

An Empirical Study of Requirements-to-Code Elaboration Factors

Ali Afzal Malik, Supannika Koolmanojwong, Barry W. Boehm

Center for Systems and Software Engineering – University of Southern California
Los Angeles, U. S. A.

alimalik@usc.edu, koolmano@usc.edu, boehm@usc.edu

Abstract. *During a software development project's lifecycle, requirements are specified at multiple levels of detail e.g. objectives, shall statements, use cases, etc. The elaboration of requirements proceeds from one level of detail to the other till the final level – lines of code – is reached. This empirical study conducts a quantitative analysis of this hierarchical process of requirements elaboration. Multi-level requirements data from 25 small e-services projects is examined and the ratios between the numbers of requirements at consecutive levels are determined. Knowledge of these ratios – called elaboration factors – can play a crucial role in early software cost estimation. A case study demonstrating the utility of this approach in producing an early size estimate of a large commercial project is also presented.*

Keywords: Cockburn metaphor, elaboration factor, empirical study, requirements elaboration, requirements level, software cost estimation, software requirement, software sizing.

1. Introduction

Requirements for a software development project are refined with the passage of time. At the beginning of a project, for instance, requirements may be represented by high-level objectives. As more information becomes available, these objectives are expanded into more specific shall statements. After some time, shall statements are transformed into use cases. Use cases, in turn, are elaborated by component steps. In this way, the process of requirements elaboration progressively adds more and more detail at each step.

Figure 1 illustrates the hierarchical process of requirements elaboration. The left column contains various requirements levels while the right column shows the corresponding altitude levels of the Cockburn metaphor [6]. These altitude levels help in visualizing the level of detail present at each step of elaboration. Cloud-level objectives, for instance, contain the least amount of detail while clam-level source lines of code (SLOC) contain the maximum amount of detail.

The aim of this empirical study is to gain a better understanding of the hierarchical refinement of requirements by quantifying the entire process of requirements elaboration. This is achieved by determining the elaboration factors between all pairs of consecutive requirements levels shown in Figure 1. The elaboration factor between any two consecutive requirements levels – say, A and B – indicates the ratio of the number of requirements at level A to the number of requirements at level B. For instance, the

elaboration factor between objectives and shall statements indicates the average number of shall statements obtained as a result of the elaboration of a single objective.

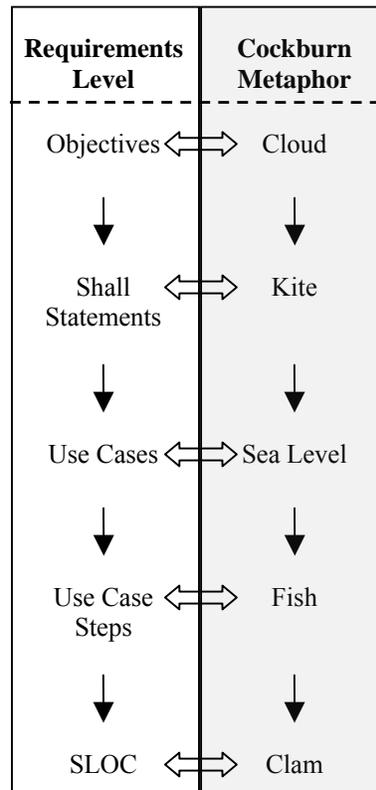


Figure 1. Requirements elaboration

Each project has its own set of elaboration factors. These elaboration factors can play a crucial role in early software cost estimation by enabling the use of higher-level requirements data as a measure of software size. This study analyzes multi-level requirements data of 25 small real-client e-services projects. Quantification of the elaboration factors of these projects provides an opportunity to compare different projects with respect to the elaboration of their requirements and to estimate the size of similar new projects.

The next section presents the motivation behind this study, mentions relevant prior work, and explains how this study is unique. The experimental setting is described in Section 3. This is followed by an outline of the data collection approach in Section 4. Section 5 summarizes the results and discusses the salient aspects of these results. A case study demonstrating the application of the approach defined in this empirical study on large commercial projects is presented in Section 6. Finally, Section 7 summarizes the contributions of this study and mentions future work planned in this area.

2. Motivation and Related Work

Knowledge of elaboration factors can be extremely valuable in the domains of system and software sizing and cost estimation. Most current cost estimation models use some notion of size as input. COSYSMO [16], for instance, uses high-level size notions such as

the number of itemized requirements. COCOMO II [3], on the other hand, uses low-level size notions such as SLOC. Each of these two categories of models poses a different problem at the time of early cost estimation.

Models relying on high-level notions of size need to ensure consistency in the level of detail of requirements being input and the level of detail of requirements used during calibration. Using, for instance, cloud-level requirements in a model that has been calibrated using sea-level requirements will result in under-estimation. This problem can be overcome by using the elaboration factors to convert from one requirements level to another. These elaboration factors can also be used to help managers identify requirements that were either incompletely specified or were specified at too high a level [14].

Models that employ a low-level notion of size avoid the problem of inconsistency between the levels of detail of inputs and calibration data but give rise to another serious problem. This problem clearly manifests itself at the time of a project's inception when only high-level requirements are known. In such a situation also, it is extremely beneficial to know the elaboration factors between various consecutive requirements levels so that one can derive the number of SLOC from the number of high-level requirements.

Cockburn's analysis of multiple goal levels [6] is one of the most comprehensive treatments of the process of requirements elaboration. He employs an altitude metaphor to describe the hierarchical refinement of goals in the context of use cases. Very high-level use cases are represented by clouds, high-level ones by kites, medium-level ones by sea waves, low-level ones by fish, and very-low level ones by clams. Cockburn's metaphor is used in Figure 1 to label the various levels of requirements.

While other works such as [1] and [10] have proposed ways of improving and facilitating the process of requirements elaboration, the work of Malik and Boehm [11] contains the first quantitative analysis of this process. New metrics are defined to quantify the elaboration of high-level goals into low-level requirements. The utility of these metrics is tested on concrete data obtained from small real-client projects. This empirical study is an extension of that work. It defines an approach to quantify the entire requirements elaboration process by measuring elaboration across all pairs of consecutive requirements levels starting from high-level objectives and ending at SLOC.

3. Experimental Setting

This empirical study examines multi-level requirements data of 25 small real-client e-services projects [13] done in the past few years by graduate students studying software engineering at USC [15]. These projects are listed in Table 1. Each of these 25 projects was done in two semesters (i.e. 24 weeks) by a team of 6 to 8 graduate students. Clients of these projects included USC-neighborhood organizations, USC departments, and USC Center for Systems and Software Engineering (CSSE) affiliate organizations.

Table 1. Projects used by empirical study

S#	Year	Project Name	Project Type
1	2004	Online Bibliographies on Chinese Religions	Web-based Database
2	2004	Data Mining of Digital Library Usage Data	Stand-alone Analysis Tool
3	2004	Data Mining from Report Files	Stand-alone Analysis Tool
4	2005	Data Mining PubMed Results	Stand-alone Analysis Tool
5	2005	USC Football Recruiting Database	Web-based Database
6	2005	Template-based Code Generator	Stand-alone Conversion Tool
7	2005	Web-based XML Editing Tool	Web-based Editor
8	2005	EBay Notification System	Stand-alone Analysis Tool
9	2005	Rule-based Editor	Stand-alone Database
10	2005	CodeCount™ Product Line with XML and C++	Stand-alone Analysis Tool
11	2006	USC Diploma Order and Tracking System	Web-based Database
12	2006	Student's Academic Progress Application	Web-based Database
13	2006	USC CONIPMO	Stand-alone Cost Model
14	2007	USC COINCOMO	Stand-alone Cost Model
15	2007	BTI Appraisal Projects	Web-based Database
16	2007	LAMAS Customer Service Application	Web-based Database
17	2007	BID Review System	Web-based Database
18	2007	Proctor and Test Site Tracking System	Web-based Database
19	2008	Master Pattern	Web-based Database
20	2008	Housing Application Tracking System	Web-based Database
21	2008	EZBay	Stand-alone Analysis Tool
22	2008	AAA Petal Pushers Remote R&D	Web-based Database
23	2008	Online Peer Review System for Writing Program	Web-based Database
24	2008	The Virtual Assistant Living and Education Program	Stand-alone Database
25	2008	Theatre Script Online Database	Web-based Database

Special care was exercised in selecting projects for this empirical study. To enable a complete end-to-end comparison of elaboration factors, projects with incomplete information on any level of requirements were filtered out. Similarly, COTS-intensive applications, for which requirements data is not readily available at all levels, were not considered. Thus, each of these 25 projects was a fully-documented custom-development project. Furthermore, each project followed an architected-agile process [4] such as MBASE [12], LeanMBASE [5], or Instructional ICM-Sw [8].

4. Method

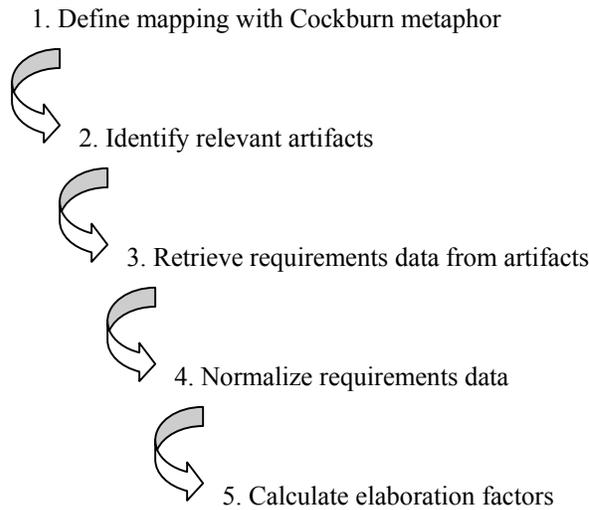


Figure 2. Data collection process

Figure 2 shows the process followed in collecting data for this empirical study. Step one involved defining a concrete mapping between the various levels of the Cockburn metaphor (see Figure 1) and the requirements levels of these e-services projects. This mapping is shown in Table 2. Each subsequent requirements level builds on the previous by adding more detail. Capability goals are high-level functional goals of a project. A typical capability goal taken from one of these 25 projects states: “Data Analysis: The system will analyze the subsets of data as requested by the user to calculate the required statistics.” Capability requirements, on the other hand, are more precise statements about the functional requirements of a project. A typical capability requirement specifies the pre-conditions, post-conditions, actors, inputs, outputs, sources, and destinations in addition to a specific textual description. The description of one of the capability requirements corresponding to the above-mentioned capability goal states: “The system shall count the number of times each journal appears in the imported dataset”. Use case steps include steps in the main scenario as well as the alternative scenarios of a use case.

Table 2. Mapping

Cockburn Metaphor	Requirements Level
Cloud	Capability goals
Kite	Capability requirements
Sea Level	Use cases
Fish	Use case steps
Clam	SLOC

The next step was identification of relevant artifacts that contained data for each requirements level. Table 3 contains the list of artifacts [5, 8, 12] examined for each e-services project. Information about a project’s high-level capability goals was present in the Operational Concept Description (OCD) document while its capability requirements were documented in the System and Software Requirements Definition (SSRD) document. The System and Software Architecture Description (SSAD) document contained information on use cases as well as their component steps. SLOC information was available from a project’s source code. In order to ensure consistency in the presence of multiple versions of each artifact, it was necessary to get each artifact from the same anchor point milestone [2]. For this purpose, the Initial Operational Capability (IOC) milestone was used since it corresponds to the end of the Construction phase [9]. At this point the product is ready to be transitioned to the client.

Table 3. Relevant artifacts

Requirements Level	Artifact
Capability goals	Operational Concept Description
Capability requirements	System and Software Requirements Definition
Use cases	System and Software Architecture Description
Use case steps	System and Software Architecture Description
SLOC	Source Code

Step three required retrieving relevant data from these artifacts. Capability goals, capability requirements, use cases, and use case steps were obtained from inspection of relevant documents. A project’s SLOC were obtained by running a code counter on its source code.

In order to ensure consistency in the level of detail at each requirements level and to improve the accuracy of analysis, the retrieved data was normalized in the next step. The normalization of the capability goals of a project included, for instance, merging related low-level capability goals into one high-level capability goal. As an example, one of these 25 projects specified “Add an entry”, “Edit an entry”, and “Delete an entry” as three separate capability goals. During the normalization process, these three capability goals were merged into one “Manage entry” capability goal. The normalization of capability requirements involved steps like splitting aggregate requirements into their individual components and treating each component as a separate capability requirement. One project, for instance, treated generating five different types of reports as one capability requirement. During normalization, each of these five types of reports was counted as a separate capability requirement. A similar procedure was followed in normalizing use cases and use case steps. SLOC were normalized by ensuring that auto-generated code was not counted.

Table 4. Formulae for elaboration factors

Elaboration Stage		Elaboration Factor Formula
<i>Cockburn Metaphor</i>	<i>E-Services Projects</i>	
Cloud-to-Kite	Capability Goals-to-Capability Requirements	$\frac{CR}{CG}$
Kite-to-Sea Level	Capability Requirements-to-Use Cases	$\frac{UC}{CR}$
Sea Level-to-Fish	Use Cases-to-Use Case Steps	$\frac{UCS}{UC}$
Fish-to-Clam	Use Case Steps-to-SLOC	$\frac{SLOC}{UCS}$

Once the requirements data had been normalized, the final step involved counting the number of capability goals (*CG*), capability requirements (*CR*), use cases (*UC*), use case steps (*UCS*), and SLOC and, thereafter, calculating the elaboration factor for each pair of consecutive requirements levels. Table 4 displays the formulae used for calculating the elaboration factors for these e-services projects. As is evident from these formulae, each elaboration factor is a ratio of the number of requirements at two consecutive requirements levels. The capability goals-to-capability requirements elaboration factor, for instance, is the ratio of *CR* and *CG*. The value of this ratio indicates the average number of capability requirements obtained as a result of the elaboration of a single capability goal.

5. Results and Discussion

Table 5. Elaboration factor sets for each project

S#	Project Name	$\frac{CR}{CG}$	$\frac{UC}{CR}$	$\frac{UCS}{UC}$	$\frac{SLOC}{UCS}$
1	Online Bibliographies on Chinese Religions	3.00	1.00	7.89	64.75
2	Data Mining of Digital Library Usage Data	3.67	0.18	9.00	118.56
3	Data Mining from Report Files	3.00	0.43	6.00	40.72
4	Data Mining PubMed Results	3.50	0.29	7.00	5.89
5	USC Football Recruiting Database	1.83	0.91	8.90	26.80
6	Template-based Code Generator	2.50	0.40	3.50	278.57
7	Web-based XML Editing Tool	1.33	1.25	5.80	140.52
8	EBay Notification System	2.75	0.82	2.44	38.27
9	Rule-based Editor	2.00	0.75	4.83	32.17

10	CodeCount™ Product Line with XML and C++	1.20	0.08	15.00	96.07
11	USC Diploma Order and Tracking System	1.60	1.22	3.45	31.63
12	Student's Academic Progress Application	1.60	0.89	3.63	15.76
13	USC CONIPMO	1.29	0.90	5.44	108.92
14	USC COINCOMO	2.80	0.38	6.67	199.15
15	BTI Appraisal Projects	3.20	1.00	5.75	23.74
16	LAMAS Customer Service Application	1.00	1.00	6.57	67.72
17	BID Review System	1.17	1.29	13.78	42.58
18	Proctor and Test Site Tracking System	1.90	0.84	8.38	74.46
19	Master Pattern	1.75	1.57	5.32	100.15
20	Housing Application Tracking System	3.00	2.88	8.63	16.84
21	EZBay	3.25	0.88	5.33	60.51
22	AAA Petal Pushers Remote R&D	4.17	1.00	6.48	4.29
23	Online Peer Review System for Writing Program	3.57	0.93	9.72	38.72
24	The Virtual Assistant Living and Education Program	3.57	0.88	9.68	16.49
25	Theatre Script Online Database	2.83	0.60	7.25	29.49



Figure 3. $\frac{CR}{CG}$ distribution

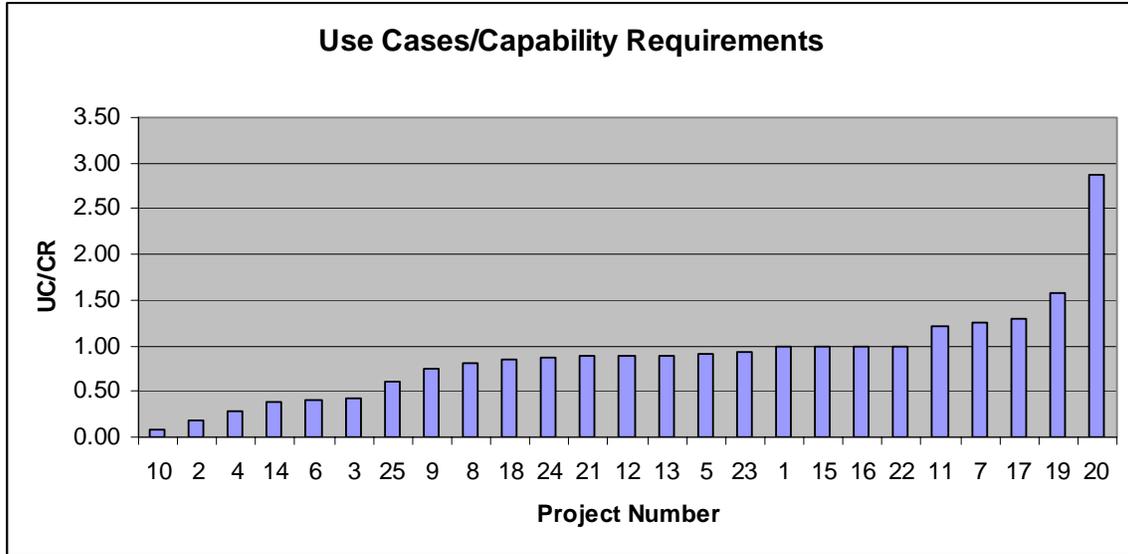


Figure 4. $\frac{UC}{CR}$ distribution

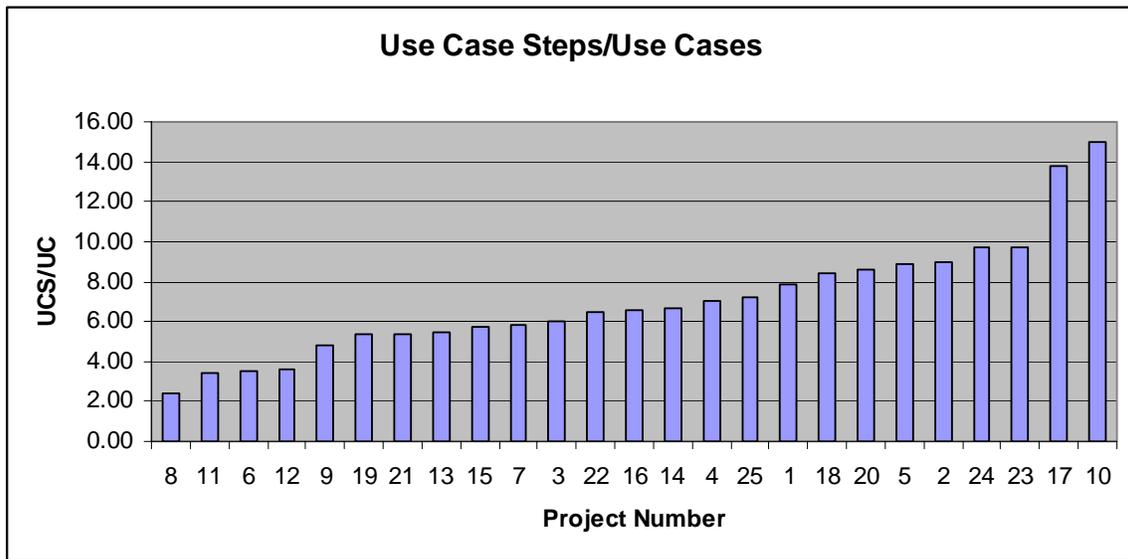


Figure 5. $\frac{UCS}{UC}$ distribution

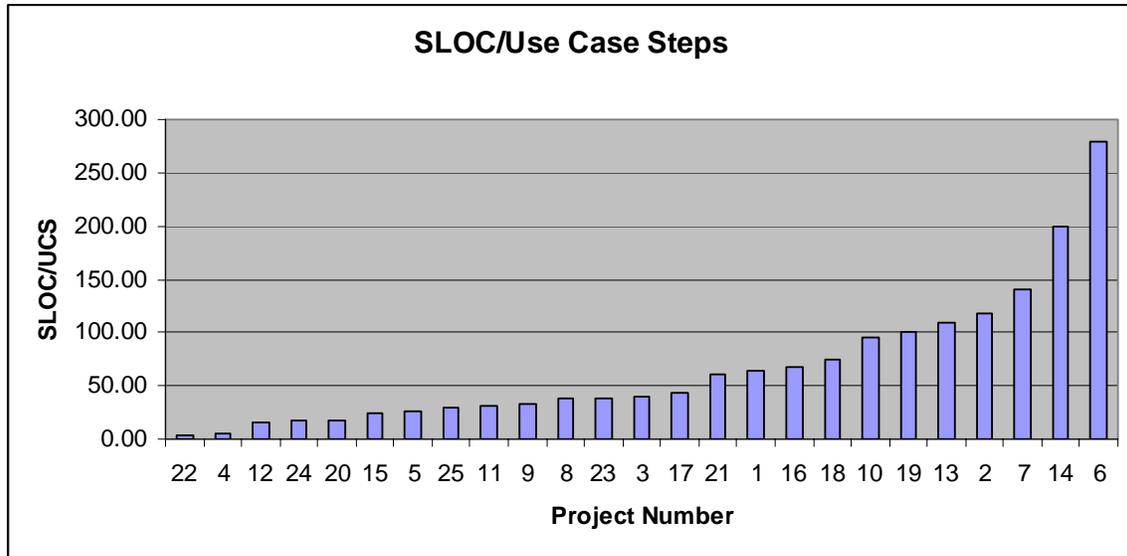


Figure 6. $\frac{SLOC}{UCS}$ distribution

Table 5 lists elaboration factor sets for each of the 25 e-services projects and Figures 3 – 6 display distributions of the four elaboration factors. This data is summarized in Table 6. On average, a capability goal is elaborated into about two or three capability requirements, a capability requirement maps to a single use case, a use case has about seven steps, and a use case step is implemented by about 67 SLOC.

Table 6. Summary of elaboration results

Statistic	Elaboration Factors			
	$\frac{CR}{CG}$	$\frac{UC}{CR}$	$\frac{UCS}{UC}$	$\frac{SLOC}{UCS}$
Average	2.46	0.89	7.06	66.91
Median	2.75	0.89	6.57	40.72
Standard Deviation	0.94	0.55	2.97	64.46

The capability requirements-to-use cases elaboration factor - $\frac{UC}{CR}$ - is almost one. In other words, on average, use cases were expressed at almost the same level as capability requirements. This discrepancy may be explained by the students' desire to get better grades. This desire motivated them to specify capability requirements at the same level as use cases. This also identifies an opportunity to save valuable time during the development of such projects. The absence of little or no value addition from the capability requirements level to the use cases level signals that activities in one of these

levels are effectively redundant. These activities, therefore, may be omitted without a significant impact on the project.

As shown by the relatively small standard deviation, the variation in the use cases-to-use case steps elaboration factor - $\frac{UCS}{UC}$ - is very little. This is confirmed by Figure 7 which plots the number of use case steps against the number of use cases for all 25 projects.

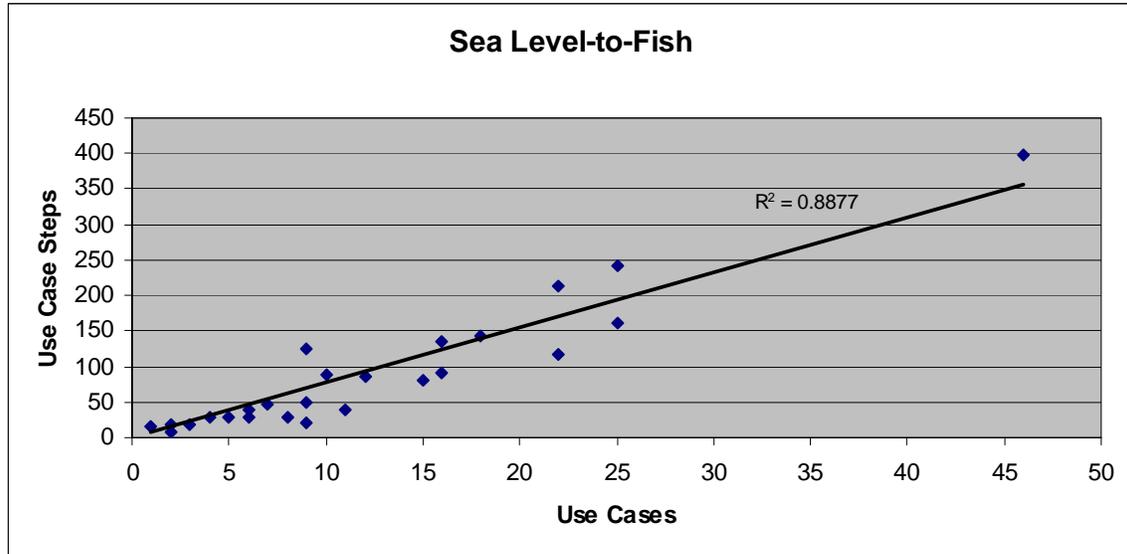


Figure 7. Elaboration of use cases to use case steps

The standard deviation of the use case steps-to-SLOC elaboration factor - $\frac{SLOC}{UCS}$ - is disproportionately large. This may be due to the fact that some SLOC are devoted to implementing level of service (LOS) requirements. Apart from SLOC, all other requirements levels examined in this study deal only with capability or functional requirements. Therefore, everything else being constant, a project with more LOS requirements will have more SLOC. This will result in a larger use case step-to-SLOC elaboration factor even though the number of use case steps remains unchanged.

The information summarized in Table 6 can be easily used to estimate the SLOC of a similar new e-service project at the time of its inception. Once the high-level capability goals of this project have been determined, the following formula can be used to estimate the number of SLOC for this project:

$$SLOC = CG \times \frac{CR}{CG} \times \frac{UC}{CR} \times \frac{UCS}{UC} \times \frac{SLOC}{UCS}$$

The first term on the right-hand side of the above equation is the number of capability goals and the remaining four terms represent the elaboration factors defined in Table 4. Consider, for instance, a new e-service project that has identified 5 high-level capability goals at inception. The most likely SLOC for this project can be estimated by using the

above equation and plugging-in the value of $CG (= 5)$ along with the average-values of elaboration factors shown in Table 6.

$$\text{Most Likely SLOC} = 5 * 2.46 * 0.89 * 7.06 * 66.91 \approx 5171$$

A one-standard deviation SLOC-range can also be calculated for this project using the standard deviations shown in Table 6.

$$\begin{aligned} \text{Minimum SLOC} &= 5 * (2.46 - 0.94) * (0.89 - 0.55) * (7.06 - 2.97) * (66.91 - 64.46) \\ &\approx 26 \end{aligned}$$

$$\begin{aligned} \text{Maximum SLOC} &= 5 * (2.46 + 0.94) * (0.89 + 0.55) * (7.06 + 2.97) * (66.91 + 64.46) \\ &\approx 32,256 \end{aligned}$$

6. Requirements Elaboration of Commercial Projects

This technique of analyzing a project's multi-level requirements data to determine its set of elaboration factors can be applied to large commercial projects as well. An opportunity to conduct such an analysis was provided by ballpark data in [7] on multiple requirements levels of a certain category of large IBM projects. Table 7 summarizes this data.

Table 7. Multi-level requirements data of large IBM projects

Requirements Level	Number of Requirements
Business events	50
Use cases	200
Main steps	1,500
Detailed operations	20,000
SLOC	1,500,000

Each requirements level shown in Table 7 can be mapped to an altitude level of the Cockburn metaphor. Table 8 shows this mapping along with the set of elaboration factors of these large IBM projects. On average, a business event is elaborated into four use cases, a use case contains about seven or eight main steps, a main step is expressed by about thirteen detailed operations, and a detailed operation is implemented by seventy-five SLOC.

Table 8. Elaboration factors of large IBM projects

Elaboration Stage		Elaboration Factor
<i>Cockburn Metaphor</i>	<i>Large IBM Projects</i>	
Cloud-to-Kite	Business Events-to-Use Cases	4.00
Kite-to-Sea Level	Use Cases-to-Main Steps	7.50
Sea Level-to-Fish	Main Steps-to-Detailed Operations	13.33

Fish-to-Clam	Detailed Operations-to-SLOC	75.00
--------------	-----------------------------	-------

Elaboration factors obtained from this analysis can be helpful in early sizing and cost estimation of similar future projects. Using these elaboration factors, an estimate of the size of a new project can be obtained as early as the inception stage. Consider, for instance, a similar large IBM project that identifies 60 business events at the time of inception. Based on the elaboration factors in Table 8, a rough sizing of this project would be about $(60 * 4 * 7.5 * 13.33 * 75 \approx) 1,800$ KSLOC.

However, even within IBM, one would have to be careful about binding such an estimate into a firm commitment. If the project is highly similar, much of its software could be reused from previous projects at much less cost. If it is dissimilar, it is not clear how to adjust the estimate. The strongest approach would be to obtain distributions of elaboration factors such as the ones shown for eservices projects in Figures 3 – 6, and to determine confidence intervals for SLOC estimates based on these distributions for one’s own organization.

7. Conclusions and Future Work

This empirical study has defined a novel approach to quantify the entire requirements elaboration process and has demonstrated the application of this approach on concrete data of real projects. Multi-level requirements data of 25 small real-client e-services projects was collected, processed, and analyzed to determine the complete set of elaboration factors for these projects. The same approach was also applied in determining the elaboration factors of a certain category of large commercial projects. Knowledge of these elaboration factors can be extremely useful for estimating the size and, hence, the cost of similar future projects.

Even though this work has defined the framework essential for quantitatively measuring all stages of requirements elaboration a few tasks are still pending. Firstly, the causes of variation in elaboration factors of projects in the same development environment need to be pinned down. As is evident from the distributions shown in Figures 3 – 6, the variance in some elaboration factors is more than the others. A precise framework for predicting this variance needs to be defined so that more guidance is available for early size and cost estimation when employing these sets of elaboration factors.

Secondly, other mappings between requirements levels and altitude levels of the Cockburn metaphor need to be explored. The mapping used by this study is not cast in stone. Other mappings, such as those using classes and operations instead of use cases and use case steps, are also valid. These different mappings will provide further valuable insights in the process of requirements elaboration. Moreover, the elaboration data of more real projects needs to be collected and analyzed. This will help in constructing a database of elaboration factors for projects of different categories and sizes. Such a database will be extremely valuable during early cost estimation of similar projects.

7. References

- [1] Antón, A. I. 1996. Goal-Based Requirements Analysis. In Proceedings of the 2nd IEEE International Conference on Requirements Engineering (Colorado Springs, CO, USA, April 15 – 18, 1996). ICRE '96.
- [2] Boehm, B. 1996. Anchoring the software process. *IEEE Software* 13, 4 (1996), 73-82.
- [3] Boehm, B., Abts, C., Brown, A. W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D., and Steece, B. 2000 *Software Cost Estimation with COCOMO II*. Prentice Hall.
- [4] Boehm, B. and Lane, J. 2008. A Process Decision Table for Integrated Systems and Software Engineering. In Proceedings of the Conference on Systems Engineering Research (Los Angeles, CA, USA, April 04 – 05, 2008). CSER '02.
- [5] Boehm, B. et al. 2006 Guidelines for Lean Model-Based (System) Architecting and Software Engineering (LeanMBASE). University of Southern California. http://greenbay.usc.edu/csci577/spring2006/site/guidelines/LeanMBASE_Guidelines_V1.5.pdf
- [6] Cockburn, A. 2001 *Writing Effective Use Cases*. Addison-Wesley.
- [7] Hopkins, R. and Jenkins, K. 2008 *Eating the IT Elephant: Moving from Greenfield Development to Brownfield*. IBM Press.
- [8] Instructional ICM-Sw Electronic Process Guidelines (EPG), <http://greenbay.usc.edu/IICMSw/index.htm>
- [9] Kruchten, P. 2003 *The Rational Unified Process: An Introduction*. Addison-Wesley.
- [10] Letier, E. and van Lamsweerde, A. 2002. Agent-Based Tactics for Goal-Oriented Requirements Elaboration. In Proceedings of the 24th IEEE International Conference on Software Engineering (Orlando, FL, USA, May 19 – 25, 2002). ICSE '02.
- [11] Malik, A. A. and Boehm, B. 2008. An Empirical Study of Requirements Elaboration. In Proceedings of the 22nd Brazilian Symposium on Software Engineering (Campinas, Brazil, October 13 – 17, 2008). SBES '08.
- [12] MBASE website, <http://sunset.usc.edu/csse/research/mbase/>
- [13] Projects archive, <http://ebase.usc.edu/Projects/index.jsp>
- [14] Selby, R. W. 2009. Analytics-Driven Dashboards Enable Leading Indicators for Requirements and Designs of Large-Scale Systems. *IEEE Softw.* 26, 1 (Jan. 2009), 41-49. DOI= <http://dx.doi.org/10.1109/MS.2009.4>
- [15] USC Software Engineering Course Website, <http://greenbay.usc.edu/csci577/fall2008/site/index.html>
- [16] Valerdi, R. 2005 *The Constructive Systems Engineering Cost Model (COSYSMO)*. PhD Dissertation. Department of Computer Science, University of Southern California