Functional Size Measurement in Agile

A Thought Experiment with Measuring Functional Size in Agile Development

Dr. Thomas Fehlmann
Euro Project Office AG
8032 Zurich, Switzerland
E-mail: thomas.fehlmann@e-p-o.com

Eberhard Kranich
Euro Project Office
47051 Duisburg, Germany
E-mail: eberhard.kranich@e-p-o.com

Abstract — **In Agile Software Development, story points indicate the effort needed to implement a user story up to the Definition of Done. Hence, story points can be applied to track the progress of a software product under development. A major drawback of their use is that they do not allow predicting the number of sprints needed to create or modify a software product, not even for a minimum viable product. A more promising way to measure sprints is to use functional size counts determined by IFPUG or COSMIC. Both methods yield useful results when correctly interpreted. However, the functional size of the product is not simply determined by the total functionality implemented in sprints. Agile teams often touch the same functionality more than once; adding new requirements to existing functionality must be handled adequately, and some already implemented functionality is disregarded. Moreover, refactoring, removing technical debt and software testing adds effort, measured in story points, but adds no functionality.**

Keywords—Agile Software Development; Software Metrics; Software Sizing; Sprint Performance; Product Cost Management; Thought Experiment; Gedankenexperiment.

I. Introduction

A *Software Metric* is a measure of some property of a piece of a software artifact or its specifications. A measure relies on a measurement method that fulfills the definition of the VIM and the GUM:

- The VIM: ISO/IEC Guide 99:2007, 2007. International Vocabulary of Metrology – Basic and general concepts and associated terms (VIM) [1];
- The GUM: ISO/IEC CD Guide 98-3, 2015. Evaluation of measurement data – Part 3: Guide to Uncertainty in Measurement (GUM) [2].

ISO 19761 COSMIC [3] defines a software metric; ISO 20926 IFPUG [4] a size count. Both methods are useful for the purpose of measuring sprint performance in Agile.

A. Combining Measurements

One can add, subtract, compare, and multiply measurements for instance to combine part measurements into a measurement of the total. Sometimes, for instance when measuring a distance between two places, trigonometry is needed to combine two section measurements into one measure for the whole distance because of hills, slopes and angles.

Counting function points to characterize functional size, or software development effort, or something, are not necessarily metrics. Counting points does not measure anything else except points unless the points mark some unit on a measurement scale.

B. Measurement Methods

ISO 19761 COSMIC [5] measures size by counting data movements. Two measured applications can be combined into one by simply adding their counts. This works because the system boundary does not impact the count; contrary to IFPUG. ISO 20926 IFPUG [6] has five elementary units whose counting value depends on the boundary, because of the *File Types Referenced* (FTR). Adding two applications yields questions: what to do if the *Internal Logical File* (ILF) of one application becomes an *External Interface File* (EIF) of the other? What if a new requirement adds some *Data Element Type* (DET) to an elementary data unit? Does it affect all previous counts? There does not seem to be an easy way to combine two IFPUG counts and get the correct count for the combined application. IFPUG does not comply with the VIM and the GUM, whereas COSMIC does.

However, compliance is needed when counting each sprint and trying to get a valid size estimate of the total product by the sum. Each sprint creates a mini application that – theoretically – should already provide value to the customer and provide some functionality. The next sprint adds new functionality, changes existing functionality, and even might remove some already obsolete functionality. A set of stated requirements often varies because during agile software development some requirements can be removed from the backlog, whereas new requirements are added. In addition, modified requirements remain in the backlog.

In COSMIC, these activities can be modeled, basically by distinguishing data movements created anew from those amended or enhanced. The total of data movements, new developed, enhanced, or re-developed, describes the size of the product at any given moment in time – i.e., at the beginning of a sprint – while the performance of a sprint should be measured by counting all data movements touched, be it the total of newly created, enhanced, re-developed, or deleted. This implies that data movements count every time they are touched for the sprint, but only once for size. Since there are usually different product delivery rates for new development and enhancement [7], one can compare such metrics with performance data from other software development undertakings.

With IFPUG, modeling the elementary data functions and transactions also allow sizing the product, or the sprint. However, these sizes do not add easily but still approximate performance of a sprint.

For detailed information about these measurement methods, consult the manuals (COSMIC [3] and IFPUG [4]) and the ISO standards (COSMIC [5] and IFPUG [6]). Fehlmann [8, p. 130ff] provides a comparison for using these methods. When to use which method depends from the application or business domain. For transactional systems, use IFPUG; for *Internet of Things* (IoT), communication, and service architectures, developers prefer COSMIC; compare with Fehlmann [9].

C. *Research Questions*

While the term 'project' has disappeared from Agile [10] and DevOps [11], the old lore about projects regarding quality, effort and time constraints remains valid. The same laws hold for agile sprints. *Story Points* [12] are widely in use to predict the content of the next sprint, based on a team's effort prediction. There is no distinction between *Functional User Requirements* (FUR) and *Non-functional Requirements* (NFR) [13]. Consequently, story points are not suitable for benchmarking, even if some sort of standard story points are defined for comparison between different teams.

The question is whether functional sizing conformant to ISO/IEC 14143 [14] provides value to agile teams. Some developers will deny that, with the argument that function point sizing traditionally has been used to predict the size of the final product. But agile development works without a known finished product, therefore no such size can be determined.

Nevertheless, there is a need to assess and predict cost of development, operation, and maintenance for software. Traditionally, measuring functional size and benchmarking has been used for predicting operational and maintenance cost [7]. Why should this not be possible for agile sprints?

To make functional size measurement useful for agile, we need to answer the following research questions, for both COSMIC and IFPUG-based size measurements:

- What exactly to measure?
- When to measure?
- How exactly to measure?
- How does the final product relate to sprints?
- How does functional size relate to story points?
- How many sprints are needed to build a *Minimum Viable Product* (MVP)? [11]

To answer these questions, we conduct a *Gedankenexperiment* addressing mobile app development, closely based on actual experiences.

We introduce the two methods for measuring functional size, then describe the approach and the sprints measured, then do a Retrospective and finally present the findings.

II. The Measurement Approach

Per sprint we execute two distinct functional size measurements in parallel:

- Functional size of each sprint, corresponding to work performed per sprint, and therefore to the development team performance;
- Total product size, corresponding to the total value of work performed that contribute to the product in use.

The total sum of sizes of all sprints is not equal to the total product size, else it would not be agile development.

This is due to new requirements that change functionality that already had been implemented. Also, some functionality might initially be implemented in a provisional way. Still, such work had been performed in the sprints, and it would not be correct to discard such work as "non-productive". Sometimes, the initial functionality was necessary to uncover the correct requirements; sometimes, such functionality was a stub that made the piece of software valuable to the user in an initial, provisional state. Examples include automated data connectivity that initially was substituted by some manual input facility.

A. *Data Movement Maps*

For measurement, we use *Data Movement Maps* and *Transaction Maps*. From data movement maps, a COSMIC count can be obtained automatically; from a transaction map, an IFPUG count. The automated counts are reasonably good approximations, enough for the purpose of monitoring agile sprints.

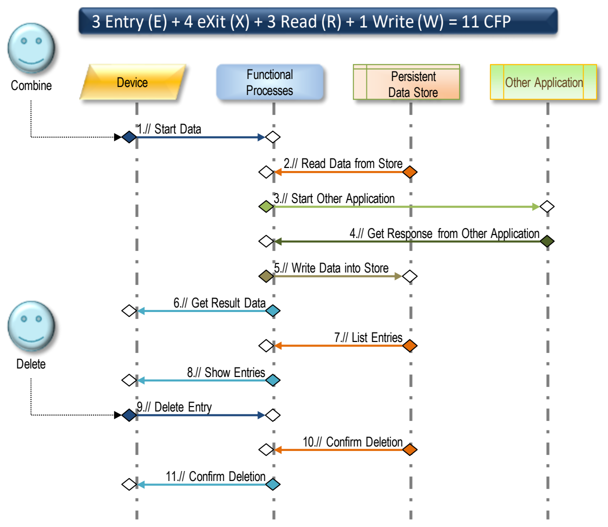


Fig. 1. Data Movement Map with Four Objects of Interest and two Triggers

How to create these maps and its automatic counting, consult Fehlmann [8, p. 130 & 160]. We distinguish four types of *Objects of Interest*:

- Functional Processes: Objects that perform functional processes in the COSMIC sense. One object of interest can perform more than one functional process; thus, it represents for instance one Virtual Machine (VM), or Electronic Control Unit (ECU) performing different calculations rather than a single COSMIC functional process;

- Persistent Store: Objects that persistently hold data. Contrary to the COSMIC definition, they can provide data services to several functional processes;
- Devices: A device can be a system user or anything providing or consuming data;
- Other Applications: other applications use functional processes the same way as devices do, however, they typically represent other software or systems that can be modeled the same way using data movement maps.

As shown in Fig. 1, *Triggers* indicate the starting data movement of one or more COSMIC functional processes. Thus, one object accommodating several functional processes can have multiple triggers. The automatically calculated total count appears on the top.

Data Movements always move a *Data Group*, which can be thought as a data record. Its uniqueness is indicated by color-filled trapezes. A second move of the same data group between the same objects within a COSMIC functional process lets it blank, because it does not add any additional functionality. It is not counted for functional size according the COSMIC method [3]**.**

Data movement maps can automatically be counted for ISO/IEC 19761 COSMIC [3] functional size. Moving the same data group twice between the same objects of interest is counted as one function point only. On the other hand, one can combine as many data movement maps as possible and count the same total of data movements, notwithstanding how the boundaries are drawn.

B. *Transaction Maps*

The IFPUG model [4] defines a count for functional size by counting model elements that are conceptually familiar to traditional mainframe software: Elementary Data Functions and Elementary Transactions.

The following five functional components of the software evaluate for the count according to the ISO/IEC 20926 IFPUG rules based on the user requirements:

- Internal Logical File (ILF): A user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through External Inputs.
- External Interface File (EIF): A user recognizable group of logically related data or control information referenced by the application being measured; however, maintained within the boundary of another application.
- External Input (EI): An elementary process in which data crosses the boundary from outside to inside. The data can be either control information or business information. If the data is business information, it maintains one or more internal logical files. If the data is control, it does not have to update an internal logical file.
- External Output (EO): An elementary process in which derived data passes across the boundary from inside to outside. The data creates reports or output files sent to other applications. These reports and files originate from one or more internal logical files and external interface file.
- External Inquiry (EQ): An elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files. This information crosses the application boundary. The input process does not update any Internal Logical Files and the output side does not contain derived data.

For counting, these five elementary types of data functions or transaction are categorized as either low, medium, or high complexity. This yields its functional size. The complexity depends on the *Data Element Type* (DET) handled by each element, and the number of *File Type Referenced* (FTR). Consequently, ISO/IEC 20926 IFPUG defines a count with jumps, not a metric.

Adding data elements can let the complexity assessment jump from one level into another. Moreover, replacing functionality is sometimes not reflected in the count.

For counting model elements in ISO/IEC 20926, it is necessary to know the boundary for the complete system. The reason is that the total number of FTR – represented as connectors in data transaction maps – impact the size of the transaction-type model elements. Without knowing the whole system, parts cannot be counted, if following the rules of the IFPUG manual [4] exactly.

As already mentioned, the IFPUG count does not conform to the VIM and the GUM. This makes the IFPUG counting method unattractive both for agile software development that needs to count the functional size of sprints, and even more for test metrics. Any such metric relies on the VIM and the GUM when comparing size of sprints, adding sprints to the whole product, or when sizing test cases.

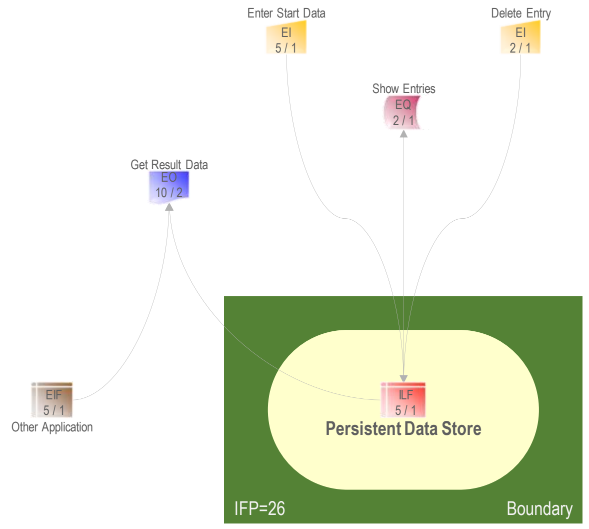


Fig. 2. Transaction Map for the Piece of Software Already Shown in Fig. 1

Transaction Maps, as shown in Fig. 2, are a way to visualize the IFPUG model for a software system. Depending upon the architecture, more than one transaction map is needed for a modern architecture software system. Then, typically, some

ILF has its data managed by one transaction map while others access the same elementary data elements as an EIF. Because of the boundary rules in IFPUG counting rules [4], this leads to double counting. The automatically calculated total count is shown at the bottom.

Both, data movement maps and transaction maps are well suited for use with agile teams, for visualizing which elements of software are touched in each sprint. The model elements can be used for communicating work done in sprints, at the same time providing its functional size. Business people usually prefer the transaction maps; developers the data movement maps.

III. An App Development Example

Software development typically starts with a vision, often described by an initial backlog of user stories – sometimes, rather *Epics*; that are functional user requirements in a granularity way above what is needed for implementation and coding. Usually, epics evolve into user stories during initial sprints. A vision is not what will be implemented at some later time – it is the idea of the vision that it remains a vision but changes while requirements get better understood and change as well. Comparing the initial vision with the product released later, after enough sprints, is nevertheless interesting and helpful for product owners and requirements engineers alike.

A. *The Vision*

The vision consisted of eight rather simple user stories, based around the introduction of barcodes allowing for scanning paper bills by an ordinary smartphone, creating transaction that a bank can execute.

TABLE 1. INITIAL BACKLOG USER STORIES FOR THE ANDROID MOBILE APP

Label	As a…	I want to…	Such that…	So that…
Login	App User	be sure to access my Giro Account	By using Fingerprint for identification and TAN for authentication	I can be confident for my privacy
Scan QR Code	App User	scan my bills	typing in IBAN and reference information is no longer necessary	paying bills is with one click
Use Giro Account	App User	use my Giro Account	I can access banking services with my Smartphone	to pay bills
Create Transactions	App User	create transactions	it's simple	to pay bills
Edit Transactions	App User	view & edit transactions	I'm informed about what I'll pay	account status remains under control
Schedule Execution	App User	select the date of execution	I can plan for my account balance	account status remains under control
Account Status	App User	review account status	all pending transactions are considered	account status remains under control
Refill	App User	link to a savings account	I can refill my Giro Account	I'm able to pay my bills

In real life, there are a lot more functional users involved – from Compliance Officer to the bank's customer care department – providing additional user stories such that a real set contains rather a hundred stories instead of eight.

Counting the vision for the Android Mobile App with the eight user stories shown in TABLE 1 only yields 188 IFP; the COSMIC count is 105 CFP. We always start with a vision; thus, product size is approximately known. Sizing the vision allows identifying layers, data functions by IFPUG, respectively objects of interest when using COSMIC.

B. *The Architecture*

The example shown here is invented; however, it follows as closely as possible real experiences made in practicing size counts in agile development.

The architecture is standard. The smartphone never accesses banking data directly but always uses a middleware – here called *App & Web Server* – to connect to real banking data. Most of the architectural components already exist but need some enhancements for the new Mobile App.

For sizing, the pre-existing parts are tagged as "enhancements", compared with the "new development" needed for the additional features. Product size is the sum of both.

We have functional users for all five application systems; thus, they add to functional size. The functional users are represented by the broad arrows in Fig. 3. TAN, IAM, and CMS services are already existing and will not need any enhancements. The App & Web Server also is standard, but many functions require a server part, for accessing data or storing data that does not belong to the relatively unsafe smartphone. For instance, transactions and access keys are never stored on a smartphone.

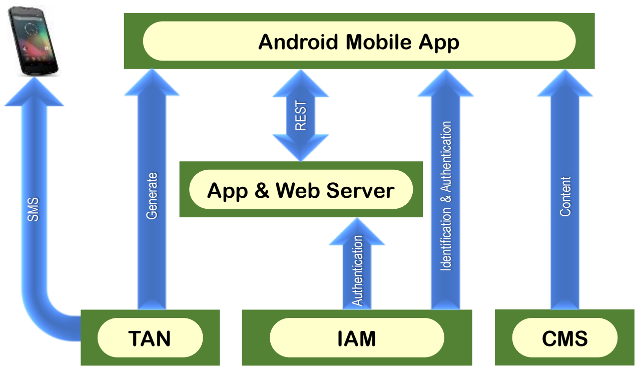


Fig. 3. Architecture Overview

C. *The Sprints*

Sprint 01 – Allegro. We assume, the development platform for Android was already installed, and thus we do not need an extra sprint to set it up.

Also, the team is experienced and used in cooperation. Accessing a camera on a smartphone is nothing new for nobody. Therefore, the team started without hesitation. The team selected the "View Giro Account Status" user story as its top priority. This user story had been badly implemented before and drawn much criticism. This priority decision allows business to see valuable results within two weeks' time.

In the first sprint, the *View Account Status* user story was selected from backlog – actually, from the vision. Story Point (StP) estimations are in parenthesis:

- As App User, I want to review my Giro Account status such that I can plan for my account balance to keep account status under control (8 StP).

The Product Owner discovered two new user stories, not part of the initial vision, now added to Backlog. The team considered them uttermost important:

- As App User, I want to include my credit cards in the Giro Account statement such that I see the amount necessary to pay my monthly statements, controlling Giro balance (8 StP).
- As App User, I want to connect with my credit card accounts such that I can use the same Giro for credit card payments, keeping account status under control (8 StP).

These two new user stories let the product size grow. Initially, the vision, or initial backlog, can be used to estimate the final product size, but the product size growths with each sprint finally expected when new user stories are discovered and added to the backlog. Therefore, it is safer to measure the real product size, as implemented after each sprint. In our case, the need for linking credit cards affects middleware as well; thus, the App & Web Server also starts growing with the new Android Mobile App product.

Also, the Login procedure was selected for its technical importance. This functionality was reused from a previous Mobile App and thus is not a new development:

- As App User, I want to be sure to access my Giro Account by using fingerprint for identification and TAN for authentication such that I can be confident for my privacy (3 StP).

Involving credit cards from foreign banks has become possible thanks to a new standard ISO 20022, that has been adopted throughout the EU, defining account access interfaces between different banks [15]. This new feature seems to increase customers' acceptance for the new Android Mobile App.

Sprint 02 – Andante. This sprint implements the main functionality, namely scanning a bill and creating a transaction that the Backoffice systems can execute:

- As App User, I want to scan bills received with my smartphone and pay without typing any IBAN or other reference information, with only one click (13 StP).
- As App User, I expect that scanning creates a transaction scheduled for next day to pay my bills (8 StP).
- As App User, I want to use my Giro Account, without a complicated process, to pay my bills (3 StP).

Note that scanning and creating a transaction is only one elementary transaction is IFPUG. We also want a link to Savings Accounts:

- As App User, I want to link my Giro Account to some other Savings Account such that I can refill my Giro for paying my bills with my smartphone (5 StP).

As before, the App & Web Server is also affected and needs some additional functionality. We link the Giro account to some savings accounts in case normal income, e.g., regular salary payments, are not enough to keep the balance in the positive.

Avoiding double counting, we attach a "Not Counted" label to Session Key and Giro Account that was already created in Sprint 01. For IFPUG, the EIF already initiated such as Identity Access Management and Mobile Content are also not counted, since not enhanced.

Sprint 03 – Scherzo. Managing and scheduling transactions means that we somewhat refine two user stories that already were part of the vision:

- As App User, I want to view, edit and delete transactions that I scanned before execution, such that I am informed about what I pay, and my account balance remains under control (8 StP).
- As App User, I want to see pending transactions such that I know what will happen to my account balance (5 StP).
- As App User, I want to reschedule transactions such that I can plan for my account balance and my Giro account is not overdrawn (3 StP).

Now we listen to the voice of the *Private Banking Counselor* and add yet two more user stories:

- As Counselor (or as Compliance Office), I want to be sure that all transactions are traceable such that any misuse or erroneous action can be reversed (5 StP).
- As Counselor, I need the possibility to view the Transaction Log (8 StP).

Thus, the log file becomes a *Transaction Log* and will get enhanced functionality that allows those two functional users to help their customers, respective meet legal requirements, e.g., when scanning transactions for tax fraud.

The transaction log is not something that the user needs, or wishes, but is useful when the bank needs blocking and reopening transactions. Since the App user needs such functionality, and since he or she also wants to see transaction history, this is a FUR that must be counted. The main reason for tracing transaction is compliance. However, we also listened to the voice of the Private Banking Counselor who wants to be able to support its customers effectively.

The need to view the transaction log is logically a part of middleware, the App & Web Server, although it might not exactly belong there, and most probably will be implemented somewhere else.

Sprint 04 – Marche Funèbre. The Security Officer found out that Android posts all images on some Cloud Service as soon as connected to some WLAN. However, this service can be switched off.

The ability to switching off needs additional functionality on the smartphone. It requires saving the user settings that are valid outside of the Mobile Payment App. Therefore, there is another new user story:

- As App User, I want to make sure that my camera scans a bill only for my Mobile App such that nobody can trace my payments, and things keep private (13 StP).

Instantaneous posting of pictures taken with a smartphone to Instagram, Twitter or some cloud service is a standard feature; however, it is not straightforward to consider it. It is not a bug; it is a new feature.

Obviously, a workaround could be to access the camera directly, exclusively. Using the preinstalled camera app increases portability but carries the risk that usually such code is not open. Hidden backdoors might persist even if settings are switched to “No share”. For security reasons, the camera settings are kept persistent.

The transaction, or data movement, already counted in Sprint 02 therefore gets changed and replaced by some newly developed software for scanning bills. This sprint provides work on the Android Mobile App only. Because access to internal smartphone camera settings is rather tricky – one must hack around internal privacy protection – total amount of added or enhanced functionality remains low.

Moreover, there is a “Refill Giro” user story implemented in Sprint 04 that complements the link to some savings accounts, and alert functionality:

- As App User, I want to refill my Giro account in case normal funding is late or insufficient, such that I can pay my bills that are scheduled for payment (8 StP).
- As App User, I want an alert in case of missing balance, such that I can pay my scheduled bills (5 StP).

Sprint 05 – Intermezzo. The code quality static tester tool in use by our team, is not very happy with the amount of *Technical Debt* that already has piled up. Source code needs refactoring. This gives rise to a new user story:

- As Developer, I want to make sure that my software is bug free, maintainable, and contains minimal technical debt (13 StP, non-functional).
- Another user story was added to the sprint backlog:
- As Business Owner, I want to make sure that our Mobile Payment App runs on all major smartphone brands and software versions in use (13 StP, non-functional; a suitable test service is commercially available).

This requires software tests:

  - Static Tests and fixing of bad code.
  - Dynamic compatibility tests with as many popular smartphone brands as possible.

The Intermezzo Sprint does not add or enhance any functionality neither to the Android Mobile App nor to any other applications. Both user stories are nonfunctional.

Sprint 06 – Menuetto. Overall progress is good, thus new ideas find fertile ground with the following new user stories:

- As App User, I like to see my spending history graphically, such that I can distinguish what I spent by payments and credit card, for managing liquidity (13 StP).
- As App User, I want to set the time slot for the graphical spending history, such that I see trends, and I can determine when to refill my Giro Account (3 StP).

A similar new user story also seems important enough for implementation:

- As App User, I like to see spending statistics, such that I can distinguish how much I spent for what, to plan my future spending (8 StP).

Sprint 06 – Menuetto is meant to add better appealing functionality to the Android Mobile App. Graphics are always better than just numbers; the Android library used allows to create graphs with relatively little effort. Thus, the team assigned high priority to these user stories.

This results in rather few new functionality, but some significant enhancements of already implemented functionality.

Sprint 07 – Finale. For the final sprint, we have only one additional user story added, that makes work done previously for the graphical representations partially obsolete, inducing enhancements to the already finished functionality providing graphic settings.

A little late maybe, our product owner might have talked to some user representative and found out that graphics are welcome, but they should be available on the web and – obviously – look the same.

Thus, we must move the graphical settings data function from the smartphone back to the server. This has the additional advantage that users retrieve their settings even after losing or exchanging their smartphone but does not impact the Mobile App. The new functional user story reads as:

- As App User, I like to share my spending history graphics with other platforms, by seeing the same graphics in Web Banking (8 StP).

The necessary finishing touches are represented again as non-functional user stories:

- As a developer, I want everything well documented (13 StP, non-functional).
- As product owner, I want automated tests before release (8 StP, non-functional).

This user story makes the previous solution (keeping graphical settings on the phones) obsolete.

IV. RETROSPECTIVE

In retrospective, process metrics are analyzed.

A. Product Size Growth

Product Size at Start of Sprint according IFPUG

	Android Mobile App				*App & Web Server*			
	New Dev	*Enhanced*	*Re-Dev*	*Total*	*New Dev*	*Enhanced*	*Re-Dev*	*Total*
Vision	151 IFP	37 IFP		188 IFP		129 IFP		129 IFP
Sprint 01		27 IFP		27 IFP		129 IFP		129 IFP
Sprint 02	48 IFP	50 IFP		98 IFP	23 IFP	129 IFP		152 IFP
Sprint 03	111 IFP	60 IFP		171 IFP	23 IFP	129 IFP		152 IFP
Sprint 04	153 IFP	65 IFP		218 IFP	36 IFP	129 IFP		165 IFP
Sprint 05	167 IFP	68 IFP	6 IFP	241 IFP	36 IFP	129 IFP		165 IFP
Sprint 06	167 IFP	68 IFP	6 IFP	241 IFP	36 IFP	129 IFP		165 IFP
Sprint 07	199 IFP	68 IFP	6 IFP	273 IFP	36 IFP	129 IFP		165 IFP
Final	**174 IFP**	**86 IFP**	**13 IFP**	**273 IFP**	**52 IFP**	**129 IFP**		**181 IFP**

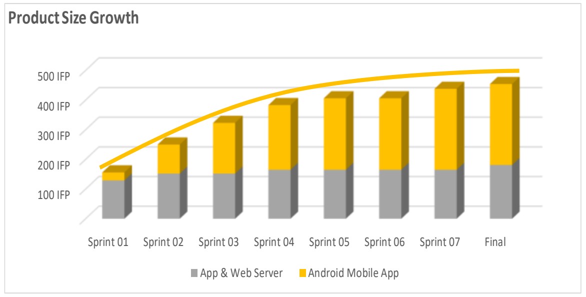
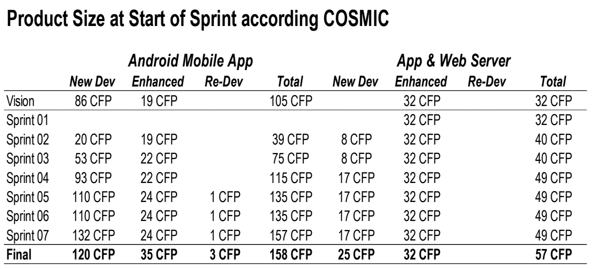


Fig. 4. Application Growth according IFPUG



Product Size at Start of Sprint according COSMIC

	Android Mobile App				*App & Web Server*			
	New Dev	*Enhanced*	*Re-Dev*	*Total*	*New Dev*	*Enhanced*	*Re-Dev*	*Total*
Vision	86 CFP	19 CFP		105 CFP		32 CFP		32 CFP
Sprint 01						32 CFP		32 CFP
Sprint 02	20 CFP	19 CFP		39 CFP	8 CFP	32 CFP		40 CFP
Sprint 03	53 CFP	22 CFP		75 CFP	8 CFP	32 CFP		40 CFP
Sprint 04	93 CFP	22 CFP		115 CFP	17 CFP	32 CFP		49 CFP
Sprint 05	110 CFP	24 CFP	1 CFP	135 CFP	17 CFP	32 CFP		49 CFP
Sprint 06	110 CFP	24 CFP	1 CFP	135 CFP	17 CFP	32 CFP		49 CFP
Sprint 07	132 CFP	24 CFP	1 CFP	157 CFP	17 CFP	32 CFP		49 CFP
Final	**120 CFP**	**35 CFP**	**3 CFP**	**158 CFP**	**25 CFP**	**32 CFP**		**57 CFP**

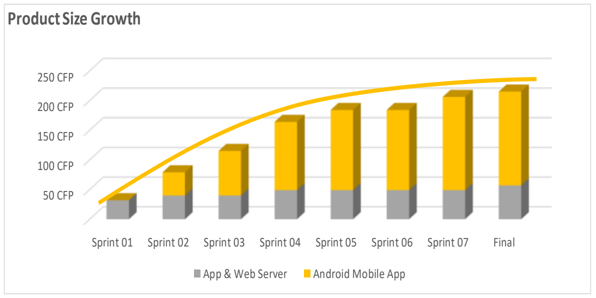


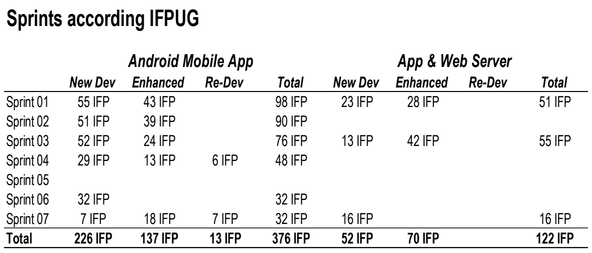
Fig. 5. Application Growth according COSMIC

In COSMIC (Fig. 5), we have less "Enhanced" functionality compared with IFPUG (Fig. 4), because a data movement might be newly developed even if connecting two already existing objects. Therefore, we found no "Enhanced" data movements within the Android Mobile App.

Note that when newly developed functions become enhanced, because of new requirements, they change the status from "New Development" to "Enhanced" for the product size. In fact, it means that these functions have been changed.

B. *Sprint Performance*

The IFPUG sprint sizes sum up to something higher than product size; this is an indication that the vision has been implemented in full. In the Android Mobile App, there is some deleted functionality in Sprint 04 and 07 that does not add to size but to sprint performance. The App & Web Server has no re-developed functionality.



Sprints according IFPUG

	Android Mobile App				*App & Web Server*			
	New Dev	*Enhanced*	*Re-Dev*	*Total*	*New Dev*	*Enhanced*	*Re-Dev*	*Total*
Sprint 01	55 IFP	43 IFP		98 IFP	23 IFP	28 IFP		51 IFP
Sprint 02	51 IFP	39 IFP		90 IFP				
Sprint 03	52 IFP	24 IFP		76 IFP	13 IFP	42 IFP		55 IFP
Sprint 04	29 IFP	13 IFP	6 IFP	48 IFP				
Sprint 05								
Sprint 06	32 IFP			32 IFP				
Sprint 07	7 IFP	18 IFP	7 IFP	32 IFP	16 IFP			16 IFP
Total	**226 IFP**	**137 IFP**	**13 IFP**	**376 IFP**	**52 IFP**	**70 IFP**		**122 IFP**

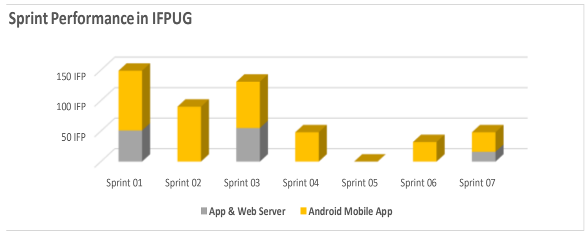
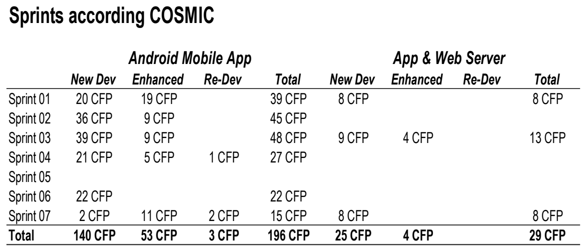


Fig. 6. Sprint Performance measured with IFPUG

Sprint 05 does not add any functionality, for both measurement methods.



Sprints according COSMIC

	Android Mobile App				*App & Web Server*			
	New Dev	*Enhanced*	*Re-Dev*	*Total*	*New Dev*	*Enhanced*	*Re-Dev*	*Total*
Sprint 01	20 CFP	19 CFP		39 CFP	8 CFP			8 CFP
Sprint 02	36 CFP	9 CFP		45 CFP				
Sprint 03	39 CFP	9 CFP		48 CFP	9 CFP	4 CFP		13 CFP
Sprint 04	21 CFP	5 CFP	1 CFP	27 CFP				
Sprint 05								
Sprint 06	22 CFP			22 CFP				
Sprint 07	2 CFP	11 CFP	2 CFP	15 CFP	8 CFP			8 CFP
Total	**140 CFP**	**53 CFP**	**3 CFP**	**196 CFP**	**25 CFP**	**4 CFP**		**29 CFP**

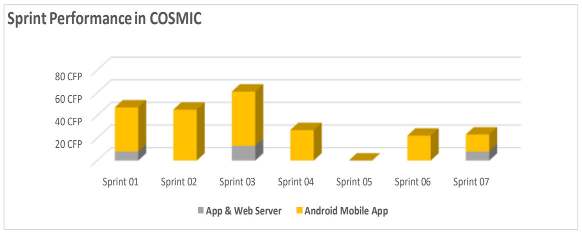
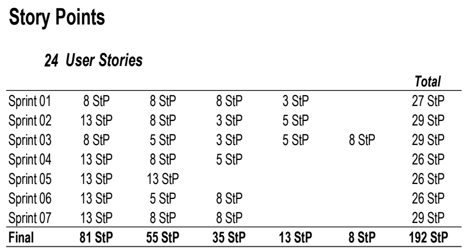


Fig. 7. Sprint Performance measured with COSMIC

Sprint 03 and 04 needed enhancements in existing functionality that was impacted by new, or changed, user requirements. Performance covers both new development and enhancements since both types of work require effort. The increment of product size, in contrary, as shown in Fig. 4 and Fig. 5, deteriorates even more from sprint to sprint.

The team estimated story points as shown in Fig. 8, according to its own habits and rules. The break-in in performance with Sprints 05 and 07 does not appear in Story Points; the team had a lot to do but did not add new functionality.



Story Points

24 User Stories

						Total
Sprint 01	8 StP	8 StP	8 StP	3 StP		27 StP
Sprint 02	13 StP	8 StP	3 StP	5 StP		29 StP
Sprint 03	8 StP	5 StP	3 StP	5 StP	8 StP	29 StP
Sprint 04	13 StP	8 StP	5 StP			26 StP
Sprint 05	13 StP	13 StP				26 StP
Sprint 06	13 StP	5 StP	8 StP			26 StP
Sprint 07	13 StP	8 StP	8 StP			29 StP
Final	**81 StP**	**55 StP**	**35 StP**	**13 StP**	**8 StP**	**192 StP**

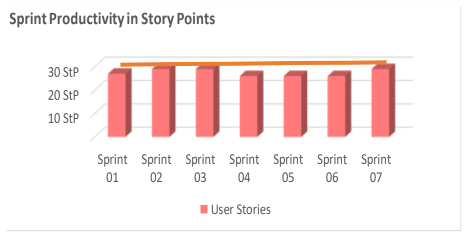


Fig. 8. Story Point Estimates by the Development Team (non-comparable with other teams)

The team estimated a total of 192 StP for the final 24 functional and non-functional user stories, grown from the original eight user stories for the initial vision (Table 1).

Productivity in IFPUG FP

Total Dev	*#Sprints*	*Length*	*Hours/Day*	*Team Size*	*PDR*
498 IFP	7	10 Days	7.2 h	6.8	7 h/IFP

Productivity in COSMIC FP

Total Dev	*#Sprints*	*Length*	*Hours/Day*	*Team Size*	*PDR*
225 CFP	7	10 Days	7.2 h	6.8	15 h/CFP

Fig. 9. Overall Productivity, in IFPUG and COSMIC

This yields a *Productivity Delivery Rate* (PDR) of 7 h/IFP, respectively 15 h/CFP. The same calculation can also be done for story points and yields a productivity number.

Productivity in StP

Total StP	*#Sprints*	*Length*	*Hours/Day*	*Team Size*	*Delivery Rate*
192 StP	7	10 Days	7.2 h	6.8	18 h/StP

Fig. 10. Overall Productivity, expressed in story points

Agile methodology uses velocity – the number of story points that the team can implement in one sprint – for predicting duration and cost of product development. The respective velocity in our Gedankenexperiment amounts to 18 h/StP. However, velocity does not allow to predict what functionality can be provided at the cost of those sprints.

C. *Cost of Agile Software Development*

As explained above (section A), cost estimates depend on the sprint productivity; however, it is difficult to predict how many new requirements will be detected during the agile sprints. Nor does the number of sprints planned predict the amount of functionality, and thus size, of the resulting product.

Nevertheless, such predictions are feasible using *Quality Function Deployment* (QFD) [16] by analyzing how well the vision meets the needs of the customer or user [17].

D. *Findings*

From Fig. 6 and Fig. 7 it becomes apparent that functional size growth diminishes when looking at later sprints. Intuitively this is clear, because tests, refactoring and documentation become more and more dominant later in the product life cycle. The relation between effort and product functional size increment deteriorates over time. The amount of this deterioration is paramount for predicting cost of product development.

Sprint performance in terms of functional size deteriorates from the start value to about one third. Product size increment follows a logarithmic curve whose parameters should be highly interesting to cost estimators. Surprisingly, not much is published in the scientific literature about this curve. It looks like this curve describes some functional growth, without an apparent limit – it is not a saturation curve – even while the effort, expressed in story points, remains stable and constant (Fig. 8). The implemented FUR seem to generate additional FUR like fractals.

This "fractal growth curve" resulting from the shift from FUR to NFR has been observed in all agile development undertakings that the authors have monitored in the last five years. The reasons could be: Teams shift from the focus on developing new functionality to testing, refactoring, and preparing deployment. New, more detailed FUR are uncovered in this process. This seems characteristic for the product under development, and is supported by DevOps. The form of this "fractal growth curve" depend on the amount of shift-lest, and on the details of the DevOps approach. For instance, with *Autonomous Real-time Testing* (ART) [18], testing becomes an ongoing activity. Then, even while effort, and thus cost, is evenly distributed over the full product life cycle, functional growth is not.

Smoothing the curve, the mathematical representation of this curve suggests a logarithm; that would suit to "fractal". However, "fractal" suggests relatively simple growth rules deserving high interest. This needs further investigation. It should become the target when benchmarking performance of software product development methods, and teams.

Our thriving experiment opens more questions than it answers. Nevertheless, for a few research questions we have answers, thanks to our Gedankenexperiment:

- Management should monitor the characteristics of the fractal growth curve.
- We know what to measure when, and how. We need both functional size and story points.
- We prefer the subjective team measure captured by story points over effort measurements by counting hours. Story points better reflect the difficulties encountered and mastered by the team.
- Measuring agile development must address each sprint, not just the final product, an initial vision (or backlog), or the MVP. The reward for these additional measurements is apparent.
- Comparing the size of the product with the effort spent in sprints indicates how much work was spent in getting the requirements right, implementing NFR, refactoring, removing technical debt and other quality improvements.
- While the product can be compared with the vision in terms of size, the implemented features might differ quite a bit. Also, the MVP does not remain stable and undergoes change.
- The question whether IFPUG or COSMIC shall be used for measuring agile depends on the product domain; both methods work for managing development despite the lack of VIM/GUM compliance of IFPUG.

It should be restated that this Gedankenexperiment reflects the practical experiences made when monitoring agile software development, in various industries, over five years now. Also, the tools we use for counting are adapted to agile development, avoiding double counting for sprints and the product. Thus, the effort for measuring sprints is equal to the effort for measuring the final product; only the work is in sprints rather than monolithic.

V. Conclusion

Using story points has the disadvantage that the team must already be available and ready to assign story points to user stories. In contrary, using functional size measurements enables product managers to gauge their vision also in view of product improvement plans and schedules. This works better because the PDRs of Fig. 9 already include the characteristics of agile development. These values can now be used to predict future performance of the same team.

The values presented here with this freely invented simple app product are near to what had been observed in practice in development of mobile applications.

Comparing story points with size metrics is useful on product level but not on the sprint level. The aim of the sprints – expressed by classical musical terms – plays a major role. For predictions, it is safe to assume that the vision covers about half of the product that will be implemented. Often, the initial vision backlog contains user stories or even epics that will become obsolete during development of the product.

COSMIC allows to precisely gauge size in sprints and better sizes NFR that address networking and performance [13]. For technical software, developers find data movement maps useful, see [8]. IFPUG allows for less precision due to the lack of compliance with the VIM and the GUM; however, for transaction-oriented applications such as Web and Mobile development, it is good enough and eases communication with less technical people. From the viewpoint of sizing agile, both methods are equally useful.

The difference between size of the product and total size of all sprints, plus the amount of enhancement works, reflects the effort needed to find the correct requirements by the agile team. A smaller product sometimes better reflects the true needs of its users, and smaller products fit better in DevOps life cycle. Thus, the enhancement effort is not lost.

Functional sizing allows to better understand the percentage of effort that is needed for NFR, refactoring and testing and may vary strongly per sprint. Typically, nonfunctional efforts count for more than half of the total effort; thus, the value of functional sizing for sprint planning is limited. However, for predicting the number of sprints needed to reach a MVP [19], for monitoring progress, and for managing DevOps, functional sizing is without alternative.

Moreover, COSMIC can be used for managing agile development by the Buglione-Trudel Matrix, see [8] and [20]. Finally, COSMIC is the method of choice for sizing tests, especially for Autonomous Real-time Testing (ART) [18]. Combining ART with Agile and DevOps yields particular benefit for large software-intense systems, such as autonomous vehicles, and intelligent things.

Acknowledgment

Thanks to the customers of Euro Project Office and QSMA Switzerland who allowed us to monitor sprint progress in close cooperation with the agile teams and inspired us for this Mobile App Gedankenexperiment.

We are also grateful to the reviewers who provided us with valuable hints how to improve this paper.

References

[1] ISO/IEC Guide 99:2007, "International vocabulary of metrology – Basic and general concepts and associated terms (VIM)," TC/SC: ISO/TMBG, Geneva, Switzerland, 2007.

[2] ISO/IEC CD Guide 98-3, "Evaluation of measurement data - Part 3: Guide to uncertainty in measurement (GUM)," TC/SC: ISO/TMBG, Geneva, Switzerland, 2015.

[3] COSMIC Measurement Practices Committee, "COSMIC Measurement Manual for ISO 19761 – Version 5.0 – Part 1-3," COSMIC Measurement Practices Committee, Montréal, 2020.

[4] IFPUG Counting Practice Committee, "Function Point Counting Practices Manual - Version 4.3.1," International Function Point User Group (IFPUG), Princeton Junction, NJ, 2010.

[5] ISO/IEC 19761, "Software engineering - COSMIC: a functional size measurement method," ISO/IEC JTC 1/SC 7, Geneva, Switzerland, 2011.

[6] ISO/IEC 20926, "Software and systems engineering - Software measurement - IFPUG functional size measurement method," ISO/IEC JTC 1/SC 7, Geneva, Switzerland, 2017.

[7] P. Hill, Ed., Practical Software Project Estimation 3rd Edition, New York, NY: McGraw-Hill, 2010.

[8] T. M. Fehlmann, Managing Complexity - Uncover the Mysteries with Six Sigma Transfer Functions, Berlin, Germany: Logos Press, 2016.

[9] T. M. Fehlmann, "When use COSMIC FFP? When use IFPUG FPA? A Six Sigma View," in *COSMIC Function Points - Theory and Advanced Practices*, R. Dumke and A. Abran, Eds., Boca Raton, FL, CRC Press, 2011-3, pp. 260-274.

[10] M. Rehkopf, L. Daly, C. Drumond, D. Radigan, S. Mansour and M. Suntinger, "Atlassian Agile Coach," Atlassian Corporation Plc, Sydney, NSW, Australia, 2020.

[11] F. Erich, C. Amrit and M. Daneva, "A Qualitative Study of DevOps Usage in Practice," *Journal of Software: Evolution and Process,* vol. 29, no. 6, June 2017.

[12] M. Cohn, Agile estimating and planning, New Jersey, NJ: Prentice Hall, 2005.

[13] COSMIC Consortium, "Guideline on Non-Functional & Project Requirements V1.03," November 2015. [Online]. Available: http://cosmic-sizing.org/publications/guideline-on-non-functional-project-requirements/. [Accessed 18 November 2015].

[14] ISO/IEC 14143-1, "Information technology - Software measurement - Functional size measurement - Part 1: Definition of concepts," ISO/IEC JTC 1/SC 7, Geneva, Switzerland, 2007.

[15] The SWIFT Standards Team, ISO 20022 for Dummies, John Wiley & Sons, 2020.

[16] ISO 16355-1, "Applications of Statistical and Related Methods to New Technology and Product Development Process - Part 1: General Principles and Perspectives of Quality Function Deployment (QFD), Geneva, Switzerland: ISO TC 69/SC 8/WG 2 N 14," ISO TC 69/SC 8/WG 2 N 14, Geneva, Switzerland, 2015.

[17] T. M. Fehlmann and E. Kranich, "Early Software Project Estimation the Six Sigma Way," *Lecture Notes in Business Information Processing,* vol. 199, pp. 193-208, 2014-2.

[18] T. M. Fehlmann, Autonomous Real-time Testing - Testing Artificial Intelligence and Other Complex Systems, Berlin, Germany: Logos Press, 2020.

[19] E. Ries, The Lean Startup, New York, NY: Crown Publishing Group, 2011.

[20] T. M. Fehlmann and E. Kranich, "Managing Software Projects by the Buglione-Trudel Matrix," in *11th European Conference on Information Systems Management - ECISM 2017,* Genova, Italy, 2017.