Work in Progress: Intelligent Project Failure Analysis

Song Tan, Kai Qian, and Xiang Fu
stan@spsu.edu, kqian@spsu.edu, cscxzf@hofstra.edu

Abstract **- Sophisticated course projects in senior computer science classes have raised great challenges to educators. Students now have stronger needs for instant and individualized guidance when exposed to the complexity of software development. In this paper, we describe the application of Bayesian Network, a probabilistic graphic model, to automated project failure analysis. Students can make multiple project submissions to an automated grader before the project deadline. Each submission will be graded immediately, by running a collection of test cases prepared by instructors in advance. Then failure behaviors of test cases are piped to a Bayesian Network (BN) inference engine. The BN engine generates a report on the most probable failure causes, and helps students in removing software bugs. The analysis of the BN engine can be improved by parameter learning. A case study is presented to illustrate the effectiveness of the approach.**

Index Terms - Grading, Bayesian Network, Tutoring System, Automated Testing

INTRODUCTION

Modern automated grading systems (e.g., Web-Cat [1] and APOGEE [2]) have greatly contributed to the quality computer science education. Available 24X7, automated graders like APOGEE can demo a failed test case step by step online, which helps students to learn from failures and sharpen their skills via a cyclic improvement model. However, just like an old saying, all successful projects are similar, but every unsuccessful one has its own story. The failure of a project can be caused by many factors. It is still challenging to help students trouble shoot their projects.

We present our initial attempts in tackling the above challenge. BAUT (Bayesian driven AUtomated Tutoring system) , a research prototype on automated tutoring, is built upon *Bayesian Network* (BN) [3]. BN (an acyclic graph) is a formalism for specifying conditional dependencies among random variables in statistics. Each node in a BN represents a random variable. Nodes are connected using directed edges, which represent the probable causal relationship. There exist efficient algorithms for performing inference and learning on BN.

BN has been widely applied in education, mostly for representing relation of knowledge concepts in learning (see e.g., [4] and [5]). This inspired us in using BN for modeling project failure causes. In [6], we customized BN model (using a layered approach) and presented the design details of BAUT. This paper concentrates on a case study and the initial evaluation of our exploration.

BAUT ARCHITECTURE

As shown in Figure 1, BAUT consists of three major components: (1) an automated grader (based on AWAT [7]), which reads test cases from a spread sheet, takes URL of a student project submission, runs each test case and collects failure behaviors; (2) a Bayesian Network inference engine, which is built from the MSBNx [8] toolkit. Given the running results from the grader, the inference engine produces the probability of all potential failure causes stored in a BN model and advises students about the most probable cause. Students may later verify if the advice from BAUT is accurate and report the real cause of the previous failure; (3) a Web portal which interacts with students.

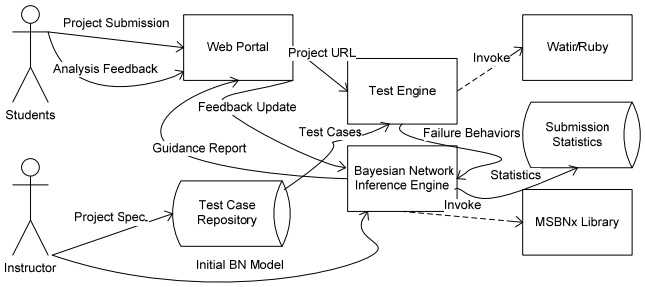

FIGURE 1
BAUT ARCHITECTURE

CASE STUDY

We now present a case study that illustrates the effectiveness of BN. A control group of seven students participated in the experiment. Each of the students is asked to implement a Java web application called "Bookstore Search Engine". The web application allows a user to enter the title or keyword of a book and returns all related books. It is required that the web application has to retrieve information from a back-end database that is prepared in advance by the instructor. Several sample test cases are given to students while the complete set of test cases is defined in a spreadsheet stored in BAUT.

An initial BN model (as shown in Figure 2, generated by MSBNx) is given by the instructor. Each node is a failure

cause (e.g., `SQLStatementError`) or a failure behavior (e.g., `NumberOfQueryResults`). The numbers associated with each node reflect its local probabilistic distribution. They are initialized by the instructor based on past project experiences. Here `SQLStatementError` encapsulates bugs related to SQL design. `NumberOfQueryResults` represents the failure behavior that the project returns more (or less) book records than specified in the test case. In the BN model, the arrows between nodes demonstrate their causal relationship. For example, a `SQLStatementError` may lead to `NumberOfQueryResults` error (this usually happens when the logical condition in the WHERE clause of a SQL query is wrong). Failure causes are layered to reflect their internal causal relations.

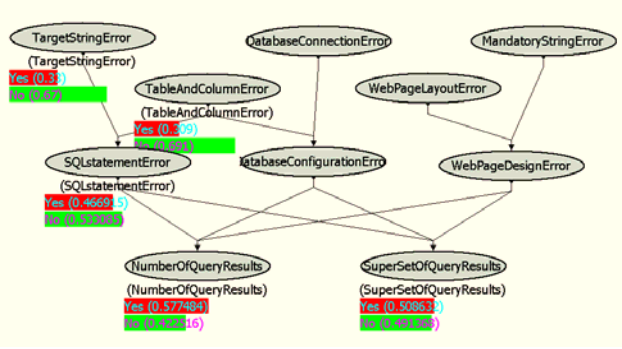

FIGURE 2
INITIAL BN MODEL

When a project is partially completed, its URL can be submitted to BAUT. BAUT generates a brief report which consists of the running results of all test cases and the analysis of the behaviors of all test cases. The most probable failure causes are reported. Then students are allowed to report the accuracy of the analysis using a web portal.

The report page contains a list of failure causes (i.e., nodes in the BN model). Because in BAUT, all nodes in a BN model are layered, they are displayed in a tabulated form in the report page. Students are required to report and select one failure cause from each of the layer in the BN model. Each feedback report from a student triggers the execution of parameter learning process immediately. The parameter learning process tries to update the local distribution table for those nodes in the bottom layers first and then propagates (reversely) along the causal links to parent nodes in upper layers. There is an additional MySQL database, which records the hit count for each failure behavior/cause.

The manual modeling of BN nodes and their causal relations may be imperfect. However, in practice, we found that imperfect model can still produce meaningful results and relatively accurate analysis. Table I shows the summary of the diagnosis provided by BAUT for the first round of submission by the seven students. The details of inference is given in [6].

TABLE I
EMPIRICAL EVALUATION DATA

SQL	Webpage	DBConfig	Recommend	Real Cause
0.53846	0.29166	0.17348	SQL	SQL
0.55172	0.29629	0.15199	SQL	WebPage
0.30140	0.57198	0.12662	Web Page	WebPage
0.29143	0.04875	0.65982	DBConfig	SQL
0.58102	0.29143	0.12755	SQL	SQL
0.15199	0.67894	0.16907	Web Page	WebPage
0	0	0	nothing	nothing

Here `SQL`, `WebPage`, `DBConf` are the acronyms for the error related to SQL query design, web page design, and database configuration (mentioned earlier in Figure 2). Each number in the first three columns is the probability (estimated by BAUT) of the error being the real cause of failure (for each student). For the first round, the accuracy of the analysis is 71.42%. After the 2nd round of submissions and the correction reported by the students (which is used to improve the BN model), the accuracy of the analysis is enhanced to 85.71%.

CONCLUSION

This paper presents a novel approach using Bayesian Network for providing instant project failure analysis information to students. The user experiences reported by both faculty and students are positive during the initial experiments. We plan to extend BAUT into a full-fledged automated tutoring tool and improve the parameter learning algorithm for handling larger BN models.

Acknowledgment: This work is supported by NSF under contract DUE-0836859 and DUE-0837275.

REFERENCES

[1] S. Edwards. “Using software testing to move students from trial-and-error to reflection-in-action.” In Proc. of 35th SIGCSE Technical Symp. Computer Science Education, ACM, 2004, pp. 26-30.

[2] X. Fu, B. Peltsverger, K. Qian, L. Tao, and J. Liu. "APOGEE: automated project grading and instant feedback system for web based computing." In Proceedings of 39th SIGCSE Technical Symposium on Computer Science Education, pp. 77-81. March 12-15, 2008.

[3] K. B. Korb, A. E. Nicholson. “Bayesian Artificial Intelligence.” ISBN-10: 1584883871, Chapman & Hall/CRC, 2004.

[4] L. Zhang, Y. Zhuang, Z. Yuan, and Guo-hua Zhan, “Auto diagnosing: An intelligent assessment system based on Bayesian Networks”, pp.T1G7-10, in Proc. Frontiers in Education (FIE), 2007.

[5] C.J Butz, S. Hua, R. B. Maguire . “A web-based Bayesian Intelligent Tutoring System for Computer Programming”, vol 4, issue 1, pp. 77-97, Web Intelligence and Agent System, 2006.

[6] K. Qian, X. Fu, and P. Bhattacharya, “BAUT: A Bayesian Driving Tutoring System.” In Proc. 7th Int. Conference on Information Technology: New Generations, 2010.

[7] M. Sztipanovits, K. Qian, X. Fu. “The automated web application testing (AWAT) system.” In Proceedings of the ACM Southeast Regional Conference 2008, pp. 88-93, March 2008.

[8] Microsoft, Microsoft Bayesian Network Editor, available at http://research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx, retrieved Nov. 2009.