Metrics Framework for Cycle-Time Reduction in Software Value Creation

Adapting Lean Startup for Established SaaS Feature Developers

Pasi Tyrväinen, Matti Saarikallio
Agora Center, Department of CS and IS
University of Jyväskylä, Finland
pasi.tyrvainen@jyu.fi, matti.saarikallio@gmail.com

Timo Aho, Timo Lehtonen, Rauno Paukeri
Solita plc
Tampere, Finland
{timo.aho, timo.lehtonen, rauno.paukeri}@solita.fi

Abstract— **Agile software development methodologies driving cycle-time reduction have been shown to improve efficiency, enable shorter lead times and place a stronger focus on customer needs. They are also moving the process development focus from cost-reduction towards value creation. Optimizing software development based on lean and agile principles requires tools and metrics to optimize against. We need a new set of metrics that measure the process up to the point of customer use and feedback. With these we can drive cycle time reduction and improve value focus. Recently the lean startup methodology has been promoting a similar approach within the startup context. In this paper, we develop and validate a cycle-time-based metric framework in the context of the software feature development process and provide the basis for fast feedback from customers. We report results on applying three metrics from the framework to improve the cycle-time of the development of features for a SaaS service.**

Keywords-metrics framework; cycle-time; agile; software engineering process; lean startup; feedback; SaaS.

I. INTRODUCTION

The software engineering (SWE) process has traditionally been managed on a cost basis by measuring programmer effort spent per lines of code, function point or requirement. These metrics have also been used to guide software process improvement. In order to align more with business strategy and value production the focus has shifted more towards value creation instead of cost reduction. For example, value-based SWE [1], software value-map [2] and a special issue on return on investment (ROI) in IEEE Software [3] have explored value in software development. As a reaction to move away from a cost-reduction focus, the recent goal of lean thinking has been to optimize for perceived customer value [4]. Thus, we can say that leadership approach for the software development process is moving from a cost focus to a value focus.

Measuring the value of application software and cloud services is difficult to do before it is in use, as you need to consider the value of the software for the potential users, the business value for the firm developing it and the value for other stakeholders [1][5][6]. The current theories of value do not present a simple way of assessing customer value [7]. Although companies put a great amount of effort into increasing customers' perceived value in the product development process, determining how and when value is added is still a challenge even in marketing and management sciences. [7] Further, the software engineering metrics are measuring attributes of the software development process (e.g., cost, effort, quality) while these metrics remain disconnected from the attributes and metrics developed for measuring value (see Table I). Various approaches have been developed to overcome this gap [1][5][6][8][9][10][11][12][13][14][15][16] without any major break-through.

The software engineering community has adopted an iterative approach to software development in form of Scrum [17], XP [18] and other agile [19] methods. These promote fast cycle user interaction and development process to keep the effort focused on customer needs based on fast customer feedback either interactively or through analysis of service use behavior. The startup community has adopted a similar approach and commonly uses the lean startup cycle [20] to evaluate the hypothesis of customer needs using the build-measure-learn cycle, which is repeated to improve customer acceptability of the offering and the business value of the startup. The common theme of these approaches is that instead of trying to estimate or predict the value in advance, try to shorten the cycle time from development to actual customer feedback, which indicates the value of the software in use. That is, from the SWE perspective, the speed of feedback received from users is the best indicator of the value of the newly created software. This indicates that shortening the feedback cycle would drive the SWE process towards faster reaction on customer value and higher value creation.

Although there exists a common understanding about the key role of a fast customer feedback cycle in linking the SWE process to value creation, the measurement methods and metrics available in literature are positioned either as cost-based SWE methods or as value-oriented metrics with little connection to the engineering process providing little guidance for managing and developing the SWE process (see Table I). Thus, the research question of this paper is, what metrics would guide cycle-time-driven software engineering process development in established organizations?

As the answer is context-dependent, a set of metrics will be needed. This paper aims at filling this gap by proposing a metrics framework enabling adoption of such metrics in a variety of contexts where new features are incrementally added to software.

TABLE I. POSITION OF THIS RESEARCH TO BRIDGE COST-ORIENTED SOFTWARE ENGINEERING (SWE) METRICS AND VALUE-ORIENTED BUSINESS METRICS

	Measurement Domains		
	SWE Metrics	***Research Gap Addressed Here***	***Value Metrics***
Scope (measurement target)	SWE Process	Value Creation Cycle	Customer Value of Offering, Value of Startup
Measured Attribute	Cost, Effort, Quality	Cycle Time	Value for Customer, Value for Enterprise
Examples	Function Points per month, Faults per lines of code		Value in Use, ROI, Lean Analytics

Applying the guidelines of the design science method [21], this research has been initiated based on company needs presented in interviews of Software as a Service (SaaS) development firms in a large industry-driven research program [22], to target an issues with business relevance in firms.

In Section II, we construct the metrics framework artifact based on the analysis and synthesis of previous research literature selected from the perspective of the research question. Following the design science research guidelines, we also demonstrate generalizability of the framework artifact to several contexts by choosing from a variety of metrics to target the specific process development needs. We also propose a simple diagrammatic representation for visualizing some of the metrics values in operational use to pinpoint development tracks requiring attention in an organization with multiple parallel feature-development teams.

In Section III, we evaluate the metrics framework by applying it to the case of a firm developing new features for an existing SaaS service and discuss the impact of the findings on revising the target of the next process improvement actions. In Section IV, we summarize the results, draw the conclusions and propose directions for further research.

II. THE CYCLE-TIME METRICS FRAMEWORK

A. Developing the Framework

The flow of new features through a SWE process can be measured at various points in time with an aim to reduce delay between points to reduce cycle time. The scope of the process measured will impact the attention of the software developing organization. In the narrowest scope, the cycle time measured includes the basic software development cycle while the widest cycle takes into account the customer needs and experience and, thus, matches and even expands the lean startup cycle [20].

In the proposed framework (see right side of Figure 1), the feature life-cycle begins with three planning phase events: 1) a need emerges, 2) a software development organization recognizes the need, and 3) the decision is made to develop the feature. In large established organizations, the identification of feature needs has been excluded from the responsibility of the SWE organization to responsibility of the product marketing organization, while the entrepreneurship-oriented startup community has emphasized the value of including the need identification step as an inherent part in the fast business development cycle of the organization developing the software. Sometimes there is an intentional lag between events 2 and 3 as the decision may be to wait for the right time window (cf. real options [23][24]), or features with higher priority are consuming all resources available.

Continuing from the 3 events that form the beginning of the feature life cycle (above) and for the purposes of measuring the value creation cycle, the main development events included in this framework are 4) development starts, 5) development done, and 6) feature deployed. Use of XP, Scrum and other iterative and incremental development (IID) processes has aimed at reducing the time between events 4 and 5 (or fixing that to 2–4-week cycles). The cycle-time from 4 to 5 is here referred to as the Development cycle (see Figure 1). Moving from packaged software to cloud delivery and SaaS development along with moving from an annual or a six-month software release cycle to continuous integration (CI [25]) and continuous delivery (CD [26]) in development operations (devops [27]) has reduced the interval between 4 and 6.

After the event 6, the traditional software engineering process is often thought to be completed, while many entrepreneurship-oriented approaches, such as Lean Startup [20], go further, starting from building a product to measuring the use of it, which produces data used for learning and for producing ideas for the next development cycle (see left side of Figure 1). That is, building the product based on current ideas is only one of the three main events needed for value creation: build–measure–learn [20]. For considering the business and customer perspectives in this metrics framework for the value creation cycle, we need to expand beyond step 6 to include the use, measuring and learning phases: 7) when the feature gets used, 8) when feedback data is collected to support learning, and 9) when a decision is made based on the feedback. Note that events 8 and 9 resemble events 2 and 3 while not all information from customer needs is collected through measuring the use of the current product. It is also commonly assumed that the time from feature deployed (6) to first use (7) is short, while without measured data this can be an incorrect assumption. There have been cases where almost half of software features were never used [28]. Further, if software quality is high, it can take some time to get feedback, and it may require many uses of the feature before customer sends feedback about problems. Additionally, it can take time for a feature to get sufficient number of uses to allow for a reliable analysis of customer behavior (8). Also, the deployment process of the company can delay the decision to act on the feedback (9).

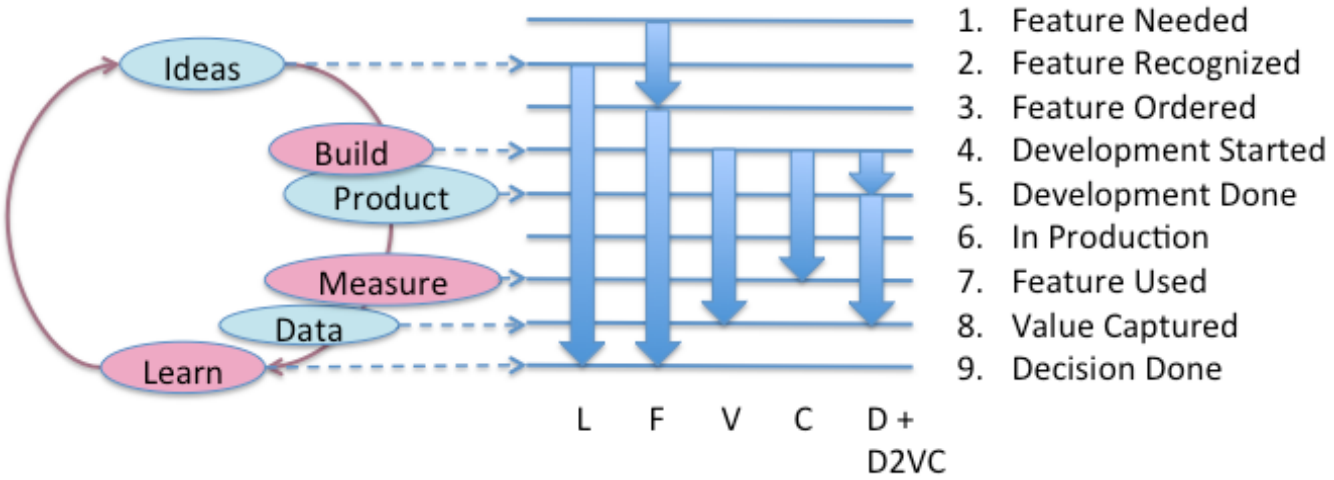

Figure 1. The value-driven metrics framework for driving software engineering cycle-time reduction (on the right), the Lean Startup cycle (on the left) and example cycles, for which cycle time can be used as the metrics driving cycle-time reduction (in the middle).

Figure 1 depicts the proposed framework. On the right side we have the sequence of events identified. On the left side, we have the Lean Startup cycle with horizontal arrows pointing from the phases to related events of the framework. The vertical arrows in the middle represent examples of cycle times that can be used as a target metric for developing SWE process. The cycles in the center are labeled as follows: L = Lean Startup cycle, F = Full cycle including fuzzy front end and full feature development cycle, V = Value cycle from starting the development to value capture, C = Core cycle from development start to first feature use, and finally D+D2VC, where D = Development cycle from start of development to production readiness and D2VC = time from development done to value captured. We emphasize that this list of cycles is not exclusive and new cycle time metrics can be created with this framework on demand for each context.

B. *Changing Process Development Focus through Metrics*

The various cycle-time metrics available in the framework can be used for focusing process development activity to specific process areas based on the need (see Table II). For example, if the basic software development process has been well developed and if some incremental development process, automated testing and continuous integration are applied, it may be useful to shift the attention to continuous deployment. In that case, the metric to be followed can be changed from Development cycle to cycle time between events 4 and 6, from start of development to start of production (see the second line in Table II). Changing the metric will also change the focus of attention and can often result in adjusting the processes, resource allocations or tools used.

TABLE II. EXAMPLE PROCESS DEVELOPMENT TARGETS WHEN USING ALTERNATIVE CYCLE-TIME METRICS

Cycle	Start Event	End Event	Addressed Capabilities	Process Development Focus
D, Development	4: Development Started	5: Development Done	XP, Scrum and other IID processes, automated testing and continuous integration (CI)	Using this cycle-time metrics addresses cycle-time of the basic SW development process
Time to production	4: Development Started	6: In Production	Same as in D, adding continuous deployment (CD) to the measurement scope	Using metrics for this cycle time focuses attention to CD capability
C, Core cycle	4: Development Started	7: Feature Used	Same as previous adding communication (diffusion) to customer base to the scope	Here the focus shifts to integrating customer facing team with development
V, Value cycle	4: Development Started	8: Value Captured	Adding customer analytics and customer feed-back capabilities to the previous scope	Shifts focus to integrating analytics capability to IID+CI+CD capability
Time to Value	4: Development Started	*: Break Even	As Value cycle, but using this metrics assumes that value produced can be evaluated.	As in Value cycle
D2VC	5: Development Done	8: Value Captured	Post-development processes needed to deliver the created value and to get the feedback	Focusing on value cycle capabilities after the basic SW development process.
Fuzzy Front End	1: Feature Needed	3: Feature Ordered	Deep customer understanding (between events 1 and 2) and market understanding (2 to 3)	Measuring capability to find customer needs close to actionable market
...	...	...	...	...

In large organizations, where the product-marketing department is responsible for collecting market requirements and for product launches, the processes crossing product development and product-marketing departments may be problematic. In these cases, choosing the Value cycle, Time to Value or Design done to value captured (D2VC) as a common metric for both of the departments will enforce collaboration between the departments and will likely improve the total value creation capability of the organization, while local metrics within the departments are likely to lead to local optimization leading to non-optimal organizational behavior. It should be noted, that this issue appears mainly in large established organizations rather than in small startup firms, the needs of which the lean startup approach has been developed.

The time to value cycle in Table II ends with the event of reaching the breakeven point, which is marked with an asterisk "*" rather than a number representing a specific ordering in the framework. In some cases a pay-per-use business model provides a basis to determine the break-even point for a feature, while in some cases the break-even point is estimated by qualitative means. A new feature may produce enough value to reach the break-even point when it is published (event 6) or when it is used for the first time (event 7). However, in many contexts this event occurs close to event 8, Value captured, that is, the feature use count is high enough, and sufficient feedback has been received, to ascertain whether the feature was worth the development effort. Based on these examples and the other examples in Table II we can observe, that the choice of applicable metrics is context dependent. Thus there is a need for a framework for metrics, which supports choosing the metric applicable for a specific situation.

C. *Depicting Cycle-Time Elements*

Depicting the proposed cycle-time metrics makes it easier to decide whether to further develop or even to drop a existing feature and will also help in communicating the cycle-time reduction agenda to software engineers and other parties involved. For this purpose we devised a simple diagrammatic representation presented in Figure 2. In this example, the development starts at point 4 and ends at point 5. The y-coordinate represents the cumulative development time, in line with the cumulative cost for the organization. This linear curve is intentionally simplistic as the focus is on the form of the curve after event 5. In contrast, software engineering oriented representations, such as the Kanban Cumulative Flow Diagram [29], focus on analysis of the development cycle from 4 to 5 and ignore activity after production readiness.

From event 5 on, the horizontal line represents the duration of the waiting time from ready-to-deploy through deployment to first use. The feature is used for the first time in production at event 7. After that the dropping logarithmic curve represents the speed at which feedback has been received. After the second use the curve comes down to half, after the third use to one third of the original, and so on.

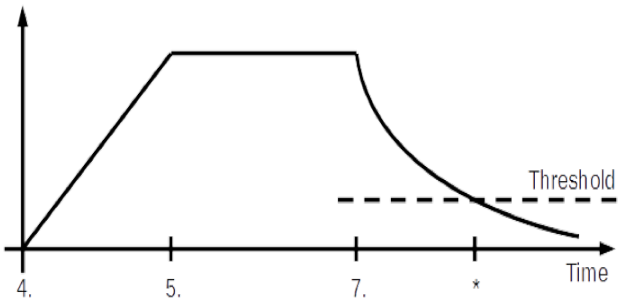

Figure 2. Depiction of the cycle times for feature analysis and process development. The numbers refer to the event number in the framework.

That is, the curve represents development time divided by number of times used. A context-specific target threshold for development time per times used is presented as a dotted line and the time when the curve reaches the threshold is marked with an asterisk "*".

In line with our approach to focus on the cycle times, this graphical representation aims at depicting the cumulative effort invested to the feature during development. There is a risk embedded in this development effort as it has not received feedback from the customers. Thus it is potential waste if customers do not accept the feature. This risk is mitigated along the narrowing gap of the asymptotic curve and the horizontal axis and reaching the threshold indicates that enough customer feedback has been received to ascertain whether it has been worth developing the feature. Event 8, Value captured, is serving this purpose as the event when sufficient user feedback is gained to evaluate the value of the newly developed feature and for adjusting the development plans accordingly, to further develop the feature or to drop it. In addition to guiding value creation, fast feedback from event 8 makes it easier for software developers to fix errors and modify the feature as long as they can still recall the implementation of the feature and have not moved on to new assignments.

Although measuring value is difficult, we would also like to identify the time-to-value cycle, that is the time from starting the development to break even, to the point at which its value to the customer exceeds the development costs. Now we face the challenge that while the cost can be easily measured in terms of time or money, value as a concept is not clearly defined and even if it were it would be hard to measure. We can speculate that the break-even point could be reached already on deployment (for features whose existence provides value even if they are not used, e.g., emergency-situation feature), on first use (when customer finds it), after a certain amount of uses (some use value derived from each), or sometimes a feature can fail to become profitable. Thus, the location of this measurement point cannot, in general, be identified in the sequence of events in the proposed framework, rather it is context dependent.

If we want to measure value, we need to define value. Historically three forms of economic value are the use value, exchange value and price [30]. There are many theoretical divisions of value to support decision making about which

software feature to work on next [2][5][7][8][9][10][11][12][13][14][15][16][24][30][31], but most theories consider the use value to the customer as essential. For the purposes of metrics development the focus will be on customer use value. It is important to note that due to market mechanism, exchange value is less or equal to use value [30]. This means that we could calculate a monetary estimate for the upper bound of the value captured by the software developer, that can be compared with cost. Still, the issue is problematic.

If we assume that there is use value for a feature, and in some cases the use value can be estimated as equal for each use, we would like to measure directly the cost versus benefits ratio: $\frac{development\ costs}{benefits}$. However, as discussed the benefits are challenging to measure and, at worst, we might need a new metric for each feature. This leads us to suggest that we isolate the hard-to-measure part, benefits, by instead measuring the precisely calculable cost per use $\beta = \frac{development\ costs}{times\ used}$ and only if possible compare it to the estimated value for the user, based on a case-by-base estimation method. Next, we will show, using a case study, that reaching events "*" and 8 produce very similar value for process development and feature decision making and that they can be used interchangeably. Thus, time to receive enough feedback is also a good, practical proxy for value produced.

III. Metrics Validation Case Study

A. Target Organization and Service

We evaluated the metrics framework in a mid-sized Finnish software company, Solita Ltd. The case software development team develops a publicly available SaaS (lupapiste.fi) used by citizen applying for a construction permit related to real estate and other structures. This privately operated intermediary service provides a digital alternative to avoid the time-consuming paper-based process of dealing directly with the public authority. This service is used by employees of the licensing authority in the municipalities (about 100 users), the applicants (citizens and companies, about 100 per month), and architects and other consultants (1-2 per application). The software development process metrics were evaluated with the usage data collected from the process flow of five new features of this SaaS service deployed during the observation period, in mid-2014.

The service has a single page front-end that connects through a RESTful API to its back-end. Each call to the API is recorded on the production log files with a time stamp. We mined and analyzed the log entries together with the development data captured by the version control system. In this case, we chose features that introduced a new service to the API and were thus possible to trace automatically with a simple script that queried the monitoring system automatically. Some manual work was needed to find the features that introduced a new API, but automation of this work is also possible.

B. Results from Applying the Metrics to Sample Features

From the recorded event time stamps we calculated three metrics values for the case features. Development cycle (D) from start (4) to done (5) in working days. Lag to production from done (5) to deployed (6) in calendar days. And finally, D2VC, time from development done (5) to value captured (8). In this context the target company estimated that enough feedback data was collected for learning when the feature was used four times per each day spent on development, which gave the context-specific definition for the value capture event (8). Table III presents the data that is depicted in Figure 3. To enable comparison, all the features are shifted in the time axis to have event 4 (start of development) at day 0. In a daily use, an alternative depiction can show the timeline representing the history of all features to current point of time from which it is easy to identify development peaks and, more specifically, to notice the curves that remain high after the peak which indicates a demand for action. Either a feature has not been deployed and promoted well for the users or there is no user need for the feature.

C. Case Analysis and Discussion

From Table III we can see that for these five features the average of development effort needed to implement and test the features was about eight working days. When the development was done, on an average 12 calendar days was spent on waiting for deployment of the feature to the production environment. We can also observe that the features with lower priority (F1647 Unsubscribe and F1332 Note) have almost double the lag to production compared to the other features.

TABLE III. Description of the sample features, their priority, development time (in working days), lag to deployment (in calendar days) and Development to Value Captured (D2VC; in calendar days)

Feature id	Priority	Development (days)	Lag to deployment (days)	D2VC (days)	Description of the feature
F1332 Note	2	10	24	24	Authority user can add a textual note that other users cannot see.
F1498 Attachment	4	9	10	N/A	Applicant user can set the target of an uploaded attachment.
F1507 Validate	4	10	1	49	System validates the form prior the user sends the application.
F1537 Sign	4	7	11	15	Authority user can require an applicant to sign a verdict.
F1647 Unsubscribe	3	2	15	28	Authority user can unsubscribe emails.
Average		**8**	**12**	**29**	

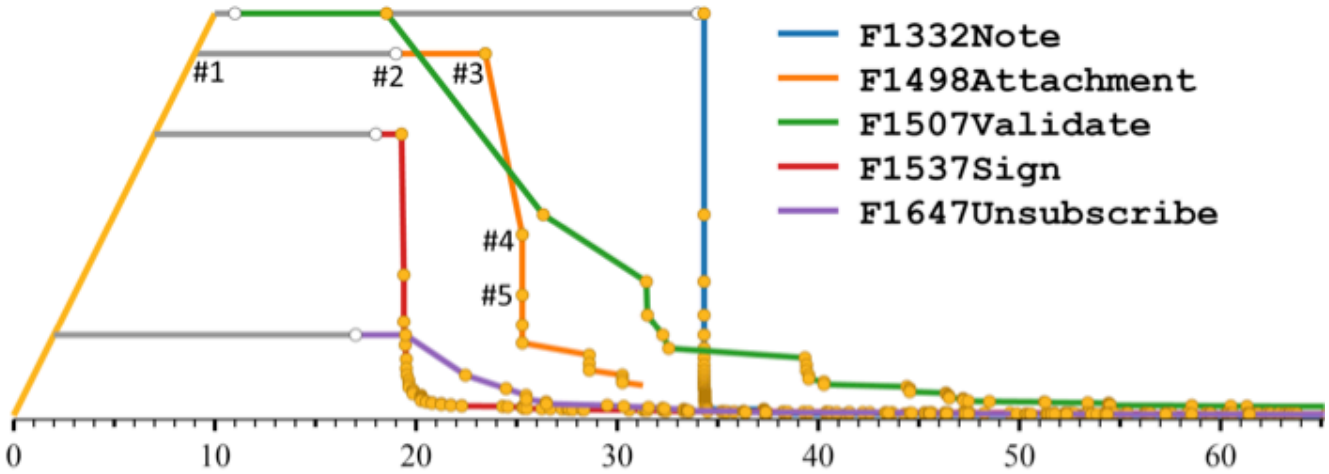

Figure 3. Depiction of the cycle-times of the five features. Development working days share the rising line starting from (event 5) and end in event 6 (start of the gray horizontal line), deployment (7, white dots in the right end of the gray part of the horizontal lines) and usage (yellow dots). To enable comparison, all the features are synced to have event 5 (start of development) at day 0.

The average time from completion of development to value capture is 29 days (this does not include feature F1498 Attachment, which did not reached the number of uses needed for the threshold). From the depiction in Figure 3, we can also see that this feature is no longer used. This feedback triggers the discussion on the reasons for the discontinuation of use of the feature to determine if there is a need to improve it or remove it from the service. When the target is to minimize the cycle times, minimizing the lag from production readiness to deployment (from event 5 to 6) and the means to increase the use of new features are clearly the places where major improvement can be reached much easier than from reducing average development time. By plotting the events in this way, it is easy to identify the places where changes can be made as well as to communicate the need with the development teams.

The results triggered also a discussion on the release practices of the firm. From the service use statistics it is possible to see that the service is heavily used from Monday to Thursday, less on Friday and very little during weekends. Thus it is likely that features released on Mondays will get used sooner than the ones released on Fridays, which provides the additional benefit that the feedback from users (8) would reach the developers when they still recall the software they were working on. Even more profound than the weekly cycle is a similar variation related to the vacation seasons. Deploying new features just prior to vacations will have negative impact on the Value cycle, as described above.

IV. Summary, Conclusions and Further Research

The feedback from practitioners suggests that the current literature lacks metrics that could be used for directing a software development organization from the business perspective to enhance effective value creation and value capture. Although the Lean Startup Methodology proposes to develop the software via the build–measure–learn cycle, we seem to lack the means to measure the value that the delivered software creates. Also the researchers have observed this problem and conclude [7], that the current theories of value do not present a simple way of assessing customer-perceived value. Although companies put a great amount of effort into increasing customers' perceived value in the product development process, determining how and when value is added is still a challenge even in marketing and management sciences [7]. Previous literature on XP, Scrum, lean startup and related approaches has indicated that in the context of SaaS services, delivering new versions of a service to the customer, collecting the usage data and making further decisions based on the data provides the most promising path for the software vendor to understand customer-perceived value. Agile methods have been shown to enable shorter lead times and a stronger focus on customer needs [32].

Shortening the cycle times provides increased flexibility maintaining options to change development direction with speed [20][22] as well as other business benefits for software service firms. This encouraged us to search for metrics that help software firms in the process development towards shorter cycles. On this basis, we formulated the research question as, what metrics would guide the cycle-time-driven software engineering process development in established organizations?

As the proposed solution, we adopted and extended the lean startup [20] value creation cycle and constructed a framework for metrics based on the times between main observable events within the cycle, all the way through to receiving and analyzing user feedback. This focuses attention on fast execution of the value creation within the user feedback cycle. That is, we are not trying to measure value of the results of the cycle, such as the value of the product produced or the value of the startup or progress of the startup in creating the offering, as in lean analytics [12].

By finding the measurable values from within the value creation cycle, the cycle-time metrics framework aims at bridging the gap between cost-oriented SWE metrics and value-oriented business metrics. Cycle-time reduction serves as the intermediary of increased value creation guiding software feature development and software process development. The metrics measure the calendar time between the key events. The first three events are related to feature need identification and the business decision to implement the specific feature (event 3). The core events following this decision are start of the development (4), the feature is ready for deployment (5), the feature is deployed (released, 6), and first use of the feature by a customer (7). These events are followed by feedback related events, the feature feedback data has been collected and analyzed (8) and a decision is made based on the feedback (9). The time intervals between the core events (4-7) are of most interest for the engineering while the other events (1-3 and 8-9) relate to the customer-perceived value analysis of the feature. We also provided examples on how changing the measurement cycle directs the process development to new process areas.

Our empirical focus was at the level of features being added to an existing SaaS offering. In the empirical part, the times between the events in the core cycle were measured for five new features in the development processes of an independent software vendor's SaaS service. The results showed that the core metrics were able to capture and bring up useful characteristics of the business process that triggered both a "drop vs. develop feature" discussion (for feature F1498) and a number of process development discussions. In these five feature development cases the average development time was shorter than the waiting time for the feature to be released. This has negative impact to the efficiency of fixing potential problems emerging during the first uses of the feature by first users, as the developers have already oriented towards another assignment. The detection of the delay of feature releases lead to a further analysis of the vendor's release practices in general and prompted quick improvements to their process.

Although the results from the empirical part showed that the metrics are useful in practice, there are still several avenues of further research that we wish to explore. The empirical part used data from the engineering system and customer feedback data to identify the core events. This seems to be a useful starting point and the firm in our case study would like to extend the collection of data to cover as many of the nine events as possible and as automatically as possible. The time from release readiness to analyzed customer feedback seems to be a particularly useful measurement of deployment performance.

In general, collecting the data can and should be automated using engineering information systems to the extent possible (events 1 and 8 cannot be detected automatically). For the other events, we propose collecting and depicting the data graphically in real-time status displays providing an overview of the development activities for business and engineering management. As we can observe from the empirical case, the results are useful both for focusing process development activities and for making business decisions regarding which features will be developed further, which will be used as they are, and which features will be removed from the service. This way the simplified depiction can provide transparency between the business and the development organization. Thus we encourage further empirical work on the automation of data collection and its depiction based on events identified in the framework.

In startups the result of value creation cycle can be analyzed in the context of the evolution of the enterprise [12]. In context of established feature development processes, this framework adopted the approach of using only cycle times between events as the metrics within the value creation cycle. This is due to limited applicability of suitable previous research results for real-time customer-perceived value analysis beyond A/B testing and similar tools that can be used between events 7 and 8. Although cycle time metrics seems to provide high added value to focus process development in connecting software development with customer value, investigating the value capture events 8 and "*" further is needed. Finding an easy to apply means for estimating the perceived user benefits would enable various new developments supporting the operative business development of a software engineering team.

ACKNOWLEDGMENT

This work was supported by TEKES as part of the Need for Speed (N4S) Program of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT and digital business).

REFERENCES

[1] B. W. Boehm, "Value-based software engineering: Overview and agenda," in Value-based software engineering, Springer Berlin Heidelberg, 2006, pp. 3-14

[2] M. Khurum, T. Gorschek, M. Wilson, "The software value map - an exhaustive collection of value aspects for the development of software intensive products," Journal of software: evolution and process. Wiley, 2012, 711-741.

[3] H. Erdogmus, J. Favaro, W. Strigel, "Return on investment," IEEE Software 3(21), 2004, pp. 18–22.

[4] K. Conboy, "Agility from first principles: reconstructing the concept of agility in information systems development," Information Systems Research, 20(3), 2009, pp. 329-354.

[5] S. Barney, A. Aurum, C. Wohlin, "A product management challenge: Creating software product value through requirements selection," Journal of Systems Architecture, 54(6), 2008, pp. 576-593.

[6] A. Fabijan, H, Holström Olsson, J. Bosh, "Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review," in Software Business. Springer International Publishing, 2015, pp. 139-153.

[7] J. Gordijn, and J.M. Akkermans, "Value-based requirements engineering: exploring innovative e-commerce ideas," Requirements Engineering 8(2), 2003, pp. 114-134.

[8] M. Rönkkö, C. Frûhwirth, S. Biffl, "Integrating Value and Utility Concepts into a Value Decomposition Model for

Value-Based Software Engineering," PROFES 2009, Springer-Verlag, LNBIP 32, 2009, pp. 362–374.

[9] R.B. Woodruff, and F.S. Gardial, Know your customer: New approaches to customer value and satisfaction. Cambridge, MA, Blackwell, 1996.

[10] C. Grönroos, "Value-driven relational marketing: from products to resources and competencies," Journal of Marketing Management 13(5), 1997, pp. 407–419.

[11] T. Woodall, "Conceptualising 'value for the customer': An attributional, structural, and dispositional analysis," Academy of Marketing Science Review, no. 12, 2003, pp. 1526–1749.

[12] A. Croll, and B. Yoskovitz, Lean Analytics: Use Data to Build a Better Startup Faste,. O'Reilly Media, Inc. 2013.

[13] P. Tyrväinen, and J. Selin, "How to sell SaaS: a model for main factors of marketing and selling software-as-a-service," in: Software Business, Springer, Berlin Heidelberg, 2011, pp. 2-16.

[14] V. Mandić, V. Basili, L. Harjumaa, M. Oivo, J. Markkula, "Utilizing GQM+ Strategies for business value analysis: An approach for evaluating business goals," The 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, 2010.

[15] M. Saarikallio, and P. Tyrväinen, "Following the Money: Revenue Stream Constituents in Case of Within-firm Variation," in: Software Business. Springer International Publishing, 2014, pp. 88-99.

[16] J. Bosch, "Building products as innovation experiment systems," in: Software Business, Springer, Berlin Heidelberg, 2012, pp. 27-39.

[17] K. Schwaber, and M. Beedle, Agile Software Development with SCRUM, Prentice Hall, 2002.

[18] K. Beck, Extreme Programming Explained: Embrace Change. Addison-Wesley, 1999.

[19] A. Cockburn, Agile Software Development, 1st edition, 256 p. Addison-Wesley Professional, December 2001.

[20] E. Ries, The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Publishing Group, 2011.

[21] A.R. Hevner, S.T. March, J. Park, S. Rami, "Design Science in Information Systems Research," MIS Quarterly, Vol. 28, No. 1, 2004, pp. 75-105.

[22] J. Järvinen, T. Huomo, T. Mikkonen, P. Tyrväinen, "From Agile Software Development to Mercury Business," in: Software Business. Towards Continuous Value Delivery, Springer Berlin Heidelberg, LNIB, vol. 182, 2014, pp 58-71.

[23] H. Erdogmus, and J. Favaro, "Keep your options open: Extreme programming and the economics of flexibility," in Giancario Succi, James Donovan Wells and Laurie Williams," Extreme Programming Perspectives", Addison Wesley, 2002.

[24] M. Brydon, "Evaluating strategic options using decision-theoretic planning," Information Technology and Management 7, 2006, pp. 35–49.

[25] M. Fowler, Continuous Integration, 2006. http://martinfowler.com/articles/continuousIntegration.html retrieved: Septmeber, 2015.

[26] J. Humble, and D. Farley, Continuous delivery: reliable software releases through build, test, and deployment automation, Pearson Education, Jul 27, 2010.

[27] P. Debois, "Devops: A software revolution in the making," Cutter IT Journal, vol. 24, no. 8, August, 2011.

[28] J. Johanson, Standish Group Study, presenation at XP2002.

[29] K. Petersen, and C. Wohlin. "Measuring the flow in lean software development." Software: Practice and experience, vol. 41, no. 9, 2011, pp. 975-996.

[30] J.S.Mill, Principles of political economy, 1848, abr. ed., J.L.Laughlin, 1885.

[31] M. Cohn, Agile estimating and planning. Pearson Education Inc. 2006.

[32] M. Poppendieck and M.A. Cusumano, "Lean software development: A tutorial," Software, IEEE, vol. 29, no. 5, 2012, pp. 26–32.