Generating GUI-Based Benchmark Applications for Testing and Analysis

The purpose of Random Utility GeneRator for Analysis and Testing (RUGRAT) is to generate syntactically correct web-based GUI programs of arbitrary complexity using the notion of stochastic parse trees (SPT)[[1]](#footnote-1). The idea of SPT is that a language grammar is represented as an Abstract Syntax Tree (AST) with its nodes assigned different probabilities. The probability for mandatory syntactic grammar features is always one, while optional features have probabilities between zero and one.

Consider the following AST for a part of the Java grammar as it is shown in Figure 1. Class declaration is a mandatory feature of Java since every Java program must have at least one class. However, method declaration is an optional feature, a class may have zero or more methods. Thus, we can assign some probability to generate methods in a class, in theory RAG should determine itself what methods to generate and add them to a given class.

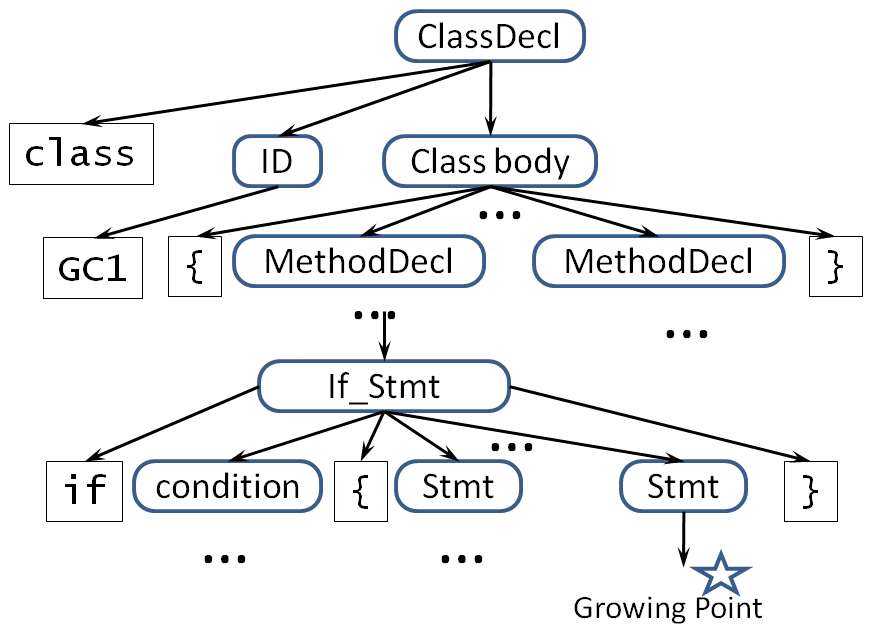


Figure 1. A part of the AST for Java grammar.

RUGRAT will employ a top-down approach for generating web-based Graphical User Interface (GUI) applications. At the absolute minimum, a GUI application must have one screen that has the button submit, which may or may not send a POST request to the web server. Other GUI features are added by RUGRAT based on configuration parameters that are defined by RUGRAT's users. For example, if the user specifies that the generated application should have between 10 and 100 screens and between 1,000 and 10,000 GUI objects, then RAGs will generate a random number C between 10 and 100 and a random number L between 10,000 and 100,000. Then, RAG will create C screens in the GUI application, adding other GUI objects to it until the number of these objects >= L. Because of the randomness, it is really difficult to keep the number of GUI objects equal to a specific integer. We could instead, fix the minimum value for L (e.g 100) and require that generated lines of code should not be more than 10% of the specified minimum value (e.g. 100 < L < 110). This is the essence of how RUGRAT works.

The rest of the paper describes features of GUI applications that RUGRAT should add to generated programs and how RUGRAT should do it. All GUI objects are generated for the Internet Explorer; JSP and other server-related technologies and reflection are not used. No concurrency will be used at this point.

## SCREENS

RUGRAT takes a configuration parameter that specifies the minimum and maximum numbers of screens and the minimum and the maximum number of GUI objects per screen. RUGRAT should have a separate properties file where all the parameters will be specified. The names of screens are unique within the generated application. The random generator picks the numbers of screens and their GUI objects within the limits of the configuration parameters. When generating a screen, RUGRAT will randomly decide what GUI objects this screen implements and it will add the corresponding GUI objects to this screen.

1. D. Slutz. Massive stochastic testing of SQL, In *Proceedings of the Twenty-Fourth International Conference on Very-Large Databases*, Morgan Kaufmann, pp. 618–622, 1998. [↑](#footnote-ref-1)