MultiObjective SearchBased Requirements Traceability Recovery for Adaptive Systems

Mohamed Salah Hamdi*, Adnane Ghannem, Hany Ammar

Ahmed Bin Mohamed Military College.
* mshamdi@abmmc.edu.qa

Complex adaptive systems exhibit emergent behavior. This type of behavior occurs in volatile environments involving cyber-physical systems, such as those aimed at smart cities operations. Adaptive systems’ maintenance aims at improving their performance by dealing with the treatment of continuous and frequently changing requirements. Adaptive systems’ behavior requires therefore up-to-date requirements traceability. To this end, we need to understand the requirements and to localize the program parts that should be modified according to the description of the new requirements. This process is known as Requirements Traceability Recovery (RTR). The process of generating requirements traceability, when done by a human (e.g., system maintainer), is time consuming and error-prone. Currently, most of the approaches, in the literature, are time consuming and semi-automatic that always need the intervention of the user. In our work, we are specifically interested in following the link between requirements and the code of the software system with the aim of helping the designer to update the system appropriately by automating the traceability process. To do this, we formulated the RTR problem as a complex combinatorial problem that could be tackled using Heuristic Search (HS) techniques. These techniques can intelligently explore a big search space (space of possible solutions) for a problem and find an acceptable approximate solution. Varieties of HS techniques exist in the literature. In our work, we use the Non-dominated Sorting Genetic Algorithm (NSGA-II), which is an improved version of a classic Genetic Algorithm (GA). NSGA-II is a multi-objective technique that aims at finding the best compromise between objectives (Pareto Front). The application of NSGA-II to a specific problem (i.e., the requirements
traceability recovery problem in our context) requires the accomplishment of the following five elements: 1. Representation of the individuals (vector, tree, etc.), 2. Definition of the evaluation function (fitness function), 3. Selection of the (best) individuals to transmit from one generation to another, 4. Creation of new individuals using genetic operators (crossover and mutation) to explore the search space, 5. Generation of a new population using the selected individuals and the newly created individuals. The proposed approach takes as input a software system, a set of requirements, and the maintenance history of the software system and produces as output the trace links, i.e., the artifacts (classes, methods, etc.) related to each requirement. Three objectives are defined to support the NSGA-II: (1) semantic similarity and (2) Recency of Change (RC), and (3) Frequency of Change (FC). We used the cosine of the angle between the vector that represents the requirement and the vector that represents the software element to measure the semantic similarity. To calculate the RC measure, we used the information extracted from the history of change accumulated during the maintenance process of the software system. The intuition behind introducing the RC measure is that artifacts (classes, methods, etc.) that changed more recently than others are more likely to change now, i.e., are related to the new requirements at hand. To calculate the FC measure, we used the information extracted from the history of change accumulated during the maintenance process of the software system. The intuition behind introducing the FC measure is that artifacts (classes, methods, etc.) that change more frequently than others are more likely to change now, i.e., are related to the new requirement at hand. Each solution consists of assigning each requirement to one or many artifacts (classes, methods) of the system. The solution should maximize as much as possible the three objectives mentioned before. Experiments have been conducted on three open source systems in order to evaluate the approach. The obtained results confirm the effectiveness of the approach in correctly generating the traces between the requirements and classes in the source code with a precision of 91% on average and a recall of 89% on average. We also compared our results to those obtained by two recent works. We noticed that our approach outperforms the two other approaches and has higher average for precision and recall for all three projects.