Classifying Presence of Classes in UML Design using Software Metrics

Hafeez Osman HOSMAN@LIACS.NL
Michel R.V. Chaudron CHAUDRON@LIACS.NL
Peter van der Putten PUTTEN@LIACS.NL
Leiden University, Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands

Keywords: software engineering, UML, reverse engineering

The Unified Modeling Language (UML) is a widely used modeling language in the software industry. UML is used to graphically represent the design of software system. UML models which were created during the design phase are often poorly maintained during development. As a result the correspondence between design and implementation degrades as the implementation evolves considerably from its initial design (A.Nugroho, 2007). Indeed, it is well known that up-to-date design documentation is important in later stages of development and especially also in the maintenance phases. Also for legacy software, reliable designs are often no longer available. One popular technique in recovering the design is reverse engineering. Nevertheless, current reverse engineering techniques do not adequately solve this problem. In particular, too many details are presented. Hence, in order to achieve better design representations we need to learn which information from the implemented system to include and which information to leave out in reverse engineered designs.

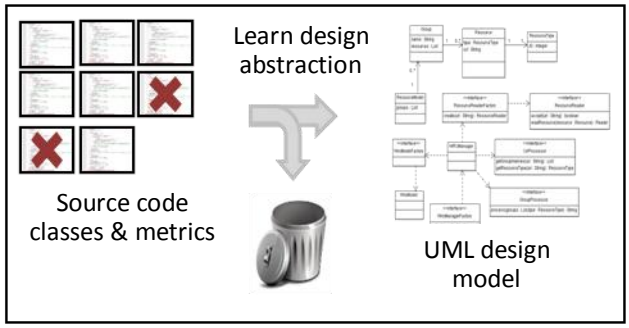

Figure 1: *Design Abstraction in UML Design Model*

This project specifically aims at reconstructing the class diagrams from source code in such a way that unnecessary detail that results from reverse engineering is eliminated (figure 1). This is work in progress and in this abstract we report on early results and open problems.

We have spent considerable effort to construct benchmark data sets for 10 pairs of UML design class diagrams and associated Java source code (obtained from open source projects). The number of classes (N) in the source code ranges from 59 to 903. The percentage of classes that is included in the UML design ranges between 3% to 47%.

Table 1: *Area under ROC Curve (AUC) from Classifiers*

Project	k-NN (1)	Func. Logistic	Dec. Tree	Dec. Table	Random Forest
Case 1 N=84	0.77 ±0.07	0.86 ±0.05	0.80 ±0.09	0.78 ±0.08	0.85 ±0.06
Case 2 N=214	0.86 ±0.02	0.82 ±0.07	0.74 ±0.07	0.81 ±0.03	0.86 ±0.03
Case 3 N=903	0.69 ±0.04	0.57 ±0.05	0.50 ±0.01	0.5 0	0.65 ±0.04

Our approach is to explore scoring models that, given a class in the source code of a system, provide a rank score for whether a class should be included in the UML design. A score provides flexibility for the end user to change the level of detail of the resulting reconstructed design. Initially we focus on using design characteristics of the source code through extracting software design metrics, which include: #in-dependencies, #out-dependencies, #operations, #attributes, #getters. In principle a large set of additional software metrics could be used (e.g. using SDMetrics).

Basic experiments show that mining this data is non trivial. For an initial study, we selected three case studies and experimented with five classifiers (Table 1). We randomly split the data set into 50% for training and 50% for test. To further improve reliability, we ran each experiment 10 times. The data can be highly unbalanced, so AUC is used as evaluation measure rather than accuracy.

Interesting future research directions include i) the use of semi-supervised learning or transfer learning to deal with the limited amount of labeled data for a given domain, ii) the integration of interactive user feedback, and the use of background or prior knowledge (existing designs). Data can be made available on request to parties wishing to collaborate on this problem.

References

A. Nugroho (2007). *A Survey of the Practice of Design-Code Correspondence amongst Professional Software Engineers*, Proc. 1st Int. Sym. on Empirical Software Engineering and Measurement, p.467-469.