Dominik Bork, Dimitris Karagiannis, Heinrich C.Mayr (Hrsg.): Modellierung 2020
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2020 109

Model-based Software Cost Estimation

Calculating Time and Effort for Software Evolution Projects

Harry Sneed[1], Wolfgang Prentner[2]

Abstract: Estimating the costs of an evolution project differs from development project estimation and must follow its own rules. When estimating development project costs the whole system is taken into consideration. When estimating evolution costs only those parts of the system are considered that have to be changed or added. The rest is left as it is, but must be included in the test. The mixing of changed components with new components and old components presents several challenges to software product management. The main challenge is how to recognize those features that have to be added or changed – feature analysis. Together they make up the change domain. The extent of this change domain is the key factor in estimating the costs of change in each new release. It is measured by means of one or more size metrics such as function-points, data-points and object-points in order to convert size into effort. The approach used here was to model the change requirements and then compare the change model with the original requirements model to ascertain the scope of the change. To this end, both the original and the current requirements had to be extracted from the requirement text and then modelled. This approach was applied here to calculate the costs of expanding a national health record system. The preliminary results are presented in this short paper.

Keywords: Software evolution, Requirement Modelling, Natural Language Processing, Software sizing, Requirement Metrics, Function-Points, Data-Points, Object-Points, Model-based Estimation.

1 Introduction

The IT system to be evolved here is the security subsystem of an information system for managing health records of state insured citizens. There are more than 5 million health records in this particular system. The system has already been in operation for more than 7 years. It was developed by a local software shop specialized in health information systems and the same shop is still maintaining it. Recently additional security requirements have come up which were not considered in the original requirement analysis, mostly as a result of new data protection legislation. Many of them deal with access rights, user protection and system security levels. They were not foreseeable at the time the system was first conceived and if they were, it was decided to postpone their implementation. Now the law requires for them to be adhered to. The

[1] SoRing Kft, Kapitanutca 6, H1123 Budapest, Hungary, Harry.Sneed@SoRing.hu

[2] ZT-Prentner-IT, Kagranerplatz 40, A1221, Wien, Austria, Prentner@ZTP.at

national health agencies have no choice but to build them into their current health administration systems. The alternative would be to build a completely new system which satisfies all the old requirements plus the new ones, i.e. a complete redevelopment. In such a redevelopment the entire requirement model must be made again from scratch. In an evolution project only the new requirements need to be re-modelled. The old requirements are assumed to be already implemented. If the new requirements are to be compared with the old ones, then these too must be modelled as they were here. The reverse engineering of the old requirement document took more than four times the effort required to model the new requirements.

2 Levels of System Modelling

In any evolving IT system there are at least two levels to be modelled. These are the requirements and the test. If the evolution team is working according to the book there will be a design model and a test model. Model-based testing presupposes a test model from which system test cases can be derived. Model-based cost estimation presupposes a requirement model from which system size metrics can be derived [Selb09].

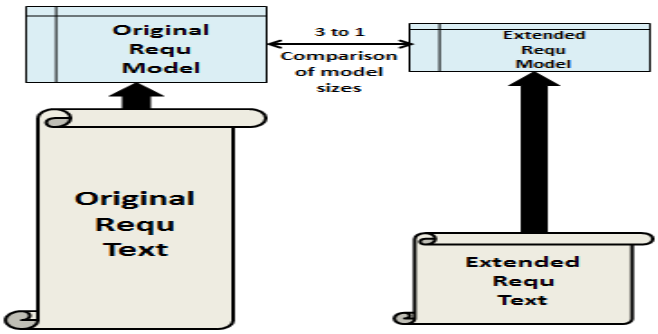


Figure 1: Modelling the Requirements

In this evolution project the requirement documentation consisted of two separate documents in German language:

- A detailed description of the existing security system requirements (Pflichtenheft)
- A general description of the new security system requirements (Lastenheft).

2.1 Existing System Requirements

The specification of the existing system requirements is a 390 page word document containing a combination of texts, tables and diagrams. The texts are for the most part

short paragraphs prescribing some security feature of the current system such as the authentication of users who log into the system. The nouns in those paragraphs are often acronyms defined in an acronym table at the end of the document. Otherwise they are a mixture of German and English medical terms such as “treatment”, “prescription”, “medication”, “Arzt”, “Behandlung” and “Ordination”. Often English and German terms are used alternately to denote the same object types, for instance “treatment” and “Behandlung”. To comprehend the texts the reader needs to be familiar with both German and English medical terms as well as with the acronyms used in this context. A sample from the text might look like this:

„Dem Gesetz entsprechend ist ein Aggregiertes Audit Record Repository (A-ARR) als zentrales Service zu errichten, das es den Teilnehmern (Bürgern) ermöglicht, Einsicht in die aufgezeichneten Protokolldaten, die ihre eigenen Gesundheitsdaten betreffen, zu ermöglichen. Die Protokolldaten werden von der ZGF bzw. von ETS und PAP erstellt und an das A-ARR mittels https weitergeleitet. Dem Bürgerportal steht eine lesende SOAP Schnittstelle zum Abrufen aller Protokolleinträge eines Bürgers zur Verfügung.“

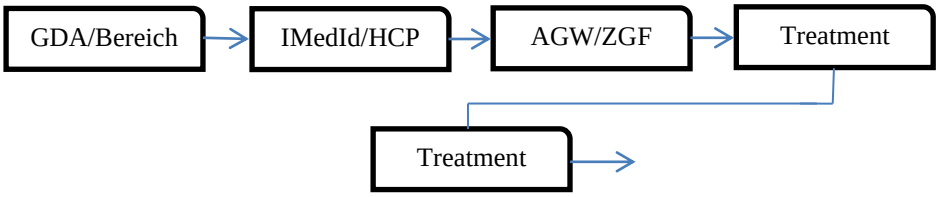
Sample 1: Specified Requirement

The texts are enhanced by tables and diagrams. If the text is describing an object there is often a table defining the attributes of that object, for instance for the term “Authentication” there is a list of authentication types equivalent to an enumeration.

# Authentication	R	
# @NotBefore	R	Time instant from which the assertion is useable. It is set as the issue instant
# @NotOnOrAfter	R	Time instant at which the assertion expires. Value is set to 4 hours
# @AudienceRestriction	R	This element contains the list of Audiences, e.g., the *contexts* (services) for whom the STS issued the assertion. Identity Assertion is used only with ETS (https://elga-online.at/ETS).
# AuthnStatement	R	
# @AuthnInstant	R	Time instant of authentication in UTC Format:yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fff'Z'
# AuthnContext	R	
# AuthnContextClassRef	R	urn:oasis:names:tc:SAML:2.0:ac:classes.*
# AttributeStatement	R	HCP identity attributes and permissions (Attribute der Identity Assertion)
# ds:Signature	R	Enveloped XML signature of the issuer of the Identity Assertion

Sample 2: Specified Data Attributes

If the text is describing a process then a process diagram is included to depict the sequence of steps within that process. The description of a process often turns out to be recursive as the steps of a process may be themselves sub-processes for which further diagrams are needed.



Sample 3: Specified Process

To measure the size and complexity of a requirement model it is necessary to distinguish between objects and actions. An object is a data entity or data set. In the code entities are implemented as structures or classes. An action can be a process, a use case or an elementary function. A process is actually a sequence of procedurally related functions. In the code processes are implemented as procedures, but in object-oriented systems, procedures are not explicitly defined, thus they are not statically recognizable. This leads to a semantic gap between requirement specifications and code. Processes or sequences of elementary functions can only be recognized by dynamic analysis [GDG06].

The detailed requirement model is detailed because what it is describing already exists and can be observed in operation. The authors of the detailed requirements are describing what happens when the system is executed. They are describing what they can physically observe. The model may not be totally accurate but it is accurate enough to be used as a basis of measurement. The only truly accurate description of a system is the code itself. By comparing the requirements with the code one can detect where they deviate from one another, but only if they are at the same level of abstraction [ChGa01]. In comparing one abstract model with another we are actually comparing two shadows of the real system.

2.2 Projected System Requirements

The specification of the new system requirements for the security system is not a single all-inclusive document like that for the current system as a whole, but a set of seven related documents prescribing what additional features the new security subsystem should have:

- An 11 page description of the proposed collaboration platform
- A 5 page description of the build process
- A 7 page description of the documentation classes
- A 10 page description of the virtual system architecture
- A 15 page description of the application container
- A 36 page description of the injection passport pilot application
- A 62 page description of the proposed x-Ray exchange service.

That adds up to 146 pages of requirement specifications for the enhanced security

system. Of course these requirement specifications are less detailed than the current system specification because the authors are not describing what they see but prescribing what they would like to have. They cannot physically observe those features but only imagine how they might be implemented. The objects and actions are therefore less precisely defined or their exact definitions are left open to be defined at implementation time (tbds) [Mugr08].

3 Modelling the Security System Requirements

The primary goal of this particular modelling process was to measure the size and complexity of the proposed system change. Since there was neither a design nor code for the extended system, the only thing that could be measured was the requirements of the new security subsystem. These could be compared with the requirements for the current system as a whole. System cost estimation implies measurement. To make an objective comparison one must compare numbers and that requires measuring the objects to be compared – in this case the two requirement documents. In order to measure a requirement text document the text of that document must first be converted to a given requirement model that coincides with the requirement-based estimation methods. Then the two models can then be matched with one another to identify the differences between them. Thus, the first step in comparing the new requirements with the existing ones is to extract a numeric model from the requirement text.

Once a numeric model has been created from the descriptive text, the model elements can be counted and the counts compared. The prerequisite to comparing models is to identify the model elements described in the text. At the current state of artificial intelligence this task can only be performed by an intelligent human being. The person assigned to this work must examine each and every text passage to decide what that text is all about and what model element is being specified by it.

Based on his many years of experience in analyzing requirement documents in different natural languages, combined with the knowledge acquired from the literature on requirement modelling, the first author identified some 30 model types ranging from use-cases to elementary function steps and from logical data entities to elementary data items, including services and user-interfaces. A full list of the model element types was published in a paper by the authors on requirements reengineering [SnPr18].

Based on his or her knowledge of the application and his or her familiarity with the terms used in the target document, the analyst assigns the text passages to a selected model type. It is human analyst who decides what a text passage is prescribing. Actually the analyst is associating the terms used in the text with the terms associated with the model types. The experienced analyst will soon recognize that the same text patterns are used again and again to describe model elements of the same type. With these patterns in the back of mind, the analyst will be able to rapidly scan through the requirement text and assign the text paragraphs to the appropriate model element types [Lefi11].

Having recognized a model element, the analyst inserts the element type identifier in a separate line before the text passage or paragraph to which that type applies.

&FREQ-011 Patientenidentifikationen

*Alle Patientenidentifikationen werden im Format **CX HL7 V2.5** gemäß IHE XDS.b Profil erwartet.*

Sample 4: Requirement Definition

&Regel Die Signatur der Assertion wird gemäß W3C XMLDSig geprüft.

Sample 5: Rule Definition

This action is referred to as marking up the text. It is similar to marking up an XML document, but much less formal. The end delimiters are left out. A model element ends where the next one begins. Considering that a page of requirement documentation will contain on average 6 model elements, it should be possible to mark up a page of requirement text within 20 minutes, or at the rate of 18 model elements per hour. A document of 100 pages would require 2000 minutes = 33 hours or some 4 working days. This is exactly what it cost to mark up the new security requirement document in this cost estimation project. The marking up of the original security document took more than 16 working days. This was not only due to the volume but also to the density of the descriptions [Bria17].

&UseCase: HCP_Assertion_anfordern

Läuft eine Benutzersession ab oder wurde invalidiert, muss auch die HCP Assertion beim ETS invalidiert werden.

&MainPath: Steps =

1) Ein GDA System oder ein Gateway eines System-Bereichs sendet eine WS Trust RST Issue Transaktion an die ZGF des System-Bereichs, um sich bei dem System wieder anzumelden.
2) Im Security Header der SOAP Nachricht befindet sich die Identity Assertion des lokalen IDPs.
3) Der Apache der lokalen AGW leitet die RST Nachricht zum zentralen ETS weiter.
4) Eventuelle Fehler werden in einer WS Trust Tabelle zurückgeliefert.
5) Fehlerfälle werden gemäß der Fehlerbehandlungsrichtlinie behandelt.

Sample 6: Use Case Definition

Once a requirement document has been marked up it can be automatically measured. The text parser will not only be able to recognize and count the nouns but also to count the model element types, i.e. it can count individual functional and non-functional requirements, use cases, actors, inputs, outputs, interfaces, data objects and data attributes. The tool for analyzing the marked-up text and counting the model entities has

been described in previous papers [Sned05]. From these counts it can derive data-points, function-points, object-points and usecase-points. These are the essential size metrics for determining the extent of a requirement model. The model sizes will not necessarily coincide with the code sizes as the model is only a shadow of the actual system. However by measuring the size of a shadow one can make some assumptions about the size of the real object depending upon the time of day. In the case of a requirement model, it depends upon the state of the requirements at the time when the model is measured. Unfortunately we cannot assume that the requirement document will be consistent with the live system. Some requirements may have never been implemented or were only partially implemented. Unfulfilled requirements make up for a good part of the technical debt [Ster10].

This does not hold for the model of the projected requirements, i.e. those that are yet to be fulfilled. They reflect how big the system extension would be if all of the new requirements were fulfilled. One can get an impression of the relation between the size of the original model and the new enhanced model by comparing their size metrics with one another [Sned10]. In table 1 the number of entities in the current requirement model are listed out next to the number of entities which are to be added in the extended requirement model. The new requirement documentation only includes the additional entities, i.e. the delta of the extended model. Of course the existing model will also be affected by the enhancements made, so additional costs have to be planned for adapting and testing existing artifacts impacted by the system enhancement.

Require. Entity Type	Current Entity Count	Added Entity Count	Total Count
User interfaces	80	8	88
System interfaces	154	30	184
Data Objects	133	61	194
Data Attributes	1929	193	2022
Object States	406	192	598
Actions	1139	217	1356
Business Rules	892	197	1089
Business Processes	98	19	117
Actors	12	6	18
Use Cases	101	20	121
Use Case Paths	40	9	49
Use Case Steps	484	56	540
Logical Test Cases	101	20	121
Function-Points	3868	1134	5002
Data-Points	15347	4971	20318

Tab. 1: Requirement Model Entity Counts

4 Estimating System Enhancement Costs

By comparing the size metrics of the existing requirement model with those of the projected model it was possible to determine the degree of change. The size of a system can be viewed from two points of view:

- Functional point of view
- Data point of view [CMPB05].

The functional point of view is reflected in the number of function-points and use case-points. The data point of view is expressed in data-points and object-points.

4.1 Counting Function-Points in the new Requirement Model

Function-Points are actually weighted counts of the model inputs and outputs plus the weights of the data entities and internal interfaces defined in the model. When the requirement model is extracted from the requirement text all the user interfaces, incoming messages and service requests are marked as being inputs and all the user interfaces, reports, outgoing messages and service responses are marked as being outputs. Inputs weigh from 3 to 6 points, outputs weigh from 4 to 7 points, data entities weigh from 5 to 15 points and internal interfaces weigh from 5 to 10 points, depending on the complexity of the requirement model as a whole. These weighted counts are added up to give the number of requirement function-points [IFPG99]. Of course the counting rules have to be simplified in order to automate the counting. The complexity of individual inputs and outputs cannot be measured. Instead the complexity of the model as a whole is applied to every input and output. If the complexity of the model is greater than 0.6 all of the outputs of that model will be assigned 7 points and all of the inputs 6 points. This simplification is already made in the Cosmic Function-Point method where all inputs and outputs have the same weight of 1.

In the model of the projected security subsystem 1134 function-points were counted, indicating a difference of 3.4 to 1 between the size of the original requirement model for the system as a whole and the size of the new model of the security subsystem. Carrying this difference over to the effort involved, it means that the extended security features should cost no more than 30% of what the original system had cost.

4.2 Counting UseCase-Points in the new Requirement Model

Use Case-Points are weighted counts of the use cases and actors defined in the model. The weight of a use case is determined by the number of paths and steps specified for that use case. Every use case must have at least one path and each path must have a least one step. There is no limit to the number of paths and steps a use case may have, but normally there is no more than two paths – the normal path and the exceptional path – each with some 3 to 10 steps. The authors of the use-case method assign 5 points to the

cases with up to 3 steps, 10 points to use cases with 4 to 7 steps, and 15 points to use cases with over 7 steps. In addition system actors are weighted from 1 to 3 points depending on their type of interface, whether batch, message or dialogue. The weights of the use cases and actors are summed up to give the total number of use case-points [Karn93].

In the model of the existing system as a whole there were 2035 use case points. In the model of the projected security system 220 use-case-points were counted. That indicates a ratio of 9:1 from the existing entire requirement model to the proposed partial one. That indicates that the system extension would only cost 10% of what the original system cost. This cannot be true. It results from the fact that the use-case method was intended for estimating frontend development. The planned security subsystem is mainly a backend solution. Thus, the use-case method does not apply here.

4.3 Counting Data-Points and Object-Points in the Requirement Model

The data point of view is reflected in the number of data-points and object-points. At the requirement model level these two counting procedures are very similar. Data elements, i.e. nouns, are weighted by 1, data objects by 2, user interfaces, reports and messages by 4 to give the number of data-points [Sned05b]. To compute object-points the number of data groups and messages are weighted by 4, the number of use cases by 8, and the number of actions by 3 [Sneed96]. In the model extracted from the original requirement document there were 15.347 data-points counted whereas in the model extracted from the requirement specification of the new security subsystem 4971 were counted. With object-points, it was 8669 object-points in the original model for the system as a whole as opposed to 2043 in the new partial model for the security subsystem. This gives a relation of 3.0 to 1 for data-points and 4.2 to 1 for object-points.

Model-Type	Funct-Points	Data-Points	UC-Points	Object-Points
Original	3868	15347	2035	8669
Extension	1134	4971	220	2043
Ratio	3.4:1	3.0:1	9.2:1	4.2:1

Tab. 2: Size of System Change Model relative to Original System Model

It is notable that function-points and data-points come to a similar ratio although the two methods are counting quite different model types. Function-points are counting data flows whereas data-points are counting data-elements. Use case points are counting paths and steps whereas object-points are counting data-groups. Object-points are more difficult to identify at the requirement level as data groups are not always explicitly defined. Thus the ratio of 4.2 to 1 deviates somewhat from the ratios of 3.0 and 3.4 to 1. Object-Points are more easily countable at the design level where the data structures are closer to the implementation in a programming language with well-defined model types such as packages, modules, classes, interfaces and methods. The use-case point count deviates significantly from the other counts. This is because there are only 20 use cases which can be assigned specifically to the security subsystem. The other use cases which

pass thru the security subsystem actually belong to other subsystems.

Unfortunately a requirement model is by definition fuzzy. The size counts cannot be more precise than the requirement text. If the text is inexact, inconsistent and incomplete the model extracted from that text will also be inexact, inconsistent and incomplete. The point to be made here is that even fuzzy, informal and imprecise requirement specifications can be captured in a requirement model and that this model can be used to measure the approximate size of the software to be developed. The question is whether it is better than nothing at all. Here an attempt was made to capture the size of the planned change relative to the system as a whole using fuzzy size metrics [Cohn06].

5 Estimating absolute Costs of System Enhancement

An alternative approach to estimating the costs of enhancing the security system is to convert the requirement size metrics into effort based on a benchmark productivity table. Such a benchmark table was provided by the David Consulting Company for users in the USA [BuDe08]. Here one can see that productivity varies between 9 und 25 Function-Points per person month. This range coincides with the productivity of the GEOS Project in Vienna where the developers produced an average of 1,1 Function-Points per work-day [SnHa05].

Projektart	Produktivität
Client/Server Entwicklung	17 Function-Points pro PM
Mainframe Weiterentwicklung	13 Function-Points pro PM
Websystementwicklung	25 Function-Points pro PM
E-Business Systementwicklung	15 Function-Points pro PM
Standard-Softwareentwicklung	18 Function-Points pro PM
Data Warehouse Entwicklung	9 Function-Points pro PM

Tab. 3: Function-Point Productivity in the USA

Every software development organization should have its own productivity table of past projects giving the relation of requirement size metrics to the effort required to implement them. From this table the organization can project the effort required for future projects by matching the sizes. If it took n person months to implement m function-points in the past, it can be assumed that it will take a similar effort to implement m function-points in the future. Deviations can be traced to different influence factors which are included in the cost calculation. This is the standard approach to converting function-points into effort as propagated in the literature since

1983. The tool used for converting function-points into effort was SoftCalc, a tool originally developed by the first author in 1990 [DuFo96].

SoftCalc does not assume a linear relation between points and effort. On the contrary, there is a separate productivity table for each estimation method in which the relation of points to person-months is recorded. It is via this table that the point count is related to the effort as shown in the following table.

Project Type	Project Technology	Project Subject	Nr Units	Person Months	Units perPM	Total Costs (in €)
Redevelop	4GL-Synon	Logistic	32.383	1200	30	16.000.000
Reimple	Object	Transport	12.067	694	17	11.104.000
Reimple	Object	Public	9.298	270	34	4.330.498
Redevelop	Procedural	Payroll	17.898	520	34	6.240.000
Convert	Cob2Java	Insurance	31122	513	60	615.600

Tab. 4: Function-Point Productivity in Austria

For the security system extension 1134 function-points were counted in the new requirement model. Based on the productivity from the original system, this amounted to 72 person months. Adjusting this raw estimate by the

- Influence, resource, risk and overhead factors

lead to a final effort estimation of 88,6 person months for the proposed system enhancement. The influence factors used here are those proposed by the IFPUG convention plus those added later on by the OMG.

- data communication (original IFPUG)
- system type (has been added)
- system performance (original IFPUG)
- multi mandate (has been added)
- transaction rate (original IFPUG)
- product reliability (has been added)
- product usability (original IFPUG)
- business criticality (has been added)
- process complexity (original IFPUG)
- system reusability (original IFPUG)
- migration necessity (original IFPUG)
- system security (has been added)
- installation multiplicity (original IFPUG)
- system adaptability (original IFPUG)

System quantities:		Effort estimates	
UseCases	22	Undjusted Effort:	72,0 PMs
Objects:	51	☑ Influence Factor	1,07
Interfaces:		Influence adjusted Effort:	77,0 PMs
Components:		☑ Resource Factor	1,00
Test Cases:		Resource adjusted Effort:	77,0 PMs
Change_Rate	0,20	☑ Risk Factor	1,00
		Risk adjusted Effort:	77,0 PMs
		☑ Use Overhead Factor	0,15
		Final Effort:	88,6 PMs

Size measurement		Project estimates	
Unadjusted Size:	1134,00	Minimum Effort:	77,0 PMs
Quality Factor:	1,00	Minimum Time:	15,6 Months
Quality adjusted Size:	1134,00	Minimum Cost:	1.155.472
Complexity Factor:	0,67	Actual Cost:	1.328.792
Complexity adjusted Size:	757,51	Optimal Staff:	4,9 Prs.
Final adjusted Size:	909,01	Annual Maint. Effort:	18,5 PMs

Sample 7: Absolute Function-Point Estimation

The absolute estimation by data-point converted the 4971 data-points of the extended requirement model to 56 person months which were then adjusted to 61,4 PMs by the data-point influence-factors and the project overhead. The 10 data–point influences were defined by Sneed in the original data-point paper.

System quantities:		Effort estimates	
UseCases	22	Undjusted Effort:	56,2 PMs
Objects:	51	☑ Influence Factor	0,95
Interfaces:		Influence adjusted Effort:	53,4 PMs
Components:		☑ Resource Factor	1,00
Test Cases:		Resource adjusted Effort:	53,4 PMs
Change_Rate	0,20	☑ Risk Factor	1,00
		Risk adjusted Effort:	53,4 PMs
		☑ Use Overhead Factor	0,15
		Final Effort:	61,4 PMs

Size measurement		Project estimates	
Unadjusted Size:	4971,00	Minimum Effort:	53,4 PMs
Quality Factor:	1,00	Minimum Time:	13,6 Months
Quality adjusted Size:	4971,00	Minimum Cost:	801.440
Complexity Factor:	0,67	Actual Cost:	921.656
Complexity adjusted Size	3320,63	Optimal Staff:	3,9 Prs.
Final adjusted Size:	3984,75	Annual Maint. Effort:	12,8 PMs

Sample 8: Absolute Data-Point Estimation

6 Estimating relative Costs of System Enhancement

The final step in estimating the costs of the security system enhancement was to compare the absolute effort estimation with the relative effort estimation and to reconcile the two estimations with one another. The absolute effort estimation obtained by joining the actual function-point count of the new requirement model with the current function-point productivity table was 88.6 person months. The relative effort estimation obtained by comparing the size of the extended function model with the size of the original function model was 85 person months, almost exactly the same as with the absolute effort estimation of 86.5 person months. The absolute effort estimation with the current data-point productivity table was 61.4 person months. The relative effort obtained by comparing the extended data model with the original data model was 94 person months, a difference of 50% to the absolute cost estimation. This indicates that for this particular requirement document the function-point estimation is more reliable.

7 Justifying the Costs of model-based Estimation

It cannot be denied that this estimation could have been done with less cost. Constructing a requirement model just to estimate enhancement costs is indeed somewhat extravagance. Therefore, the total effort of 32 person days (18 PDs for marking up the original requirement document + 4 PDs for marking up the new requirement document + 8 PDs for adjusting the analysis tools + 2 PDs for summarizing and reporting the results). An expert on Java programming with experience in developing security software might have read through the requirements and come to a conclusion on the costs within a day. On the other hand, such an estimate cannot be readily explained. There are neither facts nor numbers behind it. It is based on intuition and human judgement. For smaller projects it may work. The requirement document here is too big and too complex for an average developer. He or she needs a model to structure their thoughts and to make meaningful associations. Besides the requirement model provides other advantages above and beyond cost estimation. Sizing a system is only one reason for modelling. There are several other reasons such as establishing a baseline to trace requirements through a finished system.

8 Comparing estimated with actual Costs

All papers on cost estimation are confronted with the demand to compare the estimated costs with the actual costs. This author would be more than glad to satisfy that demand but there are two major barriers to overcome. One barrier is that of timing. In the case of large scale projects like this one there is a major time gap between the time the project is estimated and the time it is completed. This project is still pending, meaning it has yet to start. Therefore, there is no data to compare. The purpose of the estimation project was to collect data on whether the project would be feasible. The calculated data was

compared with the costs projected by the responsible health department managers. When they perceived that the effort calculated was within plus/minus 20% of what they projected they were satisfied. The cost estimation had served their purpose.

What could have been done was to transpose the costs of developing the current system to the planned new subsystem based on the ratio of their sizes. According to a comparison of the two requirement models, the new system should cost circa 1/3 of the original system as a whole. If the original system had taken 240 person months to development then the new subsystem would take at least 80 person months, provided the same persons are working on the project. This is a good example of cost estimation by analogy [ShSc97]. But then follows the other barrier.

The other barrier is that of organizational confidentiality. No organization is particularly open about their development costs, their productivity and their product quality. These topics are highly sensitive. In this respect industry is more secretive than the military. Thus, even if the project had been completed it would still be very difficult to get data on the accumulated costs. Managers are concerned that the data will in some way be used against them. For this reason project consultants are obliged to sign several non-disclosure agreements before they are allowed to take part in a project. Unfortunately, this also applies to projects in the public domain like this one.

9 Conclusions and further Work

The effort estimation of a system enhancement project reported on in this paper shows how difficult it is to predict the costs of evolution projects on the basis of fuzzy requirement specifications. The estimators were confronted with two fuzzy requirement documents – one for the existing security system and one for the proposed security system. One solution might have been to measure the size of the existing code, which was actually done, and to match the code metrics to the actual costs. Then one might have guessed how much less the system extension is compared to the original system. As measured here by comparing the two requirement models, it was 1/3 of the original system. Then it should cost no more than 1/3 of what the original system had cost. Unfortunately, you cannot compare an informal model with a formal one. They simply do not match, especially if they are at different semantic levels. At the code and for the most part at the design level, the model entity types are of a technical nature, e.g. modules, procedures, tables and files. At the requirement level model entity types are more of a business nature, e.g. business processes, use cases, business rules, logical entities, data attributes and business interfaces. In some cases they may match to corresponding technical model types. In many cases they do not match. The reason for having two models is that there are two classes of users – business users and technical users, each with their own language. They could in fact be united, only business-oriented persons insist on speaking their own language. So the IT world remains divided with all the consequences that go along with a divided world – redundancy, inconsistency and lack of traceability [Parn77].

Bibliography

[AB00] Abran,A.; Khelifi,A.; Buglione,L.: A System of Reference for Software Measurement with ISO 19761 (COSMIC FFP) in: Abran: Software Measurement – Research and Application, Shaker Publ., Aachen, 2004, pp. 89-108

[Anda01] Anda, B.: Comparing Effort Estimates based on Use-Case Points with Expert Estimates, Proc. of IEEE ICSE 2001, Computer Society Press, Toronto, Oct. 2001, S. 218

[Bria17] Briand, L.: "Analyzing Natural Language Requirements – The not too sexy but curiously difficult research that industry needs", 23rd Int. Conference on Requirements Engineering, Foundation for Software Quality, LNCS 10153, Springer, 2017, p. 131

[BuDe08] Bundschuh, M./ Dekkers, C.: The IT Measurement Compendium, Springer Verlag, Berlin, 2008

[CMPB05] Chen, Z.; Menzies,T.; Port, D.; Boehm, B.: „Finding the right Data for Software Cost Modelling", IEEE Software, Nov. 2005, p. 38

[ChGa01] Chechik, M.; Gannon, J.: „Automatic Analysis of Consistency between Requirements and Designs", IEEE Trans. on S.E., Vol. 27, Nr. 7, July 2001, p. 651.

[Cohn06] Cohn, M.: Agile Estimating and Planning, Prentice-Hall, Upper Saddle River, N.J., 2006, p. 35

[DuFo96] Dumke, R.; Foltin, E.: Softwarequalität durch Meßtools, Vieweg Verlag, Wiesbaden, 1996, p. 97-104

[GDG06] Greevy, O.; Ducasse, S.; Girba, T.: "Analyzing Software Evolution through Feature Views", Journal of Sw Maintenance&Evolution, Vol. 18, No. 6, Dec. 2006, p. 425

[IFPG99] International Function-Point Users Group, Function-Point Counting Practices, Release 4.1, IFPUG, Westerville, Ohio, 1999

[Jantz08] Jantzen, K.: "Verfahren der Aufwandsschätzung für komplexe Softwareprojekte von heute", InformatikSpektrum, Band 31, Nr. 1, 2008, p. 35

[Karn93] Karner, G.: Metrics for Objectory, Master's Thesis, University of Linköping, Sweden, Nr. Lith-IDA-Ex-9344:21, Dez. 1993, p. 21

[Lefi11] Lefingwell, D.: Agile Software Requirements, Addison-Wesley, UpperSaddle River, N.J., 2011, p. 258

[Mugr08] Mugridge, J.: „Managing Agile Project Requirements with StoryTest-driven Development", IEEE Software, Jan. 2008, p. 68

[Parn77] Parnas, D.: "The use of precise Specifications in the development of Software", Proc. of IFIP Congress-77, North-Holland, Amsterdam, 1977, p. 860

[Selb09] Selby, R.: „Analytics-driven Dashboards enable leading Indicators for Requirements and Designs of Large-Scale Systems", IEEE Software, Jan. 2009, p. 41

[Sned96] Sneed, H.: „Estimating the Development Costs of Object-oriented Software", Proc. of

7th European Software an Control Metrics Workshop, Wilmslow, GB, 1996, June 1996, p.135

[Sned05] Sneed, H.: “Reverse Engineering deutschsprachiger Fachkonzepte”, Software-Technik Trends, Vol. 25, No. 2, May, 2005, p. 45

[Sned05b] Sneed, H.: Software Projektkalkulation–Praxiserprobte Methoden der Aufwandsschätzung verschiedener Projektarten, Hanser Verlag, München, 2005, p. 41

[Sned10] Sneed, H.: „Vergleich zweier Aufwandsschäztungen nach Function-Point und UseCase Point“, DASMA Metrikon2010, Shaker Verlag, Kaisersläutern, Nov. 2010, p. 52

[SnHa05] Sneed,H.; Hasitschka,M.; Teichmann,M.-T.: Software-Produktmanagement, dpunkt Verlag, Heidelberg, 2005.

[SnPr18] Sneed,H.; Prentner, W.: ”Requirements Reengineering”, Objektspektrum, Nr. 6, Dec. 2018, p30

[ShSc97] Shepperd, M.; Schofield,C: “Estimating Software Project Effort using Analogies”, IEEE Trans. on SE, Vol. 23, No. 12, Nov. 1997, p. 736

[Ster10] Sterling, C.: Managing Software Debt – Building for inevitable Change, Addison-Wesley, Boston, 2011