protocol like TCP/IP, or a person interacting through a text-based user interface.	2
Complex	A person interacting through a graphical user interface.	3

Table 4: Actor Types

	Low	Average	High
Internal Logical Files (ILF)	[1]12		
External Interface Files (EIF)	1		
External Inputs (EI)	10	1	
External Outputs (EO)	2		
External Inquiries (EQ)	10		

Table 5: Function Point Data and Functional Transaction Types

	Low	Average	High
ILF	7	10	15
EIF	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

Table 6: Function Point Data and Functional Transaction Values

	Organization's Historical Average	Case Study "Actual"
Function Points (developed) Per Person Month (FPPPM)	25	1.7
Person Months Delivery (for 161 FPs)	13.4	126
Cost Per FP	$425	$14,086
Defect Per FP	0.13	.92

Total Function Points: 161; Total Person Months: 126; Total Cost $2,268,000; Total Defects: 148 (Defects from peer reviews and test—pre-release)

Table 7: (Example) Organizational Historic Values Compared to Project and Actual Project Values

Conclusions

Given the data in Table 7, it is hard to imagine that a project that deviates this much from past performance would be allowed to continue "as is." So while the data in the case study may seem extreme, a better question might be, "at what point in a project's life does one initiate corrective action?" Certainly variation outside a range should trigger corrective action; failure to adhere to that range or percent can be a sign of "inattentiveness." Still other implications related to the team and personnel performance begin to emerge; but, that is not the target of this case study.

We are hopeful that the following characterizations advance the understanding and serious analysis of measurement methods. We have at least documented one such case.

Characteristic	Function Points	Use Case Points	Story Points
Useful at the project level for estimating or planning	With historical FP data	With historical UCP data	With historical SP data
ISO / Standards based	ISO 20926	no	no
Captures customer view	Expected	Expected	Definitely
Useful for benchmarking outside the company	If used as intended	Could be	Not intended to be used across projects
Easy to calculate	Less so	More so	Yes
Easy to validate for repeatability / consistency	More so	More so	Less so
Objectivity	More so	More so	Less so (team / team member variability)
Technologically independent	Yes	Yes	Maybe
Functional measurement to customer	Yes	Yes	Not exclusively (may include refactoring, design, and other work)
Time to derive estimate	Less than one hour	Less than 30 minutes	Indeterminable but significantly larger than other measures

Table 8: Comparing Function Points, Use Case Points, and Story Points

While there are no direct correlations of function points to use case points, or vice versa, to our understanding, effort estimates may reveal useful insights. Therefore, the following effort calculations were derived for the case study:

- 161 function points / 25 function points per month (a historical organizational average) / 12 months per year = .54 year of effort
- 71 use case points * 20 hours per use case point (the low value suggested by Schneider and Winters [3]) / 2080 work hours per year (52 weeks * 40 hours per week) = .68 year of effort
- 71 use case points * 28 hours per use case point (the high value suggested by Schneider and Winters [3]) / 2080 work hours per year = .96 year of effort

Note: The effort value for Story Points is not rendered because initial values during the development of the product backlog, subsequently re-calculated by the development team during sprint planning, are subject to additional variables like refactoring, grooming, and the developer's estimation capability.

Productivity measurement is seldom a topic welcomed with open arms by project teams. Early reticence to measurement for organizations and teams is often characterized by comments like:

- Measurement is hard; (though it is even harder if postponed or ignored).
- We make only limited decisions based on measurement data, thus the need to collect it seems specious.
- How do my numbers compare to others? We may need to adjust them.
- Our numbers could be used against us.
- Our numbers could be better; for now, they are good ballpark estimates.

A Time Magazine article "Good Guess—Why we should not underestimate the alue of estimating" examines the importance of estimating. The article encourages parents to incorporate estimating into their children's thinking at a very early age citing among others, a study from Carnegie Mellon [14]. Contrast this thinking with attempts in the agile world to shift the discussion towards "level of difficulty" versus a delivery time estimate [5].

Two of the purposes of "counting" in the software development world are to provide insight for the awaiting customer and improvement across various development activities. Use Case Points, Story Points, and Function Points are three techniques that can provide measurement insight for software projects. It is less obvious that each of these provide similar value to the customer or organization for scheduling. Perhaps the "maturity" of the organization and the culture defines the "tolerance level" (adoptability?) of organizational measures. An absence of relevant comparisons has been published thus far; rendering the verification of relevant measures difficult. It is too early to suggest that valid comparisons or the emergence of preferences among these measurements are unlikely. Comparisons are more easily facilitated by industry standards as is the case for Function Points.

Acknowledgements/Disclaimers:

The authors wish to acknowledge Don Likfe for his review of the representation of the Fibonacci data in the paper.

The "Applied Story Points – A Closer Look" and "Applied Use Case Points – a Closer Look" sections of the article were funded by Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

ABOUT THE AUTHORS

Joe Schofield is the President of the International Function Point Users Group. He retired from Sandia National Laboratories as a Distinguished Member of the Technical Staff after a 31-year career. During twelve of those years he served as the SEPG Chair for an organization of about 400 personnel which was awarded a SW-CMM Level 3; he continued as the migration lead to CMMI Level 4 until his departure. Joe has taught graduate courses since 1990. He has over four dozen published papers, conference presentations and keynotes—including contributions to four IT measurement books. He is a SEI-authorized Instructor for the Introduction to the CMMI and two other SEI courses, CSQA, CFPS, CSMS, and a LSS BB. Joe is a frequent presenter in the Software Best Practices Webinar Series sponsored by Computer Aid, Inc. Joe completed his Master's degree in MIS at the University of Arizona in 1980.

Alan W. Armentrout, CPA, Inactive, is a Distinguished Member of the Technical Staff at Sandia National Laboratories. He has worked at Sandia for more than 23 years analyzing, designing, and developing information systems. Alan is a Certified Professional for Requirements Engineering, Foundation Level with the International Requirements Engineering Board. He completed his Master's Degree in Management Information Systems at Colorado State University in 1988. Prior to completing that degree, Alan worked six years as an accountant with an international petroleum company.

Regina M. Trujillo is a Member of Technical Staff at Sandia National Laboratories where she specializes in verification and validation of software systems, from complex satellite groundstations to simple web applications. She also has knowledge and experience in running security, load, performance, and transactional analysis tests against these software systems to ensure compliance with corporate standards. Regina has a Bachelor's degree in Computer Information Systems from New Mexico Highlands University.

REFERENCES

1. Roy Clem; "Project Estimation with Use Case Points"; 03/22/2005; retrieved from <http://www.codeproject.com/KB/architecture/usecasep.aspx> 12/2011
2. Mike Cohn; Estimating with Use Case Points; <www.methodsandtools.com>, retrieved 8/29/2011
3. Geri Schneider and Jason P. Winters; Applying Use Cases, Second Edition: A Practical Guide; Addison-Wesley; 2001.
4. What is a Story Point?; November 13, 2007; retrieved from <http://agilefaq.wordpress.com> 12/23/2011
5. Mike Cohn; "It's Effort not Complexity"; 9/19/2011; retrieved from <http://blog.mountaingoatsoftware.com/tag/story-points> 1/16/2012
6. IFPUG Bulletin Board; "Converting Use Case Points to IFPUG Function Points"; posted September 2006; retrieved 12/2011
7. Joe Schofield; "The Statistically Unreliable Nature of Lines of Code"; CrossTalk; April, 2005
8. <http://en.wikipedia.org/wiki/Fibonacci_number>; retrieved 1/24/2012
9. Jack Milunsky; The significance of Story points; March 20, 2009; retrieved from <http://agilesoftwaredevelopment.com/blog/jackmilunsky/significance-story-points> 12/2011;
10. Abdollah Zawari; "Project Estimation with Use Case Points using Enterprise Architect," retrieved from <http://community.sparxsystems.com/tutorials/project-management/project-estimation-use-case-points-using-enterprise-architect-ea> 12/2011
11. Remi-Armand Collaris and Eef Dekker; "Software Cost Estimation Using Use Case Points: Getting Use Case Transactions Straight", 03/13/2009; retrieved from <http://www.ibm.com/developerworks/rational/library/edge/09/mar09/collaris_dekker/> 12/2011
12. A. J. Albrecht, "Measuring Application Development Productivity," Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17, IBM Corporation (1979), pp. 83–92.
13. International Function Point Users Group (IFPUG); Counting Practices Manual, v 4.3; January, 2010
14. Annie Murphy Paul; "Good Guess – why we shouldn't underestimate the value of estimating"; TIME; (December 12, 2011): 68

NOTES

1. This includes treating the source of many dropdowns as customer-requested ILFs since the customer will maintain this data in a future release; thus the apparent disparity among ILFs and EIs.