UMLx: A UML Diagram Analytic Tool for Software Management Decisions

Kan Qi, Dr. Barry Boehm

University of Southern California, Department of Computer Science, United States

kqi@usc.edu, boehm@usc.edu

Introduction

UML diagrams have been the major artifacts of the model-based approach of software engineering. It comprises 14 diagrams to describe the behavioral and structural aspects of a system. Also a subset of the diagrams have been adopted in the attempts to bring light-weight modeling into Agile software development[6]. In addition to its effectiveness in describing the behavioral and structural aspects of the system being developed, system specifications defined using UML diagrams are also effective indicators of software size, an accurate estimate of which can further be used to estimate effort, cost, and schedule[1] and guide decision making in various areas of software project management. For this reason, we propose a UML diagram analytic tool called UMLx which integrates a set of automated analytic procedures to extract information from UML diagrams based on the proposed metrics to facilitate decision making in risk management, planning, resource allocation, and system design.

Model Definition and Analytic Procedure

Parsing XMI files

The tagged elements of the XMI files are mapped into the UML elements defined in the UML metamodels.

User-System Interaction Model Construction

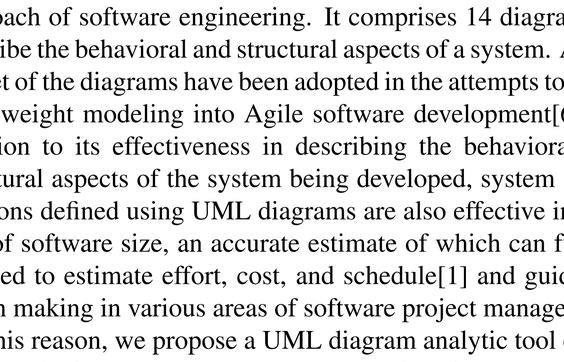

Figure 1: A XMI file is first parsed to construct a user-system interaction model. A user-system interaction model is defined by the following elements: activities (A), precedence relations (PR), system boundary (SB), system scope (SCP), stimuli (STL), and components (CMP).

Transaction Identification

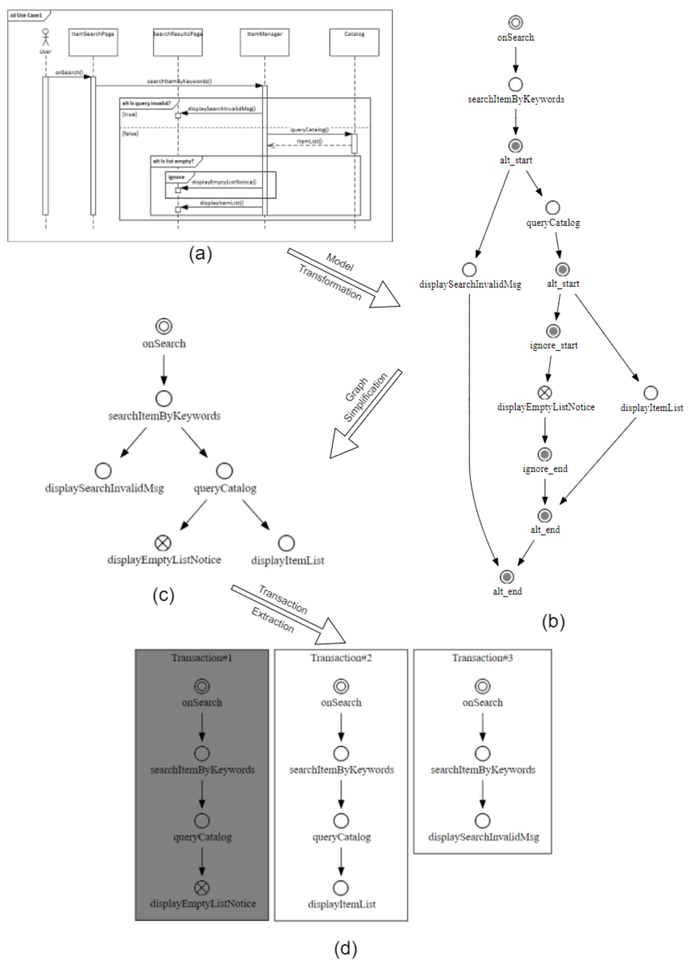

Figure 2: A sequence diagram (a) is first parsed into a control flow graph (b) with messages as nodes and precedence relations as edges, which is then simplified (c). A graph traversing algorithm is applied to identify the transactions (d).

Transaction Classification

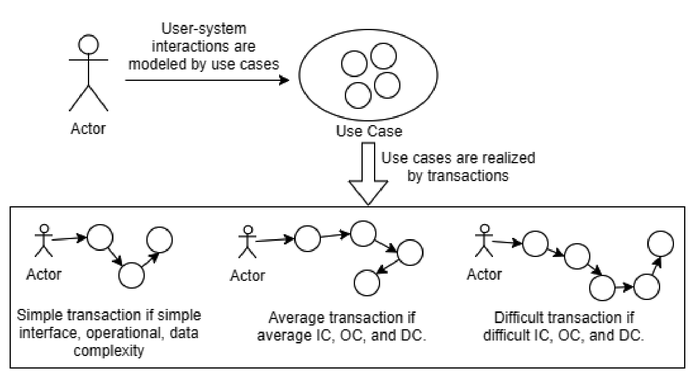

Figure 3: Transactions are classified into three levels of complexity with respect to interface, operational, and data complexities, which are evaluated based on the properties of the transactions.

Model Profiling

The information at the different levels: UML elements, transactions, use cases, and models, is profiled based on the proposed metrics.

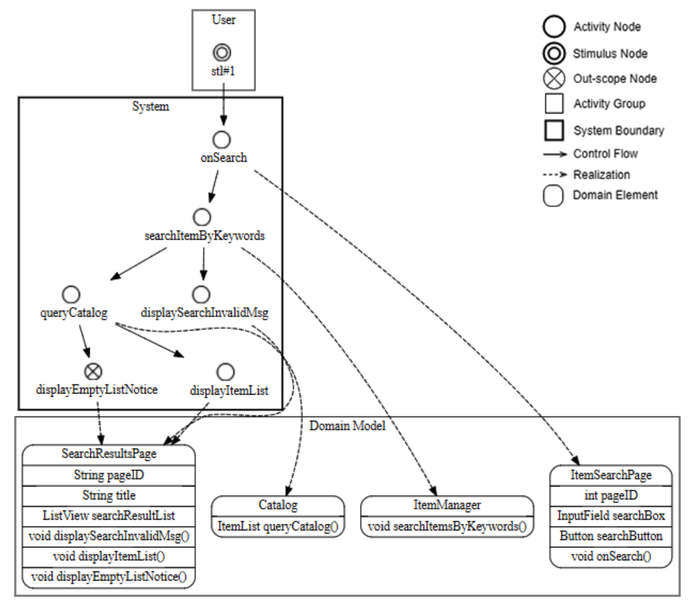

Figure 4: The derived user-system interaction model is visualized by a set of notations that represent the key elements.

UML-based Metrics

NT, SWT-I, SWT-II

Three size metrics are defined:

- NT, the number of transactions, which represents the number of basic units of system functionality.
- SWT-I, the sum of weighted transactions (type I), for which the transactions are weighted by TL and TD.
- SWT-II, the sum of weighted transactions (type II), for which transactions are weighted by TL, TD, and DETs.

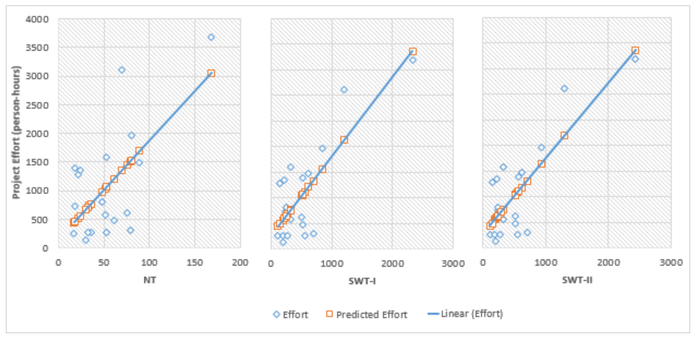

Figure 5: Linear regressions of project effort on the size metrics are applied based on empirically collected project data to calibrate linear models for effort prediction

CT

Transactions are categorized by their operational characteristics (CT). The categories indicate the different types of development workforce that are required for a project.

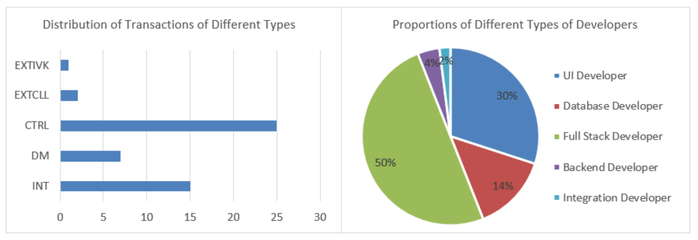

Figure 6: 7 operational characteristics are identified: interface management (INT), external input (EI), external inquiry (EQ), data management (DM), control (CTRL), external call (EXTCLL), and external invocation (EXTIVK). The distribution of the transactions with respect to the operational characteristics indicates the proportions of different types of developers to hire, which optimizes resource allocation strategy.

ATC, ATD

Two architectural complexity indices are defined:

- ATC, the average number of components of a transaction of a use case, represents the expected number of system components a developer needs to understand, create, modify, and test when implementing a transaction of a use case.
- ATL-ATD, the product of ATC and ATD, represents the architectural difficulty a developer needs to deal with when developing a transaction of a use case. ATD, the average transaction degree, represents the expected interface complexity when implementing a transaction of a use case.

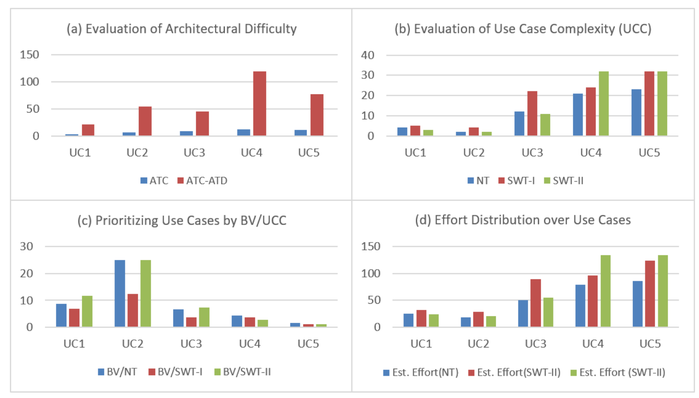

Figure 7: (a) Evaluate architectural difficulty for each use case using ATC and ATC-ADT to maintain the efficiency of development activities. (b) Evaluate use case complexity with NT, SWT-I, and SWT-II. (c) Prioritize use cases by BV/NT, BV/SWT-I, and BV/SWT-II to optimize ROI. (d) Estimated project effort is distributed to each use case based on their evaluated complexities.

Conclusions

- Introduced UML diagram analytic tool called UMLx, which is capable of applying various analyses to derive the information that facilitate decision making in various areas of software project management.
- Proposed a user-system interaction model that synthesizes information from the exploited UML diagrams to support the automated analyses. This computational model fills the gap between the sizing model and the UML metamodels, and preserves the potential of integrating other types of UML diagrams into the evaluation paradigm.
- Examples of how the derived data affect decision making are provided based on an empirical study of 24 student projects.

Future Directions

- To extend the capability of the automated framework to analyzing other types of UML diagrams, for example, state machines, object analysis diagrams, component diagrams, etc, would extend the use of the tool.
- Further evaluation of the relevance of the proposed metrics in making the software management decisions requires more data points to be collected.
- To support data collection, the tool needs to be implemented compatible with the XMI files exported from different UML modeling Tools.

References

[1] Barry W. Boehm. *Software engineering economics*. Prentice-Hall, Englewood Cliffs, N.J, 1981.

[2] International Function Point Users Group (IFPUG). *Function Point Counting Practices Manual*, 4.1.1 edition, 2002.

[3] Ivar Jacobson. *Object-oriented software engineering: a use case driven approach*. ACM Press ;Addison-Wesley Pub, [New York] :Wokingham, Eng. ;Reading, Mass, 1992.

[4] G. Karner. *Metrics for Objectory. Diploma thesis*. PhD thesis, University of Linkoping, 1993.

[5] Debasish Kundu, Debasis Samanta, and Rajib Mall. An approach to convert xmi representation of uml 2. x interaction diagram into control flow graph. *ISRN Software Engineering*, 2012, 2012.

[6] Doug Rosenberg, Barry Boehm, Bo Wang, and Kan Qi. Rapid, evolutionary, reliable, scalable system and software development: The resilient agile process. In *Proc. of the ICSSP17*, 2017.