An Empirical Study of Requirements Elaboration

Ali Afzal Malik, Barry Boehm

Center for Systems and Software Engineering – University of Southern California
Los Angeles, U. S. A.
alimalik@usc.edu, boehm@usc.edu

Abstract. *This paper describes an empirical study undertaken to investigate the quantitative aspects of the phenomenon of requirements elaboration which deals with the transformation of high-level goals into low-level requirements. Prior knowledge of the magnitude of requirements elaboration is instrumental in developing early estimates of a project's cost and schedule. This study examines the data on capability goals and capability requirements of 20 real-client, MS-student, team projects done at USC. Metrics for data collection and analysis are described along with the utility of the results they produce. These results suggest some relationship between the nature of projects and the size of requirements elaboration.*

1. Introduction

Early estimation of a software project's cost and schedule requires a viable set of parameters for estimating a project's size. One such parameter is the number of unique requirements to be satisfied by the software. However, as illustrated graphically in [Cockburn 2001], the same requirement at the top (goal) level may be elaborated into several requirements at an intermediate level, and a large number of requirements at the detailed acceptance-test level.

An opportunity to analyze the requirements elaboration phenomenon has been provided by USC's annual two-semester series of real-client, MS-student, team-project courses. These two project-based courses – Software Engineering I (SE I) and Software Engineering II (SE II) – allow students to get a first-hand experience of a considerable chunk of the software development life cycle. According to the terminology of the Rational Unified Process (RUP) [Kruchten 2003], SE I exposes them to the Inception and Elaboration phases whereas SE II deals with the Construction and Transition phases.

Most of the time students work in development teams of four to six people. The projects provided to these teams come from a variety of sources including the industry and academia. By negotiations with the client the requirements for these projects are prioritized to facilitate completion in two semesters.

This study deals with 20 such two-semester-long projects [SE I 2008, SE II 2008] done by student teams in the past few years. Table 1 presents a summary of these projects. The column titled 'Year' indicates the year when the project was initiated. Special care was taken in selecting these projects for analysis. COTS-based projects and custom-development projects with incomplete data were filtered out. Thus, each of these 20 projects is a fully-documented custom-development project. In addition to this,

each project followed the same MBASE/RUP [Boehm et al. 2005, Kruchten 2003] development process.

Table 1. Projects summary

S#	Year	Project	Type
1	2004	Bibliographies on Chinese Religions in Western Languages	Web-based database
2	2004	Data Mining of Digital Library Usage Data	Data mining
3	2004	Data Mining from Report Files	Data mining
4	2005	Data Mining PubMed Results	Data mining
5	2005	USC Football Recruiting Database	Web-based database
6	2005	Code Generator – Template based	Stand-alone application
7	2005	Develop a Web Based XML Editing Tool	Web-based application
8	2005	EBay Notification System	Stand-alone application
9	2005	Rule-based Editor	GUI
10	2005	CodeCount™ Product Line with XML and C++	Code Counter Tool
11	2006	California Science Center Newsletter System	Web-based database
12	2006	California Science Center Event RSVP System	Web-based database
13	2006	USC Diploma Order/ Tracking Database System	Web-based database
14	2006	USC Civic and Community Relations web application	Web-based database
15	2006	Student's academic progress web application	Web-based database
16	2006	New Economics for Woman (NEW)	Web-based database
17	2006	Web Portal for USC Electronic Resources	Web-based GUI
18	2006	Early Medieval East Asian Tombs	Web-based database
19	2006	USC CONIPMO	Cost model
20	2006	An Eclipse Plug-in for Use Case Authoring	Stand-alone application

To provide an overview of the development process used by these projects, a part spanning just the Inception and Elaboration phases is sketched in Figure 1. Boxes represent the various steps carried out in the process. Multiple activities within a step are presented as a bulleted list. Each step has been numbered for convenience. Arrows signify the general flow of results/artifacts between the steps in the process. Step 3, for instance, uses the results produced by both step 1 and step 2 to produce a result which, in turn, is used by step 4. Steps 4 through 8, though presented as a sequence, contain several concurrent activities. As explained later in Section 3, the Operational Concept Description (OCD) and the System and Software Requirements Definition (SSRD) are the two documents [Boehm et al. 2005] we focus on in this study.

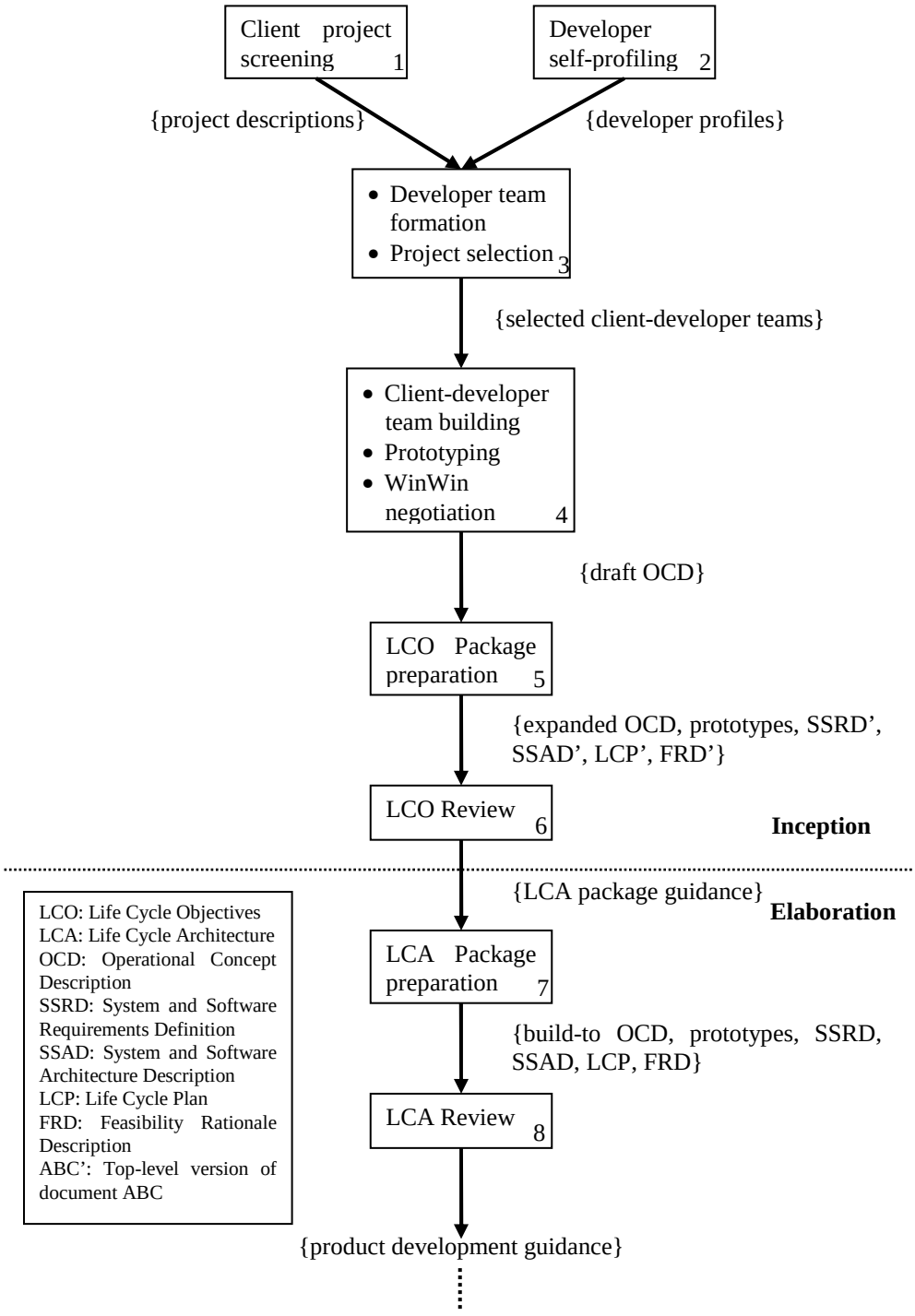

Figure 1. Partial development process

At the time of inception, most projects are specified in terms of informal statements of the goals wished to be achieved. With the passage of time, and as the understanding of the problem and its domain matures, these goals are transformed into much clearer low-level requirements. The aim of this study is to quantify and measure this phenomenon of requirements elaboration across different types of projects thereby gaining some useful insights about its utility with respect to early software cost and schedule estimation. The main contribution of our work lies in defining the metrics for quantifying requirements elaboration and analyzing the results produced by these metrics on concrete data.

The rest of this paper is organized as follows. Section 2 briefly discusses some related work and gives the motivation behind this study. Section 3 presents the methodology used for gathering the data required for this study. It defines appropriate metrics and indicates the rationale for their usage. Section 4 summarizes the results while section 5 comments on the salient aspects of these results. Finally, section 6 concludes with a brief outline of future work in this area.

2. Motivation and Related Work

Researchers in the past have looked at a number of ways of supporting and improving the process of requirements elaboration. Letier and van Lamsweerde proposed an agent-based approach towards requirements elaboration [Letier and van Lamsweerde 2002]. Several formal tactics for refining goals and then assigning them to single agents (e.g. human stakeholders, software components etc.) that can realize these goals were defined. Earlier, Antón had described the GBRAM (Goal-Based Requirements Analysis Method) and the results of its application in a practical setting [Antón 1996]. Using the GBRAM goals were identified and elaborated for later operationalization into requirements.

While the research done in this area so far has focused on facilitating and improving the process we have adopted an altogether different approach. We define metrics to analyze the process itself. This quantitative approach is augmented by looking at some qualitative aspects of the projects we scrutinize. In particular, we look at the differences in elaboration with respect to the type of the projects.

A better understanding of this process will be of great value in areas such as software sizing and software cost estimation. Most software cost estimation models such as Putnam's SLIM [Putnam 1978], RCA's PRICE-S [Freiman and Park 1979], and COCOMO II [Boehm et al. 2000] rely on software size as one of their primary inputs. Information about software size, however, is not available at the early stages of the project life cycle. The only information available at this time is about the overall goals and nature of the project. If this information can somehow be leveraged to estimate the size of the project then cost estimates made at the time of inception would be much more accurate. This, in turn, requires a thorough understanding of the transformation of and the relationship between high-level goals and low-level requirements. This study is a step forward in developing such an understanding.

3. Methodology

As an approximation to the phenomenon of requirements elaboration we examine the relationship between the capability goals and the capability requirements of these 20 projects. Capability goals and capability requirements represent, respectively, the functional goals and the functional requirements of a project. A typical capability goal, taken from one of these 20 projects, states: "Integrate all existing USC libraries resources search services into a single web-based portal". This goal is later refined into multiple implementable capability requirements. The description of one of these low-level requirements states: "A user shall be able to perform searches through either web feat or serial solutions through a single portal webpage".

A number of metrics have been gathered to examine the relationship between the capability goals and the capability requirements. All metrics related to the capability goals are collected from documentation produced at the time of the Life Cycle Objectives (LCO) milestone [Boehm 1996] which is achieved at the end of the Inception phase. Specifically, we gather data from the Operational Concept Description (OCD) document [Boehm et al. 2005] delivered by teams as part of their LCO package [SE I 2008]. This is represented as "expanded OCD" in Figure 1.

When dealing with capability requirements we examine both nominal and off-nominal requirements since all types of capability goals specified in OCD are considered. In other words, we consider the requirements of system behavior in normal as well as abnormal conditions. The System and Software Requirements Definition (SSRD) document [Boehm et al. 2005] delivered at the end of the Construction phase (not shown in Figure 1) is the source of all metrics related to capability requirements. The end of the Construction phase corresponds to the Initial Operational Capability (IOC) milestone [Boehm 1996]. Therefore, all capability requirements-related metrics have been gathered from the SSRD document which is a part of the last IOC Working Set. Occasionally, when the deliverables of the last IOC Working Set are not available in the project archives [SE II 2008], we have used the SSRD document present in the As-built Specification for collection of our metrics. At worst, this slight inconsistency in our data makes an insignificant difference to our analysis.

Table 2. Metrics summary

S#	Metric	Description
1	NCG_I	Number of initial capability goals
2	NCG_R	Number of capability goals removed
3	NCR_D	Number of delivered capability requirements
4	NCR_N	Number of new capability requirements
5	NCG_A	Number of adjusted capability goals
6	NCR_A	Number of adjusted capability requirements
7	EF	Elaboration Factor

Table 2 summarizes the metrics we have employed in our study. Out of these seven metrics the first four are collected directly from the project documentation by inspection as described above. The first two (NCG_I and NCG_R) are related to capability

goals whereas the next two (NCR_D and NCR_N) pertain to capability requirements. NCG_I represents the number of capability goals specified during the Inception phase whereas NCG_R specifies the number of capability goals that were removed or not considered during the later phases of the project. NCR_N indicates the number of capability requirements that were added later-on in the project and had no relationship to the goals included in NCG_I. This metric gives a crude indication of the extent of the requirements- or feature-creep phenomenon. NCR_D simply records the number of capability requirements satisfied by the product delivered to the client upon reaching the IOC milestone.

The last three metrics (NCG_A, NCR_A, and EF) are derived metrics. These are calculated according to the following formulae:

$$NCG_A = NCG_I - NCG_R$$

$$NCR_A = NCR_D - NCR_N$$

$$EF = NCR_A / NCG_A$$

As is clear from the formulae above, NCG_A and NCR_A are adjustments while EF is a ratio of these adjusted metrics. NCG_A signifies the capability goals that were retained till the end of the project while NCR_A indicates the capability requirements that emerge solely from these retained capability goals as opposed to being introduced later on. The EF metric quantifies the phenomenon of requirements elaboration. Projects with more adjusted capability requirements per adjusted capability goal have higher EF values.

Ranges of EF values can be used to classify projects in different groups. Keeping in view the nature of projects encountered in our academic setting and based on the data we have observed (see Section 4) we have come up with a simple criterion for defining these groups. This criterion is depicted in Figure 2. The continuous spectrum of EF values shown in this figure by a horizontal line is divided into four sections. It is trivially true that all EF values must be positive since EF is a ratio of counts. Moreover, any project with an EF value less than 1 is an outlier. This is because, under normal circumstances, NCR_A is at least as large as NCG_A. Projects with EF values between 1 and 1.5 (both inclusive) are assigned to the Low Elaboration Factor (LEF) group while those with EF values between 1.5 and 2 (inclusive) constitute the Medium Elaboration Factor (MEF) group. Finally, projects with EF values greater than 2 form the High Elaboration Factor (HEF) group. In our setting, even though it is hard to imagine a project with an EF value of greater than 10 the upper bound of the range for the HEF group has been left unspecified to accommodate exceptional cases. Here it must be mentioned that the EF ranges defining these project groups may be tailored according to the data observed in a particular setting.

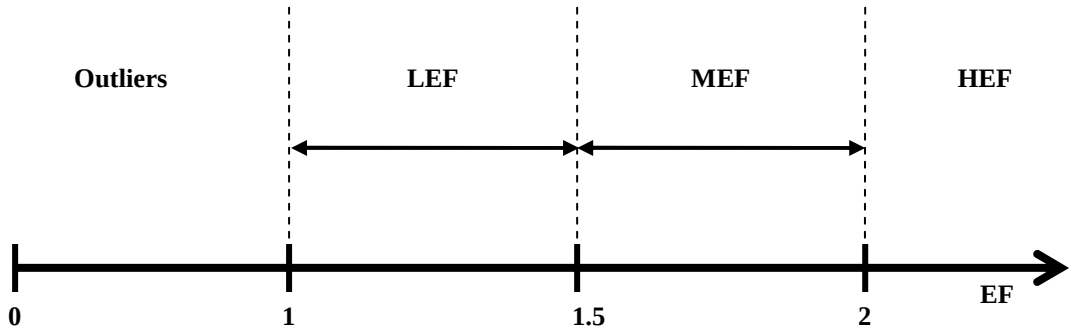

Figure 2. EF ranges defining groups

4. Results

Table 3 lists the values of the seven metrics introduced in the previous section for each of the 20 projects. For the sake of brevity, project names have been omitted from this table. Serial numbers (first column), however, have been retained for reference. These correspond to the serial numbers in Table 1. Also, for the sake of convenience, the data in this table is presented in ascending order of EF values.

Table 3. Project data for different metrics

S#	NCG_I	NCG_R	NCR_D	NCR_N	NCG_A	NCR_A	EF	Group
10	14	2	10	1	12	9	0.75	Outliers
19	8	1	8	2	7	6	0.86	
3	3	1	7	5	2	2	1	LEF
16	5	2	3	0	3	3	1	
7	10	5	6	1	5	5	1	
1	12	3	12	2	9	10	1.11	
8	10	2	12	2	8	10	1.25	
9	7	4	8	4	3	4	1.33	
2	3	0	9	4	3	5	1.67	MEF
20	5	2	7	2	3	5	1.67	
17	7	1	21	10	6	11	1.83	
6	4	1	7	1	3	6	2	
4	5	1	14	6	4	8	2	
15	5	1	11	3	4	8	2	
14	3	0	10	3	3	7	2.33	HEF
18	6	0	20	5	6	15	2.5	
5	4	1	12	3	3	9	3	
13	2	0	11	3	2	8	4	
12	6	2	19	2	4	17	4.25	
11	8	5	16	3	3	13	4.33	

The last column (Group) categorizes these 20 projects according to the groups defined in Section 3. Note that the first two rows (projects with serial numbers 10 and 19) are clear outliers since their EF values are less than 1. In other words, they underwent a 'negative' elaboration. A detailed examination of these two projects helped in understanding the reason behind this anomalous behavior. In the first case (serial # 10) a few pairs of capability goals were merged into single capability requirements whereas in the second case (serial # 19) the system was so well-understood that the capability goals were already specified at the level of capability requirements. These two outliers have been omitted from all subsequent analysis.

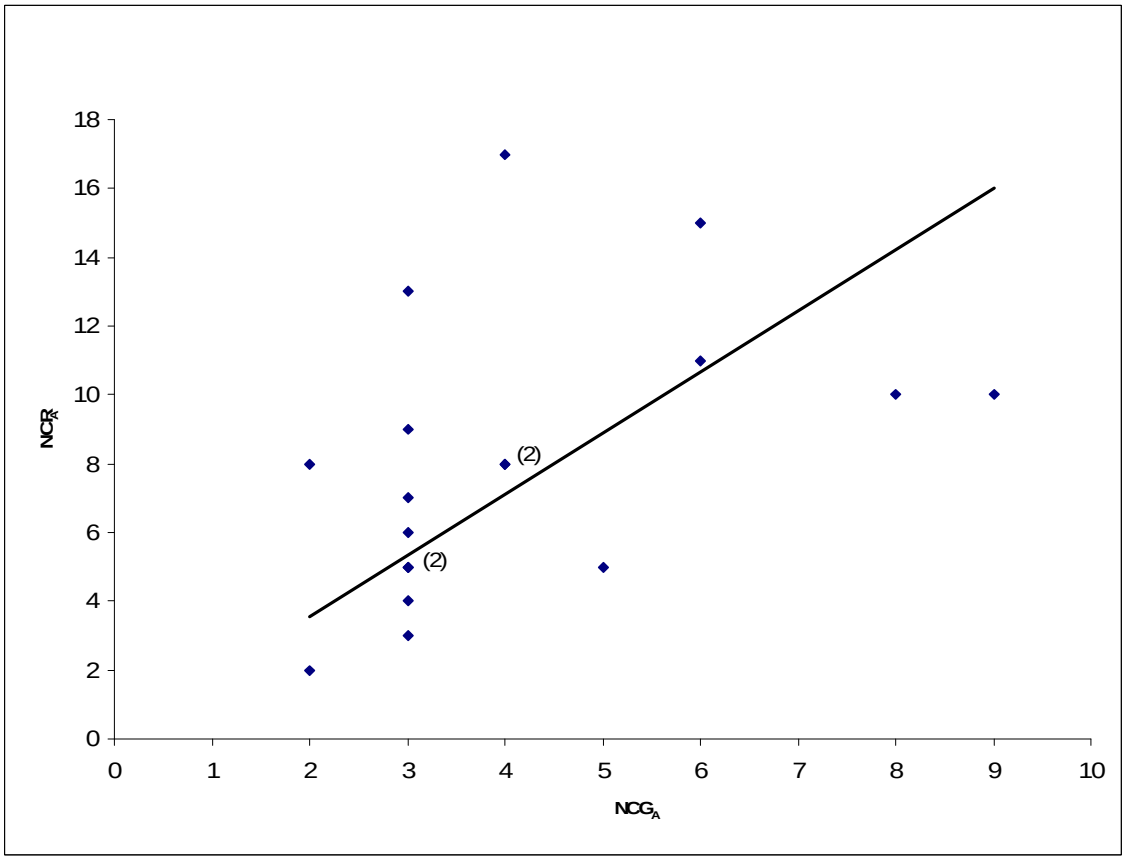

Figure 3. NCR_A vs. NCG_A

Figure 3 displays the graph obtained by plotting NCR_A against NCG_A. By default each dot represents a single project. However, in the event where two or more projects have identical values of the pair (NCG_A, NCR_A), we indicate this by annotating the dot with a parenthesized number specifying the number of projects aggregated by that dot. Thus, as is apparent from this figure, 2 of the 16 dots represent two projects. A regression line has also been added for convenience. Here it must be pointed out that the y-intercept of this line (and all subsequent regression lines) has been set to zero. This makes intuitive sense since a project that has no capability goals to start with will not exhibit the phenomenon of requirements elaboration. Though new requirements may be added, these will be discounted in the adjusted metric NCR_A.

5. Discussion

A glance at Figure 3 indicates a roughly increasing relationship between the two metrics: NCR_A and NCG_A. This relationship, however, is not very strong and there is a lot of variation around the fitted regression line. A careful observation of the same figure reveals something very subtle – the presence of different groups of elaboration. This subtlety is made apparent in Figure 4 which displays three regression lines along with their equations and coefficient of determination (R^2) values. This figure divides the remaining 18 data points into three distinct groups (LEF, MEF, and HEF) as defined in Section 3.

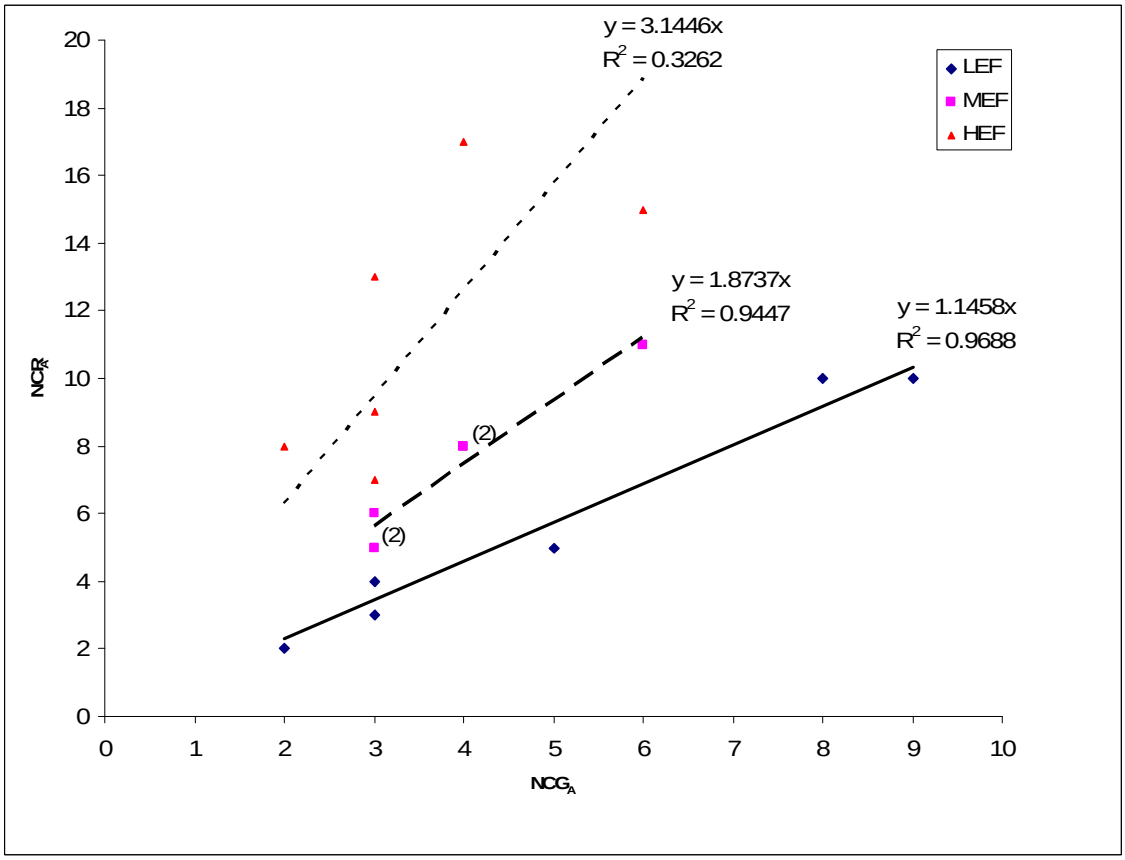

Figure 4. NCR_A vs. NCG_A with project groups identified

Points around the solid line at the bottom of Figure 4 represent projects with the lowest EF values (1 – 1.33). The dashed line in the middle fits points that correspond to projects with intermediate EF values (1.67 – 2). Finally, points around the dotted line at the top map to projects with the highest EF values (2.33 – 4.33). This classification is also indicated in the last column of Table 3.

It is obvious from these results that, as one might suspect, there is no one-size-fits-all formula for requirements elaboration. Depending on how well they are understood at the Inception phase, different projects will undergo different rates of elaboration. Knowledge of a project's type, however, can give some sort of an indication of its EF value even at the time of inception. A closer examination of the projects in the HEF group revealed that all of these projects were of type "Web-based Database" (see Table 1). These were the ones that underwent extensive elaboration. Other factors such

as the project's complexity and novelty also need to be considered. These additional factors could explain why, for instance, all projects in the HEF group were of type "Web-based Database" but not all projects of type "Web-based Database" had an EF value greater than 2.

Another side benefit of early determination of a project's EF group is that there is potential to save valuable time, money, and effort. This is especially true for projects that belong to the LEF group. Since these projects have low EF values some steps of the Elaboration phase may be skipped and the Construction phase activities may begin earlier.

A number of techniques such as Use Case Points [Karner 1993] and Predictive Object Points [Minkiewicz 1997] have been proposed to come up with early estimates of the effort required for a software project. All of these techniques, however, are applicable only after some preliminary analysis or design of the software project at hand. Early determination of a project's EF group may enable a much earlier goals-based estimation. It should not be hard to see that there is a positive relationship between the EF value and the size of a project. Assuming everything else is constant, a project that undergoes more elaboration of goals will be of a bigger size than the one which undergoes less elaboration. Thus, accurate a priori determination of a project's EF group (which, in turn, bounds the EF value) has a direct bearing on the accuracy of early estimation of a project's cost and schedule. For instance, a project belonging to the HEF group is likely to be more risky in terms of schedule-slippage and budget-overrun vis-à-vis a project classified in the LEF group. Project managers can, therefore, take this into consideration during the Inception phase and reflect it in their estimates of the project's cost and schedule.

6. Future Work

While this empirical study has developed the basic framework for the quantitative analysis of the phenomenon of requirements elaboration the study is by no means complete. For one, we have restricted ourselves to examining the relationship between only the capability (or functional) goals and the capability requirements. The relationship between the level-of-service (or non-functional) goals and level-of-service requirements still needs to be examined. Due to the stark contrast between the nature of capability and level-of-service requirements we suspect this relationship to be very different.

Other relationships worth looking into are between the metrics already examined (e.g. number of capability goals, number of capability requirements, etc.) and the metrics provided by the architectural documents such as the System and Software Architecture Description (SSAD) document [Boehm et al. 2005]. These new metrics include, but are not limited to, the number of actors, the number of use cases, the number of sequence diagrams, and the number of classes. We are in the process of gathering data for this purpose.

Moreover, we intend to investigate the relationship between our current metrics and software functional size metrics (such as those defined in [IFPUG 2000] and [COSMIC 2003]) collected from these projects. Also pending is our analysis of industrial data. Efforts are underway to obtain comparable data of commercial projects.

Among other things, the sheer difference between the magnitude and duration of industrial and academic projects may result in further valuable insights into the phenomenon of requirements elaboration.

References

Antón, A. I. (1996). "Goal-Based Requirements Analysis", *Proc. of the IEEE Int. Req. Eng. Conf. (RE)*, pages 136–144.

Boehm, B. (1996). "Anchoring the Software Process", *IEEE Software 13(4)*, pages 73–82.

Boehm, B., Abts, C., Brown, A., Chulani, S., Clark, B, Horowitz, E., Madachy, R., Reifer, D., and Steece, B. (2000), Software Cost Estimation with COCOMO II, Prentice Hall.

Boehm, B., Klappholz, D., Colbert, E., et al. (2005). "Guidelines for Lean Model-Based (System) Architecting and Software Engineering (LeanMBASE)", Center for Software Engineering, University of Southern California.

Cockburn, A. (2001), Writing Effective Use Cases, Addison-Wesley.

COSMIC (2003). COSMIC measurement manual version 2.2, Common Software Measurement International Consortium.

Freiman, F.R. and Park, R. E. (1979). "PRICE Software Model–Version 3: An Overview", *Proc. IEEE-PINY Workshop on Quantitative Software Models*, pages 32-41.

IFPUG (2000). Function point counting practices manual version 4.1.1, International Function Point Users Group.

Karner, G. (1993). "Resource Estimation for Objectory Projects". Objectory Systems.

Kruchten, P. (2003), The Rational Unified Process: An Introduction, Addison-Wesley.

Letier, E. and Lamsweerde, A. van (2002). "Agent-Based Tactics for Goal-Oriented Requirements Elaboration", *Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE)*, pages 83–93.

Minkiewicz, A. (1997). "Measuring Object-Oriented Software with Predictive Object Points", *Applications in Software Measurement (ASM'97)*.

Putnam, L. H. (1978). "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", *IEEE Trans. Software Engr.*, pages 345–361.

SE I (2008). Links to websites of all past semesters of Software Engineering I (CSCI 577A) course at USC, http://sunset.usc.edu/csse/courseroot/course_list.html#577a

SE II (2008). Links to websites of all past semesters of Software Engineering II (CSCI 577B) course at USC, http://sunset.usc.edu/csse/courseroot/course_list.html#577b