# An Approach to Determine Software Requirement Construction Sequences based on Use Cases

José Antonio Pow-Sang[1,3], Arturo Nakasone[2], Ricardo Imbert[3], and Ana María Moreno[3]
[1] *Departamento de Ingeniería, Pontificia Universidad Católica del Perú, Peru*
[2] *National Institute of Informatics, Tokyo, Japan*
[3] *Facultad de Informática, Universidad Politécnica de Madrid, Spain*
*{japowsang, arturo.nakasone}@pucp.edu.pe, {rimbert, ammoreno}@fi.upm.es*

## Abstract

*The majority of software development projects utilize the use cases technique to define software requirements, which are necessary to determine not only the scope of the software itself, but also the sequence in which this software will be constructed. Currently, there are several proposals to define the construction sequence of software requirements, but most of these proposals lack of ease of use from the developer's perspective.*

*This paper presents an approach to determine software construction sequences based on use cases precedence diagrams, which offers some advantages from the developer's point of view. In order to demonstrate the technique, we also present a controlled experiment performed by people who had at least two years of experience in software projects. The results of this experiment show that our proposed technique, unlike other ad-hoc techniques used, enables a more precise definition of the construction sequence.*

## 1. Introduction

The utilization of use cases in software engineering processes was first proposed by Ivar Jacobson [7], and, since its inclusion in the UML standard specification [13], its use has been greatly extended, making it a mandatory requirement for any object oriented software development project.

As an established practical fact, it is quite known that the data tables for any information system are classified into one of these two types: Master Table, if the table contains data which seldom changes (e.g. customer information), and Transaction Table, if the table contains data which is frequently modified (e.g. the sales order for a customer).

Based on this table classification, we identified three types of use cases: (1) use cases that deal with master table maintenance, (2) use cases that deal with transaction table maintenance, and (3) use cases that deal with data reporting.

Even though there are several approaches to determine software construction sequences that consider which requirement must be constructed first based on a simple requirements prioritization scheme, most of them do not take into consideration the developer's perspective in terms of ease of construction to define such priorities. As an example, we applied a survey among sixty undergraduate students at our university and we obtained the following requirements sequence based on their construction easiness: use cases that maintain master tables, use cases that maintain transaction tables and, finally, use cases that present reports. In spite of the fact that these poll results cannot be applied to every software project, taking into consideration the valuable opinion of developers during the analysis phase can improve considerably the project's chance of success in the end.

Our paper presents a novel technique to determine software construction sequences based on use cases precedence diagrams and the developer's point of view regarding requirements prioritization based on their ease of construction.

The rest of the paper is organized as follows: Section 2 describes the related work in the area, Section 3 details our proposed technique to define software requirements construction sequences, Section 4 presents the background scenario for the empirical study; Section 5 shows the obtained results for the experimental study; Section 6 discusses those results.

Finally, a summary and our plans for future research will conclude our paper.

## 2. Related Work

In order to deal with the problem of construction sequence definition, there are several proposals that establish guidelines to prioritize software requirement construction by considering criteria that do not take into consideration the easiness from the developer's point of view. Denne et Al. [3] propose a method called "Incremental Funding Method", which defines construction increments based on financial decisions. Evolve, a technique proposed by Rühe et Al. [4][5][12][17], determines the construction sequence by using questionnaires among the stakeholders. Firesmith [6] presents general guidelines in which different perspectives from stakeholders are included, but without defining a specific technique. All these proposals, excepting the one made by Firesmith, do not utilize use cases at all and only make mention of the concept of requirements.

Moisiadis [11] proposes the realization of a use case prioritization only considering their dependencies based on existing pre and post conditions and the stakeholders' opinions. Moisiadis [10] and Kundu [9] proposed the elaboration of dependency diagrams for use cases scenarios in order to establish their priority. Some [20] utilizes activity diagrams to show the dependency between use cases. Nevertheless, none of these techniques indicates how they can be used to define construction sequences. Ryser presents the SCENT-Method technique [18] which makes use of diagrams that show all the different types of dependencies between use cases to generate test cases, although it does not properly define a technique for sequence determination.

In summary, the techniques that mainly consider the stakeholders' perspective do not include the construction of diagrams for dependency between requirements. And those ones that do consider them do not deliver guidelines on how they can be utilized to determine construction sequences. Moreover, some of them include too many relation dependency types which make the diagrams overly complex to determine an appropriate construction sequence.

## 3. Use Case Precedence Diagrams and the Construction Sequence

According to UML, the relations that can exist between use cases are: include, extend, and generalization. In addition to the standard, we propose the inclusion of a new relation: precedence; and all these relations will be shown in the use case precedence diagram (UCPD). The concept of this diagram was taken from Doug Rosenberg [16], who proposed the use of a similar diagram, specifying the relations "precedes" and "invoke" to determine user requirements. Figure 1 shows an example of a UCPD.

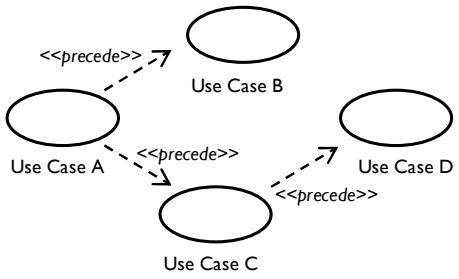


**Figure 1. Use Case Precedence Diagram**

In order to obtain precedence relations between use cases, the following rules must be considered:

*Rule 1:* A use case U1 precedes another use case U2 if there is a precondition that corresponds to the execution of a scenario in U1 that must be fulfilled before executing a scenario of U2.

For instance, to execute a scenario from the "Make Reservation" use case, the actor must have been validated by the system (i.e. execute a login). Hence, the "Login" use case precedes the "Make Reservation" use case. This is shown in Figure 2. Precedence Rule 1 - Diagram Example

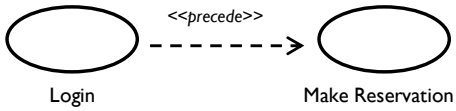


**Figure 2. Precedence Rule 1 - Diagram Example**

*Rule 2:* A use case U1 precedes another use case U2 if a U2 needs information that is registered by U1. For instance, to perform the payment for a reservation, this reservation must have been made. Having two use cases "Make Reservation" and "Pay Reservation", the former precedes the later. This is shown in Figure 3.

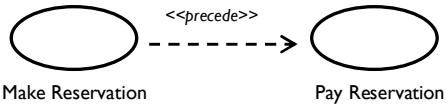


**Figure 3. . Precedence Rule 2 - Diagram example**

It is important to note that in the UCPD, Included and Extended Use Cases have not been considered

since they can be part of other use cases that refer to them. Based on this UCPD, a construction sequence is defined. The use cases that are on the left side of the diagram will be implemented before the ones that are on the right side. For instance, in Figure 1, "Use case A" will be implemented before "Use case C".

## 4. Experimental Design

The UCPD has been utilized in projects with students in a class to define construction increments, giving very good results. In those projects, students made use of the Rational Unified Process methodology [15] (the details of this experience are documented in [14]). Even though the advantages of using students in experiments are well known [2], our controlled experiment was performed with practitioners to determine the viability of our approach in real world projects.

For the experiment design, we considered the experimental software engineering suggestions made by Juristo & Moreno [8]. The goal of the experiment was to empirically corroborate if our approach provides more accurate results than informal techniques to determine the sequence use case construction.

Using the Goal/Question/Metric (GQM) template for goal-oriented software measurement [1], we defined this experiment as follows:

**Analyze**: Ad-hoc construction planning versus precedence diagram based construction planning
**For the purpose of**: Compare
**With respect to:** their accuracy
**From the point of view of:** the researcher
**In the context of:** practitioners with at least 2 years of experience in software development projects and considering that the developer is free to select the sequence to construct use cases (there are no user's constraints).

The research question was: Does our approach provide more accurate results than informal approaches when determining the sequence to construct use cases?

### 4.1. Variables Selection

The independent variable was the method used by subjects to define the construction sequence of use cases. The dependent variable was accuracy: the agreement between the measurement results and the true value.

For this experiment, it was considered as "true value" the fact that the easiest construction sequence for use cases is "master-transaction-reports", as identified in the introduction of this paper.

### 4.2. Subjects

In this experiment, we had 34 practitioners with at least 2 years in software development projects as participants. Many of the practitioners work in companies that are improving their software processes in order to obtain CMMi level 3 [19] or to obtain this certification.

Before the experiment session, we delivered a use case training class in order to standardize the knowledge of all participants. The training class was delivered four days before the experiment day.

### 4.3. Materials and Case Studies

The materials used in the experiment were two case studies and four questionnaires. The first case study corresponds to the elaboration of a sales system for a restaurant, and the second corresponds to the elaboration of an enrollment registration system for a high school. Both case studies are information systems.

For each case study, the following documentation was delivered: the use case diagram, the description for each use case along with its preconditions, and the information, in terms of classes or entities, that is needed by each use case.

Three questionnaires (1, 2 & 3) corresponds to questions in which subjects have to decide between two use cases and answer which one they would construct first. For instance, for the first case study, one of the questions included was: *Would you construct "Maintain request notes" before "Register Sale"?* The subject had to choose among these alternatives: *a) Yes, b) No, c) Indifferent*.

There are also other types of questions that allow the selection among the following use case type pairs: master-transaction, transaction-transaction, master-reports, and transaction-reports.

The fourth questionnaire was to know a practitioner's opinion regarding the easiness and usefulness of our technique, and the easiest way to construct software between master, transaction and report use cases.

In addition, in questionnaires 1, 2, and 3, we included questions regarding the sequence of tests, which were not included in this paper.

Further details of the case studies and used instruments can be found at:
http://macareo.pucp.edu.pe/japowsang/precedence/usecase.html

### 4.4. Tasks Performed during the Experiment

The practitioners had to apply the first case study with their ad-hoc techniques and the second one with

our technique. We applied two different case studies with similar characteristics (both are information systems) in order to mitigate the learning effects. Table 1 shows the tasks carried out in the session by the practitioners.

**Table 1. Tasks carried out by the practitioners**

| Task Nº | Description |
|---|---|
| 1 | Receive case study 1 and questionnaire 1 |
| 2 | Fill in questionnaire 1 |
| 3 | Receive case study 2 and questionnaire 2 |
| 4 | Elaborate use case precedence diagram for case study 2 |
| 5 | Fill in questionnaire 2 |
| 6 | Elaborate use case precedence diagram for case study 1 |
| 7 | Fill in questionnaire 3 |
| 8 | Fill in questionnaire 4 |

The session lasted approximately one hour and the practitioners performed 45 minutes on average to complete all the tasks. Even though it was not part of this study to know which technique demanded less time, we could observe that they spent less than 10 minutes to elaborate our proposed UCPD.

## 5. Results

We included the question "Select the sequence to construct use cases that you normally follow if you do not have user's constraints: maintain master tables, maintain transaction tables and reports" in questionnaire 4.

For the next results, we did not considered the questionnaires of subjects that answered a sequence different from *master-transaction-report* (8 participants) and one participant that did not understand how to elaborate the use case precedence diagram. We could observe that for some people who selected a different sequence, their answered options in questionnaires 2 and 3 were in disagreement with the UCPD that they themselves elaborated.

### 5.1 Comparison between Ad-hoc and UCPD

Table 2 presents the results obtained from the case studies. We established a significance level of 0.05 to statistically test the obtained results. In order to compare these results, we have tested the percentage of correct answers.

**Table 2. Descriptive statistics - Sequence of use case construction**

| Variable | Ad-hoc (Case Study 1) | UCPD (Case Study 2) |
|---|---|---|
| Observations | 25 | 25 |
| Minimum | 0.0% | 40.0% |
| Maximum | 100.00% | 100.00% |
| Mean | 77.33% | 90.40% |
| Std. Deviation | 23.014 | 17.436 |

In order to determine if the grades obtained with each technique followed a normal distribution, we applied the Shapiro-Wilk test for Ad-hoc (case study 1) and UCPD (case study 2). Since the computed p-value was lower than the significance level alpha=0.05 for both cases, we rejected the normal distribution hypothesis for all techniques and accepted the non-normal distribution hypothesis. Due to these results, we could not use the paired sample t-test and we had to select a non-parametric alternative. The Wilkoxon signed rank test was chosen for this purpose.

The statistical hypotheses formulated to test both techniques were:

- H0: The distributions of the ad-hoc and UCPD-Case Study 2 are not significantly different.
- Ha: The distribution of the ad-hoc sample is shifted to the left of the distribution of the UCPD-Case Study 2.

**Table 3. Wilkoxon signed rank test results ad-hoc vs. UCPD-Case Study 2**

| Variable | Result |
|---|---|
| V | 76.000 |
| Expected value | 157.500 |
| Variante (V) | 1356.125 |
| p-value (one-tailed) | 0.014 |
| Alpha | 0.05 |

Since the computed p-value was lower than the significance level alpha=0.05, we rejected the null hypothesis H0 and accepted the alternative hypothesis Ha. It means that we can empirically corroborate that our proposal produces more accurate assessments than ad-hoc techniques.

### 5.2. Qualitative Results

In questionnaire 3, we included three questions about the precedence diagram. The practitioners had to evaluate the following criteria from 0 (less) to 4 (more): ease to elaborate the UCPD, usefulness of UCPD to determine the construction sequence and if he or she would utilize UCPD in his or her own

software development projects. The results of these questions are presented in Figure 4.

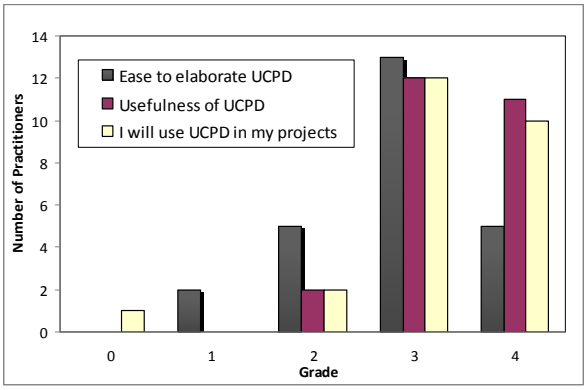


**Figure 4. Results of the questionnaire**

From Figure 4, it can be noted that in the questions most of the practitioners thought positively about the UCPD technique. Finally, as it can be observed from the figure, the questionnaire results do not disagree with the results obtained when practitioners applied the UCPD.

## 6. Discussion

In this section, we discuss various threats to validity of the empirical study and the way we attempted to alleviate them.

### 6.1 Threats to Construct Validity

Our questionnaires allowed us to quantitatively measure the precision of the ad-hoc and UCPD techniques through correct answer percentage calculations. In addition, our questions considered the construction selection among a fixed set of use case type pairs: master-transaction, transaction-transaction, master-reports, and transaction-reports. This allowed us to compare the obtained results for both case studies.

### 6.2 Threats to Internal Validity

We could conclude that an empirical evidence of the existing relationship between the independent and the dependent variables exists. We have tackled different aspects that could threaten the internal validity of the study.

*Differences among subjects*. The subjects had different background and some of them did not utilize use cases at work, but everybody knew about requirements management. To minimize this problem, we performed a training session regarding use cases previous to the execution of the experiment.

*Learning effects*. The application of two different case studies cancelled the learning effect due to similarities.

*Knowledge of the universe of discourse*. We used the same case studies (with the same type of information system) for all subjects.

*Fatigue effects*. Each student took one hour on average per session to apply both case studies and answer questionnaires. So, fatigue was not relevant.

*Persistence effects*. The students had never done a similar experiment before.

*Subject motivation*. The practitioners were motivated because they wanted to know about new techniques to improve their work. Some of them are working in companies that are preparing to obtain CMMi level 3.

### 6.3 Threats to External Validity

One threat to external validity was identified which limited the ability to apply any such generalization: the materials. In the experiment, we tried to utilize case studies which can be good representations of real life cases. Although the subjects came from a business environment, more empirical studies with real life cases from software companies must be performed.

Another aspect that could limit generalization is the fact that some practitioners considered that the easiest construction sequence from the developer's perspective was different from the master-transaction-report sequence. From the point of view of these practitioners, use cases regarding master maintenance are not important, and the technique could be useful if only use cases regarding transactions and reports were considered.

## 7. Conclusions and Future Work

This paper presents our approach to determine software requirement construction sequences based on use case precedence diagrams from the developer's perspective. This technique has been used previously in projects with students that used the Rational Unified Process model to determine the use cases to be constructed for each stage of the development process, with very good results and good comments from the students themselves.

We also included a controlled experiment in which UCPD is applied in case studies by practitioners and the obtained results show that our approach has more significant advantages over the utilization of ad-hoc techniques. And, even though polls among students

indicated that the easiest way to construct use cases was through the master-transaction-reports sequence, for some practitioners, that was not the case. As a future work, we plan to adapt our technique, so it will be able to consider other developers' and other stakeholders' perspectives.

## Acknowledgments

We are indebted to Natalia Juristo from Technical University of Madrid, for her valuable comments for the preparation of the experiment included in this paper.

This research work has been performed with the support of Dirección Académica de Investigación of Pontificia Universidad Católica del Perú, under projects DAI-E039 and DAI-4051.

## References

[1] Basili, V. R., Caldiera, G. and Rombach, H. D., "Goal Question Metric Paradigm", Encyclopedia of Software Engineering, ed. J. J. Marciniak, Wiley 1994.

[2] Carver, J., Jaccheri, L., Morasca, S., Issues in Using Students in Empirical Studies in Software Engineering Education, Proceedings METRICS'03, IEEE Computer Society, USA, 2003.

[3] Denne, M., Cleland-Huang, J., "The Incremental Funding Method: Data-Driven Software Development", IEEE Software vol 21, Number 1, IEEE Computer Society, 2004.

[4] Du G., McElroy, J., Ruhe, G., "Ad Hoc Versus Systematic Planning of Software Releases - A Three-Staged Experiment", Proceedings PROFES 2006, Lecture Notes in Computer Science, Springer, Germany, 2006.

[5] Du G., McElroy, J., Ruhe, G., "A family of empirical studies to compare informal and optimization-based planning of software releases", Proceedings ISESE 2006, IEEE Computer Society, USA, 2006.

[6] Firesmith, D., "Prioritizing Requirements", Journal of Object Technology, vol. 3, no. 8, http://www.jot.fm, September-October 2004.

[7] Jacobson, I., Object-Oriented Software Engineering. A Use Case Driven Approach, Addison-Wesley, USA, 1992.

[8] Juristo, N., Moreno, A.M., Basics of Software Engineering Experimentation, Kluwer Academic Publishers, Boston, 2001.

[9] Kundu, D., Samanta, D., "A Novel Approach of Prioritizing Use Case Scenarios", Proceedings APSEC 2007, IEEE Computer Society, USA, 2007.

[10] Moisiadis, F., "Prioritizing Scenario Evolution", Proceedings ICRE 2000, IEEE Computer Society, USA, 2000.

[11] Moisiadis, F., "Prioritising Use Cases and Scenarios", Proceedings TOOLS'37-2000, IEEE Computer Society, USA, 2000.

[12] Ngo-The, A., Ruhe, G., "A systematic approach for solving the wicked problem of software release planning", Soft Computing, Vol 12, Springer, Germany, 2008.

[13] Object Management Group, OMG Unified Modeling Language, http://www.uml.org, USA, 2008.

[14] Pow-Sang, J.A., Jolay-Vasquez, E., "An Approach of a Technique for Effort Estimation of Iterations in Software Projects", Proceedings APSEC 2006, IEEE Computer Society, USA, 2006.

[15] Rational Software, Rational Unified Process version 2001A.04.00.13, USA, 2001

[16] Rosenberg, D., Scott, K., Use Case Driven Object Modeling with UML, Addison-Wesley, Massachusets, USA, 1999.

[17] Ruhe, G., Omolade, M., "The Art and Science of Software Release Planning", IEEE Software, Vol 22, Number 1, USA, 2005.

[18] Ryser, J., Glinz, M., "Using Dependency Charts to Improve Scenario-Based Testing", Proceedings 17th International Conference on Testing Computer Software TCS'2000, USA, 2000.

[19] Software Engineering Institute (SEI), Capability Maturity Model® Integration, http://www.sei.cmu.edu/cmmi/.

[20] Some, S., "Specifying Use Case Sequencing Constraints using Description Elements", Proceedings Sixth International Workshop on Scenarios and State Machines (SCESM'07), IEEE Computer Society, USA, 2007.