Language Extended Lexicon Points:

Estimating the Size of an Application using Its Language

Leandro Antonelli
Lifia, Fac. de Informática
UNLP, Bs As, Argentina
lanto@lifia.info.unlp.edu.ar

Gustavo Rossi
Lifia, Fac. de Informática
UNLP, Bs As, Argentina
gustavo@lifia.info.unlp.edu.ar

Julio Cesar Sampaio do Prado Leite
Dep. Informática
PUC-Rio, Gávea, RJ, Brasil
julio@inf.puc-rio.br

Alejandro Oliveros
INTEC
UADE, Bs As, Argentina
oliveros@gmail.com

***Abstract*—Estimating the size of a software system is a critical task due to the implications the estimation has in the management of the development project. There are some widely accepted estimation techniques: Function Points, Use Case Points and Cosmic Points, but these techniques can only be applied after the availability of a requirements specification. In this paper, we propose an approach to estimate the size of an application previous to its requirements specification by using the application language itself, captured by the Language Extended Lexicon (LEL). Our approach is based on Use Case Points and on a technique which derives Use Cases from the LEL. The proposed approach provides a measure of the application's size earlier than the usual techniques, thus reducing the effort needed to apply them. An initial experiment was conducted to evaluate the proposal.**

***Index Terms*—Requirements specifications, Domain Analysis, Language Extended Lexicon, Use Case Points, Software Sizing.**

I. Introduction

Estimating the size of a software system is a critical task since this estimation is used to plan the software system construction. Function Points [2], Cosmic Points [12] and Use Case Points [18] are three widely accepted estimation techniques. Function Points allow us to estimate the size of an application from the decomposition of the application into basic functions and files. Cosmic Points analyze functions that input, store, retrieve and output data. Use Case Points (UCP) rely on Use Cases (UC) to estimate the software size. Though these techniques have been widely used in practice, they are being adjusted continuously [17] [10] [19] [24] [12].

In order to apply any of these techniques, we need to elicit requirements to gain a deep understanding of the application, so that we can identify functions or Use Cases, then write them and afterwards perform the counts. These techniques can only be applied after requirements elicitation, analysis, and specification have been carried out. But, generally, projects are agreed upon previously to the existence of a full requirements specification.

Thus, we need an approach which can estimate the size of the software at a very early stage and obtain an objective measure, i.e., we need an approach which allows the project manager to obtain a preliminary estimation with low effort in order to use this information to sign a contract.

In this paper, we propose an approach to estimate the size of an application prior to requirements specification. We use the Language Extended Lexicon (LEL) [21] to capture the application language and we analyze it to estimate the application's size. In previous works it has been shown that it is possible to identify ontologies [9], crosscutting concerns [5] and requirements [4] from the LEL. Use Cases can be derived from the LEL [4] and Use Case Points can be applied to these Use Cases [18]. Thus, inspired by Use Case Points and considering Use Case derivation from the LEL, we set out to develop the proposed strategy. Our approach analyzes the LEL in much the same way as Use Case Points analyze Use Cases, considering the correlation between the LEL and Use Cases in terms of the derivation strategy proposed in [4]. It is important to mention that although LEL points strategy is similar to Use Case Points, our approach can be used at an earlier stage than Use Case Points, since it is applied directly to the LEL and no derivation or writing of Uses Cases is needed.

People who regularly use the LEL will benefit from this approach. Once they have built a LEL, they need to elicit requirements and analyze them, and they may capture this knowledge in the previously built LEL. Then, they can apply our approach with the refined LEL in order to obtain a measure of the application that the LEL describes. Thus, they do not need to produce a requirements specification to apply a traditional technique to estimate its size.

We propose a technique to obtain a measure of the LEL similar to the measure of Unadjusted Use Case Points (UUCP) from the Use Cases. That is, a measure of the functionality of the application. This is possible since the LEL captures the language of the application and some of the expressions captured will be verbs. These verbs are further related to Use Cases [4]. Thus, we claim that the size of the LEL (which we refer to as ULELP) could be used instead of UUCP in the estimation with Use Case Point techniques. We performed a case study to show the applicability of our approach and we carried out an experiment in order to verify its effectiveness. The rest of the paper is organized in the following way. Section 2 presents the background necessary to understand the approach. Section 3 describes the estimation approach. Section 4 provides an example. Section 5 shows a case study. Section 6 presents the evaluation process. Section 7

describes related works. Finally, section 8 brings forward some conclusions and future works.

II. BACKGROUND

This section presents the Language Extended Lexicon, the technique we use to model the application and from which we measure its size. The Use Case Point technique is also presented here, as the approach we propose is based on it.

A. Language Extended Lexicon (LEL)

LEL [21] is a glossary whose goal is to record the definition of terms that belong to a domain. It is tied to a simple idea: *understand the language of a problem, without worrying about the problem.*

Terms (symbols) are defined through two attributes: notion and behavioural responses. Notion describes the intrinsic and substantial characteristics of the symbol (denotation), while behavioral responses (connotation) describe the link between the term being described and others.

There are two principles that must be followed while describing symbols: the circularity principle (also called closure principle) and the minimal vocabulary principle. The circularity principle states that the use of LEL symbols must be maximized when describing a new symbol. The minimal vocabulary principle states that the use of words that are external to the Lexicon must be minimized. These principles are vitally important in order to obtain a self-contained and highly connected LEL. Connections among symbols determine that the LEL can be viewed as a graph.

Each symbol of the LEL belongs to one of four categories: subject, object, verb and state. This categorization guides and assists the requirements engineer with the description of attributes. Table 1 shows each category with its characteristics and how to describe them.

TABLE 1. LEL CATEGORIES.

Category	Characteristics	Notion	Behavioral responses
Subject	Active elements which perform actions	Characteristics or condition that subject satisfies	Actions that subject performs
Object	Passive elements on which subjects perform actions	Characteristics or attributes that object has	Actions that are performed on object
Verb	Actions that subjects perform on objects	Goal that verb pursues	Steps needed to complete the action
State	Situations which subjects and objects can be in	Situation represented	Actions that must be performed to change into another state

Some examples of LEL symbols are presented here. The classic bank application is used to show symbols from each category. The example consists in a bank which allows its clients to open and close accounts. If the account is activated (open) the client can deposit or withdraw money and consult the balance. The bank can also perform a cash audit.

It is important to mention that in the written descriptions we underline the terms which correspond with other defined symbols in order to show the application of the circularity principle. The following examples are: subject *client* in Figure 1; object *account* in Figure 2; verb *withdraw* in Figure 3; and state *activated* in Figure 4.

Subject: client
Notion
Person that operates an account.
Behavioral responses
The client can open an account.
The client can deposit money into his account.
The client can withdraw money from his account.
The client can consult his account balance.
The client can close an account.

Fig. 1. Client symbol description.

Object: account
Notion
The account has a balance.
Behavioral responses
The client can open an account.
The client can deposit money into his account.
The client can withdraw money from his account.
The client can consult his account balance.
The bank performs a cash audit.
The client can close an account.

Fig. 2. Account symbol description.

Verbs: withdraw
Notion
Act of taking money from the account.
Behavioral responses
The bank must check that the account has enough money to perform the withdrawal.
The bank must check that the owner of the account has not withdrawn more times than the limit allows.
The bank must check that the owner of the account doesn't have any credit card debts.
The bank reduces the balance of the account according to the amount withdrawn.

Fig. 3. Withdraw symbol description.

State: Activated
Notion
Situation where the client is ready to use an open account.
Behavioral responses
The client can close the account and he will have a closed account.

Fig. 4. Activated symbol description

.

B. Use Case Points (UCP)

The Use Case Point method is a software sizing and measurement that uses Use Case Documents and is based on [18]. This work is an adaptation of that done by Allen Albrecht on function points [2].

UCP states that the time to construct the application is affected by:
(i) The number and complexity of Use Cases.
(ii) The number and complexity of actors.
(iii) The technical requirements of the application, such as concurrency, security and performance.
(iv) Various environmental factors such as the development team's experience and knowledge.

UCP analyzes Use Case scenarios, actors and various technical and environmental factors:
(i) Unadjusted Use Case Points (UUCP). This value includes complexity of the Use Cases and the actors.
(ii) Productivity Factor (PF).
(iii) Technical Complexity Factor (TCF).
(iv) Environment Complexity Factor (ECF).

All these factors are combined in the following equation:

$$UCP = UUCP * PF * TCP * ECF$$

Fig. 5. Use Case Points equation.

1) *Unadjusted Use Case Points (UUCP).* Unadjusted Use Case Points are calculated based on two values:
(i) The Unadjusted Use Case Weight (UUCW) based on the classes, database entities, and the total number of activities (or steps) contained in all the Use Case Scenarios.
(ii) The Unadjusted Actor Weight (UAW) based on the combined complexity of all the Use Cases Actors.

2) *Unadjusted Use Case Weight (UUCW).* Individual use cases are categorized as Simple, Average or Complex, and weighed mainly depending on the classes, database entities, and the number of steps they contain. Table 2 summarizes the characteristics.

TABLE 2. USE CASES RANKS.

Use Case Type	Description	Weight
Simple	It is a simple user interface and accesses only a single database entity; its success scenario has 3 steps or less; its implementation involves less than 5 classes.	5
Average	It involves more interface design and accesses 2 database entities; between 4 and 7 steps; its implementation involves between 5 and 10 classes.	10
Complex	It involves a complex user interface or processing and accesses 3 or more database entities; over seven steps; its implementation involves more than 10 classes.	15

3) *Unadjusted Actor Weight (UAW).* Actors are classified as Simple, Average or Complex in relation to the external interface they represent in the software system. Human interacting through graphical user interface represents the Complex level, while another software system which interacts through an API is the Simple rank. Table 3 summarizes the characteristics of the actors.

TABLE 3. ACTOR RANKS.

Actor Type	Description	Weight
Simple	Another system through an API.	1
Average	Another system through a protocol. A person through a text-based user interface	2
Complex	A person through a graphical user interface	3

4) *Productivity Factor.* The Productivity Factor (PF) is a ratio of the number of man hours per Use Case Point based on past projects. If no historical data has been collected, a figure between 15 and 30 is suggested by industry experts. A typical value is 20.

5) *Technical Complexity Factors.* Thirteen standard technical factors exist to estimate the impact on productivity that various technical issues have in an application. Each factor is weighed according to its relative impact. A weight of 0 indicates that the factor is irrelevant, while value 5 means that the factor has the most impact.

6) *Environmental Complexity Factors.* Eight environmental complexity factors are defined and they must be weighed in a similar way as technical complexity factors. According to Ribu [26], environmental factors play a very important role in the estimation. A slight variation will increase the Use Case Point by a very drastic amount.

III. OUR APPROACH

The counting scheme proposed must be applied to a LEL which describes symbols of a specific application. It is important to mention that the LEL must not describe an application domain language which can give origin to several different applications; instead, the LEL must describe a specific application. Thus, before performing the counting scheme, the application domain LEL must be refined into a specific application LEL, for example by removing the verb symbols which will not be developed as functionality.

The counting scheme proposed is based on the Use Case Points estimation technique. Since Use Cases can be derived from the LEL [4], we designed a counting scheme that uses the same calculations performed in Use Case Points, but the calculations are performed in the LEL using the relationship in the information from both models. It is worth mentioning that the measure calculated with this approach is equivalent to Unadjusted Use Case Points (UUCP), which we refer to as Unadjusted LEL Points (ULELP). This measure includes Unadjusted Verbs Weight (UVW) and Unadjusted Subject Weight (USW) which are equivalent to Unadjusted Use Case Weight (UUCW) and Unadjusted Actor Weight (UAW). Our approach does not consider measures

equivalent to: (i) Productivity Factor, (ii) Technical Complexity Factors and (iii) Environmental Complexity Factors. Since ULELP is based on UUCP and considering that (i), (ii) and (iii) do not depend on functionality described by Use Cases, occasionally the UCP framework for (i), (ii) and (iii) can be used with ULELP. The rest of the section describes how to calculate USW and UVW.

A. Unadjusted Subject Weight (USW)

The measure of UAW in UCP consists in analyzing the actors who use the application. These actors are modeled as symbols of the subject category in LEL descriptions according to [4]. Thus, subject symbols must be ranked in a similar way as actors of UCP are ranked.

TABLE 4. SUBJECT RANKS.

Subject Type	Description	Weight
Simple	Another system through an API.	1
Average	Another system through a protocol. A person through a text-based user interface	2
Complex	A person through a graphical user interface	3

B. Unadjusted Verb Weight (UVW)

The measurement of UUCW in UCP consists in analyzing the following characteristics for each piece of functionality (UC): (i) user interface, (ii) access to database entity, (iii) number of steps in the success scenario and (iv) number of classes that involve its implementation. Most of this information is also present in the LEL according to [4].

Although a LEL models the language of an application, we must consider that the language corresponds to a specific application, thus we must analyze the user interface in the same way as it is analyzed in UCP.

The UC has a description of the main success scenario. In verb symbols, this description is given in their behavioural responses. Therefore, the number of steps is counted according to the number of steps in behavioural responses. Database entities and classes must be analyzed from this attribute.

Since a LEL describes the language of the application, it does not explicitly mention database entities or classes. Nevertheless, the LEL considers symbols of the object and subject categories, which are related to database entities and classes. Wirfs-Brock et al. [30] also make this connection when they relate nouns to objects. Then, in an Object Oriented application, some objects will be persisted into a database, thus, they can be considered database entities.

In general, every object symbol of the LEL can be considered a database entity in UCP. But as the LEL is very detailed, it can have descriptions of object symbols which are not commonly described in UC, and they will only appear in a detailed Entity Relationship model as attributes instead of entities. Thus, symbols of the object category and with a low level of detail must not be considered. Additionally, symbols of the subject category which are used passively must be taken into account. Although they are subjects, if they are used passively as objects they must be considered objects too.

The last element to count in UC is the number of classes that involve its implementation. In the LEL description, subject symbols are candidates to be implemented as classes because they are active elements which perform actions (this is the definition of a class). Nevertheless, we must also consider objects and count them when they perform actions too.

Finally, Table 5 describes how to rank verb symbols.

TABLE 5. VERBS RANK.

Verb Type	UI	Objects	Steps	Subjects	Weight
Simple	Simple	1	<=3	<=4	5
Average	Average	2	4-7	5-10	10
Complex	Complex	>2	>7	>10	15

IV. AN EXAMPLE

In this section we show how to apply the strategy proposed. We use an example from a small application extracted from a Human Resources Management system. We describe how to rank and measure three verb symbols in order to exemplify the strategy. Although the example is very small, it depicts the three situations that can be encountered when applying the counting strategy.

The rest of the section is organized in the following way. First, a description of the application is given. Then, the symbols of the LEL identified from the application are shown. Finally, subject and verb symbols are described and analyzed according to the strategy proposed so as to rank them.

A. The Application

The software application's goal is to manage vacation periods for the employees of a company. Employees have a certain number of vacation days per year. This number depends on how many years the employee has been working for the company. When the employee requests a vacation period, the request needs two authorizations to become effective. First of all, the Human Resources Department must verify that the employee is not requesting more days than those he is allowed to take. After that, the employee's manager must analyze the project schedule in order to determine whether or not the employee's presence is essential at work in the vacation period he has requested.

B. LEL

The symbols identified from the vacation management application are summarized in Table 6.

TABLE 6. LEL SYMBOLS OF VACATION APPLICATION.

Subjects	Objects	Verbs	States
Employee	Vacation Request	Request vacation	New
Human Resources Department	Period	Verify request	Verified
Manager		Analyze request	Analyzed
			Approved
			Rejected

C. Ranking of Elements

ULELP is calculated from USW and UVW. There are 3 subjects in the example: *employee*, *human resources department*, and *manager*. We can consider that the application will have a graphical user interface. Thus, each subject is considered complex and its weight is 3. So, there are 3 subjects with a weight of 3 each. Therefore, USW is 9.

In the following paragraphs we describe each verb symbol and also analyze steps, objects and subjects in order to rank them. In general, all the user interfaces are simple, so they are not mentioned. The first verb symbol is *request vacation*, which is described in Figure 6.

Verb: request vacation
Notion
Act of asking for permission to take some days off work
Behavioral responses
The employee defines the time period
Human Resources Department records the vacation request

Fig. 6. Request vacation symbol description.

The strategy demands to analyze three variables in order to rank the symbol: (i) number of steps, (ii) number of objects and (iii) number of subjects. All of them are related to behavioural responses. The number of steps in the behavioral responses is two, shown in each sentence. Although the number of objects in the behavioural responses is also two (*period* and *vacation request*), we must consider only one object: *vacation request*, because *period* would actually be an attribute of *vacation request*, and would therefore not be an object, so it must not be taken into account. Finally, the number of subjects is two: *employee* and *Human Resources Department*. According to these variables, the verb symbol *request vacation* must be ranked as a simple verb. Table 7 summarizes this information.

TABLE 7. RANKING OF THE REQUEST VACATION SYMBOL

Verb Type	UI	Objects	Steps	Subjects	Weight
Simple	Simple	1	2	2	5

The following verb symbol to analyze is *verify request*, which is described in the Figure 7.

The number of steps is three, because there are three sentences in the behavioral responses. The number of objects in the behavioral responses is one (vacation request), but we must count one more object. Although employee is a subject symbol, it is used as an object, so two objects must be counted. Finally, the number of subjects is two: Human Resources Department and employee. Although three variables are in simple rank, there is one variable in average rank.

Verb: verify request
Notion
Act of verifying that the number of days requested by the employee does not exceed the days allowed to him.
Behavioral responses
Human Resources Department identifies the number of days that the employee can take.
Human Resources Department calculates the number of days that the employee has already taken, if any.
Human Resources Department checks that the number of days in his vacation request do not exceed the number of remaining days allowed.

Fig. 7. Verify request symbol description.

Using the same criteria as with UCP, the verb symbol: *verify request* must be ranked as an average verb because one variable (objects) is ranked as average. Table 8 summarizes this information.

TABLE 8. RANKING OF THE VERIFY REQUEST SYMBOL

Verb Type	UI	Objects	Steps	Subjects	Weight
Simple	Simple		3	2	
Average		2			10

The last verb symbol to analyze is: *analyze request*. It is described in Figure 8. The number of steps in behavioural responses is one. Although the manager needs to do some comparison with the project schedule, the contrast he must do is beyond the scope of the application; this is the reason why the symbol *contrast* is not defined and there are no more steps in describing the constraint. The number of objects is one: *vacation request*. The *schedule tasks* are mentioned, but they are beyond the scope of the application, too. Finally, the number of subjects is one: *Manager*. According to these variables, the verb symbol *analyze request* must be ranked as a simple verb. It is important to mention that the application only needs to make it possible for the *manager* to accept or reject the *vacation request*. This functionality is very easy to implement, so ranking it as simple makes sense. Table 9 summarizes this information.

Verb: analyze request
Notion
Act of analyzing the project schedule needs for the period requested.
Behavioral responses
Manager contrasts the vacation request with the schedule tasks.

Fig. 8. Analyze request symbol description.

TABLE 9. RANKING OF THE ANALYZE REQUEST SYMBOL

Verb Type	UI	Objects	Steps	Subjects	Weight
Simple	Simple	1	1	1	5

Summing up, USW is 3+3+3, and UVW is 5+10+5. Thus, ULELP is USW + UVW, that is, 29. This value could be used instead of UUCP in UCP. Adding the Technical Complexity Factor (TCF), the Environment Complexity Factor (ECF) and the Productivity Factor (PF), an effort estimation in man hours could be reached.

V. CASE STUDY

In order to verify and show the applicability of the approach, we applied the counting strategy to a real software system. The system is an open government platform which integrates different tools: a social network, a content repository, instant messaging, voice over IP communication, and many others. Some of the tools were developed by us while others were open source software. The development began in 2010 and nowadays there are 22 team members who play different roles: leaders, architects, analysts, testers, UX designers, and developers.

We built the LEL with the knowledge gathered from being part of the team and we also consulted a document in order to validate the completeness of the LEL. The vision and scope document was used. This document describes the boundaries and features of the system, but it does not describe the requirements. It is worth mentioning that we use this document because the focus of the case study is to verify the counting strategy and not to assure completeness of the LEL. The analyst needs to elicit and analyze requirements to refine the LEL. This activity can be done either interacting with a stakeholder or using the vision and scope document as we did for this case study. Moreover, we need a LEL and Use Cases that represent the same system in order to compare their measures, thus we need to ensure that both models represent the same functionality.

We identified a total of 234 symbols. The number of symbols from each category is shown in Table 10.

All of the subjects correspond to people using the system through a graphical user interface. Thus, they are ranked as complex. Each of the 8 subjects has a weight of 3, so USW is 8 * 3 = 24. Then, verbs are ranked according to Table 11.

TABLE 10. NUMBER OF SYMBOLS FOR EACH CATEGORY OF THE SOCIAL NETWORK SYSTEM

Category	Symbols
Subjects	8
Objects	26
Verbs	170
States	30

TABLE 11. VERBS RANK.

Verb Type	Number
Simple	140
Average	23
Complex	7

Thus, UVW is 140 * 5 + 23 * 10 + 7 * 15 = 1035. Finally, the Unadjusted LEL Points (ULELP) is USW + UVW = 1059. This case study allows us to verify the applicability of the approach and the proximity between the measure of ULELP and UUCP. The strategy to measure USV is quite simple and similar to UAW in UCP. We identified 8 subject symbols which match the 8 actors identified in the Use Case analysis. Thus, considering that the application has a graphical user interface, USW was 24 (equal to the UAW).

The weight of verbs is different from the weight of Use Cases. The main difference arises from the functionality related with workflows. Each transition of the workflow was modeled as an individual verb symbol, while one Use Case describes functionality related to several steps in the workflows. Thus, 12 verbs were ranked as simple while the same functionality was represented by 4 Use Cases ranked as average. There were also 10 verb symbols ranked as simple while their related functionality was modeled into 2 Use Cases ranked as complex. This was a remarkable difference in the definition of the LEL and the Use Cases. The difference had an impact on the measurement, but it was small. The weight related to these symbols was 110, while the weight related to the Use Cases was 70.

There were other verb symbols which were ranked as a different category from their related Use Cases. These 11 verb symbols were ranked as simple while their related Use Cases were ranked as complex. The reason for the different ranks lies in the description of the symbols, as they were not fully described; that is, their related Use Cases have more information than the LEL symbols.

Table 12 summarizes the Use Case ranks.

TABLE 12. USE CASES RANK.

Use Case Type	Number
Simple	107
Average	28
Complex	19

Thus, UUCW is 107 * 5 + 28 * 10 + 19 * 15 = 1100. Finally, the Unadjusted Use Case Points value (UUCP) is UAW + UUCW = 1124.

In summary, the ULELP and UUCP measures were similar. In fact, they were quite close: 1059 and 1100 respectively. The analysis of UAW and USW is similar in both approaches, so it is not necessary to perform any comparison between them. The analysis of UUCW and UVW depends on the construction of each model. The description of functionality related to workflows is modeled in a different way in the LEL and UC. Although the weight of both models was quite close too, we must continue to consider and study this situation.

VI. Evaluation

An experiment was performed in order to assess the precision of the strategy proposed. Since our approach is based on Use Case Points, the experiment consisted in comparing our strategy with Use Case Points. The previous case study showed that the difference between Use Case Points and LEL Points lies in the Unadjusted Use Case Weight and the Unadjusted Verb Weight. Unadjusted Actor Weight and Unadjusted Subject Weight were equal in both strategies, thus we focused on contrasting UUCW with UVW in the experiment. There are four attributes that define UUCW and UVW: (i) user interface (UI), (ii) database entities / objects, (iii) steps, and (iv) classes / subjects. Since the analysis of UI is similar in both approaches, we did not consider this attribute in the experiment. We decided to design an experiment where we provided the participants with a description of Use Cases and LEL, so we could measure the subjectivity in the application of the strategy with the same model. It is irrelevant to compare steps between Use Cases and LEL, because this measure is objective. We were particularly interested in comparing how subjective the identification of the other two variables is in the two strategies. The goal of the experiment is described according to the Goal/Question/Metric (GQM) method formulated by Basili et al. [7]:

Analyze Unadjusted Use Case Weight and Unadjusted Verb Weight for the purpose of *contrasting the variables "database entities vs object" and "Classes vs Subjects"*
with respect to *precision*

The participants of the experiment were 18 students from a Computer Science postgraduate course. The experiment was part of the course activities. All the students had a degree in Computer Science and experience in the software development industry, some of them as developers, others as analysts and some others as team leaders. Some participants were also teachers at the University. The majority of the participants were Argentinean, but there were also people from Colombia.

The materials used consisted of a slide show presentation, a basic guide about how to apply Use Case Points and LEL Points, and a description of the application, which was an issue tracker system. Three different materials were produced: (i) a colloquial description, (ii) Use Cases and (iii) a LEL. From the same colloquial description, 10 Use Cases and their related verb symbols were described. This functionality was: (i) create an issue, (ii) assign an issue, (iii) start, (iv) pause, (v) finish and (vi) cancel working on an issue, (vii) break down an issue, (viii) browse issues, (ix) calculate working factor for each member (in order to perform resource leveling) [25], and (x) perform resource leveling.

The experiment was carried out during a class. At the beginning of the experiment, the participants were instructed how to calculate UUCP and ULELP (all the participants were instructed in both strategies). Then, two groups were formed. A group of 9 students was asked to apply Use Case Points while another group of 9 students was asked to apply LEL points. We decided to make two groups of people and ask them to apply only one strategy each, in order to avoid bias from one technique to the other. The participants were asked to complete the counting in two hours. First, they needed to read the description of the application, because they were not given any information about it during the presentation, in which only the counting strategies were described. After that, every participant had to provide the following information for each Use Case or Verb: (i) Database entities or Objects identified and (ii) Classes or Subjects identified.

The analysis consisted in calculating the variance for each attribute in each technique for every requirement. These variances allow us to determine the precision of the techniques. The number of participants was small in order to obtain statistical results. Nevertheless, the experiment allows us to obtain general qualitative information to make adjustments and carry out the experiment again with more participants. For this reason, instead of showing the values for each variance, we categorize situations into four groups according to the difference in variance:

(A) Both variances of ULELP and UUCP are low and close to each other. Both techniques are equally good.

(B) The variance of ULELP is low; the variance of UUCP is high. ULELP is more precise than UUCP.

(C) The variance of ULELP is high; the variance of UUCP is higher. Neither approach is precise, but this high variance does not impact on the ranking of both strategies.

(D) The variance of ULELP is high; the variance of UUCP is higher. Neither approach is precise, but this high variance impacts on the ranking of both strategies.

In relation to database entities / object comparison, both techniques were equally good in 4 out of 10 requirements. Features i (create an issue) and vii (break down an issue) were more precisely ranked with ULELP than with UUCP. Features viii (browse issues), ix (calculate working factor) and x (perform resource leveling) were not precise in either technique.

TABLE 13. COMPARISON OF THE VARIANCE OF EACH VARIABLE IN USE CASE AND LEL FOR ALL REQUIREMENTS

Req	Variance of Database Entity in UUCP	Variance of Objects in ULELP	Variance of Classes in UUCP	Variance of Subjects in ULELP
i	B	B	A	A
ii	D	D	A	A
iii	A	A	A	A
iv	A	A	A	A
v	A	A	A	A
vi	A	A	A	A
vii	B	B	A	A
viii	D	D	A	A
ix	D	D	C	C
x	D	D	C	C

In relation to class / subject comparison, both techniques were equally good in 8 out of 10 requirements. Features ix (calculate working factor) and x (perform resource leveling) were not precise in either technique, but as the ranks are 0-4, 5-10 and 11-more this difference did not impact on rank definition.

In summary, the variance of ULELP was equal to or better than UUCP. We can thus claim that ULELP is more precise than UUCP. The reason for this is that participants must count objects and subjects and they are assisted in this by LEL symbols, while they do not have this help when they rank with UUCP. This experience allows us to design a new experiment with more participants in order to prove this finding statistically. Moreover, the new experiment will include the description of Use Cases and LEL by participants, so as to assess the variance in number of steps.

Feldt et al [13] state the importance of analyzing threats to the validity of the study and the results. Wohlin et al. [31] group validity threats into four categories: conclusion, internal, construct and external validity. The following paragraphs analyze different threats from each category.

Concerning conclusion category, one possible threat is random heterogeneity of subjects. The participants are heterogeneous as regards years of experience in industry, but there is homogeneity regarding overall experience. Since the application of the technique is very simple, heterogeneity of subjects does not represent any threat.

The second category of threats to analyze is internal validity. Selection is the main threat to internal validity. In order to tackle the effect of natural variation in human performance we selected people with experience in the application domain and in requirements engineering, but with no experience in the approaches we analyzed. Then, we carried out a randomization to assign subjects to treatments.

According to the construct validity category, we observed that the experiment did not suffer from such threats referred to as hypothesis guessing, evaluation apprehension or experimenter expectancies. The subjects were not familiar with the approaches, so they could not force specific results. Also, the experiment did not have to deal with the interaction of different treatments, because each subject was assigned only one treatment, so there was no bias.

Sjøberg et al [27] state that many threats to external validity are caused by an artificial setting of the experiment. Taking this into account, we set up an experiment which had the complexity of a small but real application (a real application was developed in an organization where one of the authors works).

VII. RELATED WORKS

The goal of our approach is to measure the size of an application. Niknafs [23] states that while a requirements engineer has in-depth domain knowledge that helps him to understand the problem easier, he can nevertheless fall for tacit assumptions of the domain and might overlook issues that are obvious to domain experts. Since we model the application through a glossary that must be refined from a general glossary of the domain, our approach makes it possible to tackle the problem stated by Niknafs, since one requirements engineer can write the general glossary, while another can write the specific application glossary.

Our approach is also in agreement with Glinz [14] and Waldmann [29]. Glinz proposes a lightweight requirements modeling language as an alternative to textual and pictorial specifications. Waldmann states that requirements engineering must learn from agile development. In our approach, requirements specifications can be automated from the LEL as in [8]

LEL can be considered a lightweight requirements model to be used in agile methodologies. Although a LEL is an early product, the measurement from early products (and early activities) was shown to be effective by Tsunoda in [28]. This early measurement is also important for business decisions [20], and it is crucial for small companies that do not have a defined process [6].

Abrahao [1] proposes a measurement procedure (ReqPoints) to estimate the size of object-oriented software projects from a requirements specification. Specifically, a set of measurement rules is defined as a mapping between the concepts of the Requirements Metamodel onto the concepts of the Function Point Analysis (FPA) Metamodel. Our approach is similar to Abrahao's since the elementary functions identified are similar to verb symbols. Then, they map the elements onto Function Points while we map them onto Use Case Points.

Harput et al [16] present an approach to apply FPA to an object-oriented requirements model which is specified with scenarios as well as sequence diagrams and class diagrams. This approach defines rules to interpret the object-oriented model and apply FPA. It is similar to our approach because in some way we provide a strategy to interpret the LEL as if it were UC; nevertheless our translation is simpler. This is an important distinction: we work with a simple and easy to use technique, i.e., LEL. These features are essential to foster applicability and improve the obtained results. Anda et al [3] and Ribu [26] report the difficulty of estimating from complex structures. Anda reports the results of three

industrial case studies on the application of a method based on Use Case Points. They state that the design of the use case models has a strong impact on the estimation, since the more complex the design of the product, the more difficult and more sensible the application of the estimation techniques is. Ribu agrees with these views, reporting that the main difficulty she encountered when applying the Use Case Points method was that practitioners wrote use cases in very different ways and with different levels of detail. Our approach relies on the LEL, and although different subjects can bias the descriptions, LEL has very basic rules which help to write homogeneous descriptions.

Cockcroft [11] empirically proved that it is possible to calculate the size of a data flow diagram (which is built early at software development) and this size is related to the code line of the coded application. MacDonell [22] works on measuring the specification and the relationship between specification and process effort. We work with the specification of the application, but the size we obtain can be translated to effort because there is a relationship between LEL-points and Use Case Points obtained from the Use Cases derived from a LEL. Zhao [32] calculates a measuring from an ER diagram. He developed a complexity path in a data-oriented model. Although our model is functional oriented, it also has object symbols which make a similar analysis possible. Grimstad [15] states that estimation-irrelevant information should be removed from the requirements specification prior to its use as the input to estimate work. Since we use raw material to construct Use Cases, our approach is in agreement with Grimstad's.

VIII. Conclusion

We have presented an approach to calculate the size of an application from the application language captured by the Language Extended Lexicon. Estimating the size of the application prior to requirements specification can be very difficult, and wrong measures are dangerous for project planning. With the proposed approach we focus on the language, and from there we estimate the size of the application. With this size in mind, a more realistic estimation can be performed before the existence of a requirements specification.

Our approach consists in analyzing a Language Extended Lexicon in the same way as Use Case Points analyzes Use Cases. Since there is a strategy to derive Use Cases from a Language Extended Lexicon, we have adapted Use Case Points based on Use Case derivation from LEL in order to measure a LEL. Thus, if we have a LEL, we can measure it directly and avoid the extra effort of deriving Use Cases and then calculating Use Case Points from such a derivation. We obtain a similar measure with less effort because our technique works with a previous product. Moreover, we obtain a measure earlier. Although it is necessary to elicit and analyze requirements to transform the domain LEL into a specific application LEL, we believe it is useful for requirements engineers who use LEL, as the extra effort they make adjusting the LEL is less than the effort necessary to specify Use Cases. From that LEL, Unadjusted LEL Points can be semi-automatically calculated obtaining a measure of the system previous to specifying its requirements.

The similarities between Use Case Points and LEL Points arise from the fact that the design of LEL Points is based on Use Case Points. In this paper we described an experiment which showed that the measures obtained using LEL Points have an equal or smaller variance than Use Case Points, thus people who do not use LEL regularly, can begin using it in order to get benefits from the better precision our technique has. In order to further sustain this initial result, we are working on more evaluations based on different case studies. Particularly, we are working on obtaining measures from enough participants in order to perform a statistical analysis. Moreover, we will work in analyzing relationships such as "is a" between symbols and their impacts on measurement. We are also working in tool support to partial automate the counting process.

References

[1] S. Abrahao, and E. Insfran, "A metamodeling approach to estimate software size from requirements specifications", in Proceeding of the Software Engineering and Advanced Applications (SEAA'08), ISBN 978-0-7695-3276-9, 3-5 Sept, Parma, 2008, pp 465 – 475.

[2] A. J. Albrecht, "Measuring application development productivity", in Proc. IBM Applications Development Symp., GUIDE Int. and SHARE Inc., IBM Corp., Monterey, CA, 1979, p. 83.

[3] B. Anda, H. Dreiem, D. I. K. Sjøberg, and M. Jørgensen, "Estimating software development effort based on Use Cases – Experiences from industry", in Proc. of the 4th Int. Conference on the Unified Modeling Language, Springer Verlag, Toronto, Canada, 2001, pp. 487-502.

[4] L. Antonelli, G. Rossi, J.C.S.P. Leite, and A. Oliveros, "Deriving requirements specifications from the application domain language captured by Language Extended Lexicon", in proceedings of the Workshop in Requirements Engineering (WER), Buenos Aires, Argentina, April 2012, pp 24 – 27.

[5] L. Antonelli, G. Rossi, J.C.S.P. Leite, and J. Araújo, "Early identification of crosscutting concerns with the Language Extended Lexicon", Requirements Engineering Journal, http://dx.doi.org/10.1007/s00766-013-0193-4, Springer London, 2013, pp 1-23.

[6] J. Aranda, S. Easterbrook, and G. Wilson, "Requirements in the wild: how small companies do it", in Proc of the 15th IEEE International Requirements Engineering Conference (RE '07), ISBN 978-0-7695-2935-6, 10.1109/RE.2007.54, 15-19 Oct. 2007, Delhi, 2007, pp 39 – 48.

[7] V.R. Basili, G. Caldiera, and H.D. Rombach, "The goal question metric approach", in Encyclopedia of Software Engineering, John Wiley & Sons, Vol. 1, 1994, pp.528-532.

[8] E. Boutkova and F. Houdek, "Semi-automatic identification of features in requirement specifications", in Proceedings of the 19th IEEE International Requirements Engineering

Conference (RE), ISBN 978-1-4577-0921-0, 10.1109/RE.2011.6051626, Aug. 29 2011-Sept. 2, Trento, 2011, pp 313 - 318.

[9] K.K. Breitmanm and J.C.S.P. Leite, "Ontology as a requirements engineering product", in Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE), IEEE Computer Society, Monterey Bay, California, USA, ISBN 0-7695-1980-6, 2003.

[10] E.R. Carroll, "Estimating software based on use case points", Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, San Diego, CA, USA, October 16-20 2005.

[11] S. Cockcroft, "Estimating CASE development size from outline specifications", Information and Software Technology 38, 1996, pp. 391-399.

[12] Cosmic Common Software Measurement International Consortium, Cosmic Functional Size Measurement. Available at: http://www.cosmicon.com/.

[13] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research - An initial survey", in proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE), July 1-3, San Francisco Bay, 2010, pp 374-379.

[14] M. Glinz, "Very lightweight requirements modeling", in Proceeding of the 18th IEEE International Requirements Engineering Conference (RE), ISBN 978-1-4244-8022-7, 10.1109/RE.2010.73, Sept. 27 2010-Oct. 1, Sydney, NSW, 2010, pp 385 – 386.

[15] S. Grimstad and M. Jorgensen, "The impact of irrelevant information on estimates of software development effort", in Proceedings of the 18th Australian Software Engineering Conference (ASWEC 2007), ISBN 0-7695-2778-7, 10.1109/ASWEC.2007.48, 10-13 April, Melbourne, Vic., 2007, pp 359 – 368.

[16] V. Harput, H. Kaindl, and S. Kramer, "Extending Function Point analysis to Object-Oriented requirements specifications", in Proceeding of the 11th IEEE International Symposium of Software Metrics, September 2005.

[17] International Function Points User Groups, available at: http://www.ifpug.org/.

[18] G. Karner, "Metrics for objectory", Master Thesis, Linkoping University (LiTH-IDA-. Ex-9344:21), Linkoping, Sweden, 1993.

[19] S. Kusumoto, M. Fumikazu, H. Shigeo, and M. Yuusuke, "Estimating effort by Use Case Points: method, tool and sase study", Proceedings of the 10th International Symposium on Software Metrics, 2004.

[20] L. Lehtola, M. Kauppinen, and J. Vähäniitty, "Strengthening the link between business decisions and RE: Long-term product planning in software product companies", in Proceedings of the 15th IEEE International Requirements Engineering Conference (RE '07), ISBN 978-0-7695-2935-6, 10.1109/RE.2007.30, 15-19 Oct., Delhi, 2007, pp 153 – 162.

[21] J.C.S.P. Leite and A.P.M. Franco, "A strategy for conceptual model acquisition", in Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, California, IEEE Computer Society Press, 1993, pp 243-246.

[22] S.G. MacDonell, "Establishing relationships between specification size and software process effort in CASE environments", Information and Software Technology, 39 - 1, 1997, pp 35-45.

[23] A. Niknafs and D.M. Berry, "The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation", in Proceeding of the 20th IEEE International Requirements Engineering Conference (RE), 10.1109/RE.2012.6345802, ISBN 978-1-4673-2783-1, 24-28 Sept, Chicago, IL, 2012, pp181 – 190.

[24] J. Ouwerkerk and A. Abran, "Evaluation of the design of Use Case Points (UCP)", MENSURA2006, Shaker Verlag, 4-5 November, Cadiz, Spain, 2006.

[25] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK Guide), Fifth Edition, ISBN 9781935589679, 2013.

[26] K. Ribu, "Estimating Object-Oriented software projects with Use Cases", Master of Science Thesis, University of Oslo Department of Informatics, 7th November 2001.

[27] D.I.K. Sjøberg, B. Anda, E. Arisholm, T. Dybå, M. Jørgensen, A. Karahasanovic, E.F. Koren, and M. Vokác, "Conducting realistic experiments in software engineering", in Proceedings of the International Symposium on Empirical Software Engineering (ISESE), ISBN 0-7695-1796-X, 2002, pp 17.

[28] M. Tsunoda, Y. Kamei, K. Toda, and M. Nagappan, "Revisiting software development effort estimation based on early phase development activities", in Proceeding of the 10th IEEE Working Conference on Mining Software Repositories (MSR), ISBN 978-1-4799-0345-0, 10.1109/MSR.2013.6624059, 18-19 May, San Francisco, CA, 2013, pp 429 - 438.

[29] B. Waldmann and A.G. Phonak, "There's never enough time: Doing requirements under resource constraints, and what requirements engineering can learn from agile development", in Proceedings of the 19th IEEE International Requirements Engineering Conference (RE), ISBN 978-1-4577-0921-0, 10.1109/RE.2011.6051626, Aug. 29 2011-Sept. 2, Trento, 2011, pp 301 – 305.

[30] R. Wirfs-Brock, B. Wilkerson, and L. Wiener, "Designing Object-Oriented software", Prentice Hall, 1990.

[31] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, "Experimentation in software engineering an introduction", ISBN 0-7923-8682-5 Academic Publishers 2000.

[32] Y. Zhao, H.B. Kuan Tanm, and W. Zhang, "Software cost estimation through conceptual requirement", in Proceedings of the Third International Conference on Quality Software, ISBN 0-7695-2015-4, 10.1109/QSIC.2003.1319096, 6-7 Nov. 2003, pp141 - 144.