Supporting Quality Assurance of Document-Based Interactive and Dynamic Web Applications

Mohamed Sharaf Aun

Department of Information Engineering, Graduate School of Engineering, Nagoya University, Japan

Abstract:

The growth of the World Wide Web (Web) and its associated technologies have already had a significant impact on business, commerce, industry, banking and finance, education, government, and our personal and working life. In parallel with this growth, there is a growing concern about the manner in which Web applications are created and about their long-term quality and integrity. To cope with this concern, *Web Engineering* has just emerged as a new discipline for the establishment and use of sound scientific, engineering and systematic approaches to the successful development, deployment and maintenance of *high quality* Web applications.

This study falls within the context of this new engineering discipline. It is devoted to supporting quality assurance of Web applications that are Document-Based, Interactive and Dynamic (hereafter DBID Web applications). DBID Web applications are document-based in the sense that they allow programs or scripts to be embedded within documents (e.g.; HTML), thus turning the formerly passive text into an application itself. They are interactive since they interact with the user. They are dynamic since the embedded programs are executable scripts. We limit our attention to DBID Web applications, where client scripted pages are considered as the building blocks. Scripts are the means by which execution of the business logic is realized and they are the reasons behind making the Web more interactive and dynamic.

The focus is on detecting defects in source code of such applications. For this reason, we establish three research goals. The first goal is to propose an approach for debugging. We have chosen to start with debugging as it represents a core activity for supporting both reliability and maintainability quality attributes. It identifies and corrects the root causes of failures. The second goal is to facilitate quality assurance with separating features out of the implementation artifacts. The third goal is to describe architecture design of an environment dedicated for addressing key challenges involved in assuring quality attributes of such applications.

To achieve the research goal (1), in addition to presenting the challenges facing the debugging process and categorizing bugs, we aim at facilitating the debugging process by extending conventional techniques that support mix of static and dynamic capabilities. Because of the nature of the applications, the major differences are as follows: (i) since the script is executed on demand, the program validation technique such as type-checking does not fit. (ii) since the server can control the script loosely, the current state is hard to be determined in the present Run-Time Environments (RTEs). To check more correctness, we first separate the script analysis from the RTE. By imposing more strict syntactic checking, we can identify the defects before execution. Since such defects are not reported by the present RTE, we categorize such defects as *"silent bugs"*. Next, by inserting inspection codes, we identify the defects in behavior. We categorize such defects as *"active bugs"* since we can observe the unexpected behavior.

As for achieving the research goal (2), we first describe the need for separating features explaining its possible impacts on assuring quality. Then, we categorize useful features by referring to the general client/server model for Web applications. Next, we introduce the challenges. After that, we describe our approach for separating features. The approach allows for representing source code of an application with a graph that enables features to be separated, and then it shows how useful features can be identified. Finally, the effectiveness of our approach is illustrated by two practical examples. We show that, the approach provides opportunities for better analysis that can support debugging and facilitate conformance to quality attributes.

Finally, to achieve the research goal (3), we investigate the key challenges involved in assuring quality attributes of DBID Web applications, promising solution strategies and topics of importance. Several challenges are involved including the need for separating features out of the implementation artifacts, the need for fulfilling the analysis vacuity of source code due to the lack of compilation, the need for managing the challenges introduced by the dynamic property of the applications, the need for providing program execution controlling mechanisms, and the need for coordinating and cooperating the methodology activities with the surrounding environments. The aims in addressing the challenges are met, by presenting the architectural design of an environment that involves four components. These components are features separator, preprocessing, dynamic processing and validating interface components.

As conclusions, we found that combining both static and dynamic processing together with feature engineering techniques can play an important role in assuring several quality attributes. Static processing eliminates faults that cause the applications not to run. Dynamic processing eliminates behavioral bugs. Separating features helps a great deal in finding source code defects by throwing away irrelevant code. It also provides a mechanism for overcoming the challenges introduced by the coexistence of multiple technologies in an application. Moreover, separating features has several inherent advantages. Once a feature is separated, it can be debugged, validated, reused, or it can help in maintaining or understanding the overall system.