Metrics Development for UML Tools evaluation

A. Dasso, A. Funes, M. Peralta, C. Salgado

{arisdas, afunes, mperalta, csalgado}@unsl.edu.ar

SEG
Universidad Nacional de San Luis
Ejército de los Andes 950
D5700HHW San Luis
Argentina
http://sel.unsl.edu.ar/
Tel.: +54 (0) 2652 42 4027 ext. 251
Fax: +54 (0) 2652 43 0224

Abstract. The Unified Modelling Language (UML) has become a defacto standard for software development practitioners. There are several tools that help the use of UML. Users of those tools must evaluate and compare different versions of the tools they intend to use or are using to assess the possibility of changing or acquiring one. There are several ways to perform this evaluation from the simple rule-of-thumb to numeric or quantitative methods. We present an ongoing project that evaluates UML tools using the Logic Scoring of Preference (LSP) method. This method is very briefly presented and also some of the ongoing work in building a model directed to UML tool evaluation is explained.

Introduction

The Unified Modelling Language (UML) [BRJ98], [OMG98] has lately become a defacto standard for software developers, whether for documenting an existing system, when doing reverse engineering on legacy systems or for development purposes from the early stages of development up to and including coding.

There are several tools that help the use of UML in any of its different applications. Users of those tools must evaluate and compare different versions of the tools they intend to use or are using to assess the possibility of changing or acquiring one. There are several ways to perform this evaluation from the simple rule-of-thumb to numeric or quantitative methods.

We present an ongoing project that evaluates different families of UML tools using the Logic Scoring of Preference (LSP) method. This method is very briefly presented and also some of the ongoing evaluation work directed to UML tool evaluation is explained.

Evaluating families of software tools such as database management systems, programming languages, web browsers, operating systems, or any other kind of software tools, etc., is done to choose one particular software among several possibilities or simply to asses several pieces of software.

Although this activity can have a great economic impact it is not always carry out with the care it should. There are several methods to do this evaluation ranging from the most informal to the more careful and formal, from the simpler form based on the personal opinion of evaluators, to the one that using the opinion of evaluators or users can construct a list of desired characteristics of the software and then analyse them against those characteristics, particularly assigning numerical values for the satisfiability of every desired characteristic for every software being evaluated. The result of this assignment can be a simple addition or more complex and sophisticated methods can

be used.

One of them is the Logic Scoring of Preference, which is the method we have been using to evaluate different families of software: web browsers, web programming languages and others to come. For more information on the method see [DUJ96], [DuBa97] and [DuEl82].

Past and Current Work

We have already used the LSP method to evaluate Data Base Management Systems [DFPS04], web browsers [FDD00] and also web programming languages [DPS03], as well as in the human resources field [DDF03]. We have constructed a list of desired characteristics for all of these evaluations and then used the LSP method to aggregate them and obtain results.

LSP is a method for the realization of complex criterion functions and their application in the evaluation, optimisation, comparison and selection of general complex systems.

As a starting point in the LSP method, it must be clearly determined what the user requirements, the main attributes of the system and their preference values are. These attributes are called performance variables. Each one of these variables is mapped into an elementary preference by defining and applying the corresponding elementary criteria.

In order to develop an exhaustive list of requirements, a hierarchical decomposition process for requirement derivation is applied. At the beginning all major groups of requirements are defined, and then through successive decompositions each group is decomposed into subgroups. By repeating this process the system Requirement Tree is obtained. The tree leaves correspond to the performance variables.

Elementary criteria are functions that transform real values from a performance variable into a value called elementary preference, which belongs to the [0,1] interval. They represent the degree of fulfilment of the requirements. Therefore, to define the different elementary criteria is necessary to have some previous experience to determine what is the range of acceptable values for each performance variable.

The elementary preferences are used as input for the LSP criterion function. This function yields a single global indicator of the degree of fulfilment of the system requirements. The LSP criterion function is built by aggregating the elementary preferences. To aggregate preferences means to replace a group of preferences (the input preferences) by a single preference (the output preference). It denotes the degree of satisfaction of the evaluator with respect to the group of input preferences. The process starts by aggregating groups of related elementary preferences and generating subsystem preferences. Therefore, the elementary preferences, corresponding to the system requirement tree leaves, are aggregated in new preferences, one by each elementary preference parent. This bottom-up process is repeated with the resulting groups of subsystem preferences until a single global preference can be computed.

If we want to aggregate n elementary preferences $E_1,...,E_n$ in a single preference E, the resulting preference E –interpreted as the degree of satisfaction of the n requirements– must be expressed as a function having the following properties:

1. The relative importance of each elementary preference E_i (i= 1...n) can be expressed by a weight W_i ,
2. $\min(E_1,...,E_n) \le E \le \max(E_1,..., E_n)$.

These properties can be achieved using the weighted power means:

$$E(r) = (W_1 E_1^r + W_2 E_2^r + ... + W_n E_n^r)^{1/r}$$, where

$0 < W_i < 1, \qquad 0 \le E_i \le 1, \qquad i = 1, ..., n$

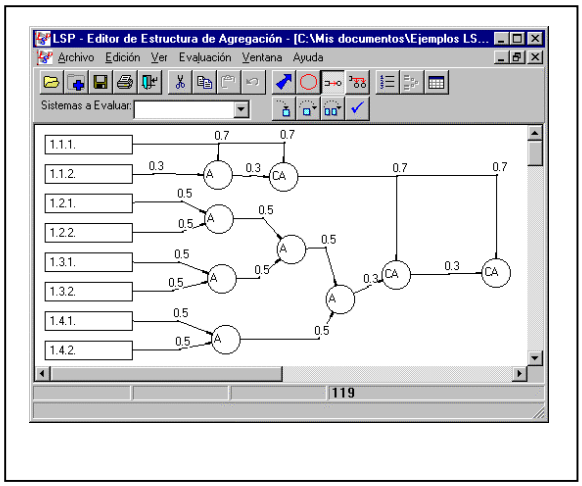
$W_1 + ... + W_n = 1,$

$\infty \le r \le +\infty$

The choice of r determinates the location of $E(r)$ between the minimum value $E_{min} = \min(E_1,...,E_n)$ and the maximum value $E_{max} = \max(E_1,...,E_n)$. For $r = -\infty$ the weighted power mean reduces to the pure conjunction (the minimum function) and for $r = +\infty$ to the pure disjunction (the maximum function), giving place to a Continuous Logic Preference (CPL). For a more detailed description of the technique for selection of r see [3], [4].

Normally the range between pure conjunction and pure disjunction is covered by a sequence of equidistantly located CPL operators: C, C++, C+, C+–, CA, C–+, C–, C– –, A, D– –, D–, D–+, DA, D+–, D+, D++, D.

In order to perform the evaluation more automatically we have developed a tool that implements the LSP method [DFPS01]. This tool has been used in several evaluations done by our group. An example of a screen of the tool is shown in Figure 1.



UML Tools Requirements

As was said above the first step when using the LSP method is to determine precisely the user's needs to be able to build the corresponding Requirement Tree.

In this particular case we are considering ourselves the users, so the process of building the Requirement Tree reverts to us. So we are currently developing the Requirement Tree and working on the desired characteristics that will help us to evaluate different UML tools.

Our ongoing research has examined some of the categories that would go into the Requirement Tree. Some of them are shown below. There is a brief discussion of what each item should evaluate.

- Complete UML support.

  This metric measures the extent to which tools support the different features of UML. Supported features are considered for all of the static and dynamic UML diagrams, that is Class Diagrams, Use Case Diagrams, Collaboration Diagrams, Sequence Diagrams, State Diagrams, Implementation Diagrams and Activity Diagrams.

- Direct and Reverse Engineering.

  Tools should support direct and reverse engineering into different languages –Java, C++, Delphi, etc. So this item will have a nearly direct relation to the number and probably the current 'weight'– of the languages supported.

  When doing Direct Engineering is useful to be able to go from the model to the code as well as from the code to the model to resynchronise the model with code at the end of every iteration.

  Also when doing Reverse Engineering this feature becomes useful in generating a model from the code when there was not a previous model.

- HTML Documentation.

Tools should permit hyperlink navigation, reasonable documentation generation times and bitmap images of the model.

- Selection Lists.

  In some diagrams –i.e. class– tools should provide the possibility of associating an object to a class chosen from a list, select messages send between objects, import or export classes to and from other packets or modules.

- Model Exportation and Importation

  A useful feature for a tool is the possibly of import and export models in a standard format such as XMI and also to save models in graphics formats such as JPEG, GIF, etc.

- Versioning.

  Different versions of the model should coexist and tools should have the possibility of keeping track of them.

- Navigation.

  This item measures the provisions for model navigation between different diagrams and versions. Zoom options as well as source code.

Conclusions and Future Work

We have been now for several years evaluating different software tools using LSP. We have presented here an outline of our current work in constructing a model for evaluating UML tools. The first step in the LSP evaluation process is the building of the Requirement Tree, which is the task that we are now undertaking.

The LSP method implies a permanent review process of every step that is quite similar to a spiral model. Therefore we expect to continue improving our current model and also starting to define Elementary Criteria and assigning values to the different Performance Variables and later constructing an Aggregation Structure.

We also are planning to continue expanding in the future the use of the LSP method to the evaluation of other software tools.

References

[BRJ98] G. Booch, J. Rumbaugh, I. Jacobson, “The Unified Modelling Language User Guide”. ISBN 0-201-57168-4. Addison-Wesley, Reading, MA, 1998.

[DDF03] N. Debnath, A. Dasso, A. Funes, G. Montejano, D. Riesco, and R. Uzal, "The LSP Method Applied to Human Resources Evaluation and Selection", Journal of Computer Science and Information Management, Publication of the Association of management/International Association of Management, Volume 3, Number 2, 2003, ISBN 1525-4372, pp.1-12.

[DFPS01] A. Dasso, A. Funes, M. Peralta, C. Salgado, “Una Herramienta para la Evaluación de Sistemas”, Workshop de Investigadores en Ciencias de la Computación, WICC 2001, Universidad Nacional de San Luis, San Luis, Argentina, May 2001.

[DFPS04] A. Dasso, A. Funes, M.Peralta, C. Salgado, “User Oriented Evaluation Models for DBMSs”, 33 Jaiio (ASIS 04), Córdoba, Argentina, 20-24 de Septiembre, 2004.

[DPS03] N. Debnath, M. Peralta, C. Salgado, A. Funes, A. Dasso, D. Riesco, G. Montejano, R. Uzal, "Web Programming Language Evaluation using LSP", Proceedings de CAINE03, Las Vegas, USA, 11-13 de Noviembre, 2003.

[DuBa97] J. J. Dujmovic and A. Bayucan, "Evaluation and Comparison of Windowed environments", Proceedings of the IASTED Interna Conference Software Engineering (SE' 97), pp 102-105, 1997.

[DuEl82] J. J. Dujmovic and R. Elnicki, "A DMS Cost/Benefit Decision Model: Mathematical Models for Data management System Evaluation, Comparison, and Selection", National Bureau of Standards, Washington, D.C., No. NBS-GCR-82-374, NTIS No. PB82-170150 (155 pages), 1982.

[DUJ96] J. J. Dujmovic, "A Method for Evaluation and Selection of Complex Hardware and Software Systems", The 22nd International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems. CMG96 Proceedings, vol. 1, pp.368-378, 1996.

[FDD00] A. Funes, A. Dasso, J. Dujmovic, G. Montejano, D. Riesco, R. Uzal, "Web Browsers Performance Analysis using LSP Method". Proceedings of the International Conference on Software Engineering Applied to Networking and Parallel/ Distributed Computing, SNPD' 00. Reims, France, May 18-21, 2000.

[OMG98] Object Management Group, "OMG Unified Modelling Language Specification". Framingham, MA, 1998. URL: http//www.omg.org