Search-based software engineering

4.1 DEFINITION: Search-based software engineering (**SBSE**) is an approach to apply metaheuristic search techniques like genetic algorithms, simulated annealing and tabu search to software engineering problems. It is inspired by the observation that many activities in software engineering can be formulated as optimization problems. Due to the computational complexity of these problems, exact optimization techniques of operations research like linear programming or dynamic programming are mostly impractical for large scale software engineering problems. Because of this, researchers and practitioners have used metaheuristic search techniques to find near optimal or good-enough solutions.

Broadly speaking SBSE problems can be divided into two types. The first is of the type black-box optimization, for example, assigning people to tasks (a typical combinatorial optimization problem). With this sort of problem domain, the underlying problem could have come from the software industry, but equally it could have originated from any domain where people are assigned to tasks. The second type are white-box problems where operations on source code need to be considered.

Definition

The basic idea of SBSE is to take a software engineering problem and convert it into a computational search problem which can be tackled with a metaheuristic. This essentially involves a number of stages. Firstly defining a search space (the set of possible solutions to the problem). This space is typically too large to be explored exhaustively and therefore a metaheuristic is employed to sample this space. Secondly, a metric (also called a fitness function, cost function, objective function or quality measure) is used to measure the quality of a potential solution. Many software engineering problems can be reformulated as a computational search problem.

The term "search-based application", in contrast, refers to using search engine technology, rather than search techniques, in another industrial application.

Brief history

One of the earliest attempts in applying optimization to a software engineering problem was reported by Webb Miller and David Spooner in 1976 in the area of software testing.[4] In 1992, Xanthakis and his colleagues applied a search technique to a software engineering problem for the first time.[5] The term SBSE was first used in 2001 by Harman and Jones.[6] Since then, the research community has grown to include more than 800 authors in 2013, from approximately 270 institutions in 40 countries.

4.3 Application areas

Search-based software engineering is applicable to almost all phases of the software development process. Software testing has been one of the major applications of search techniques in software engineering. Search techniques have also been applied to other software engineering activities, for instance, requirements analysis, software design, software development, and software maintenance.

Requirements engineering

Requirements engineering is the process by which the needs of a software's users and environment are determined and managed. Search-based methods have been used for requirements selection and optimization with the goal of finding the best possible subset of requirements that matches users' requests and different constraints such as limited resources and interdependencies between requirements. This problem is often tackled as a multiple-criteria decision-making problem and, generally speaking, involves presenting the decision maker with a range of good compromises between cost and user satisfaction.

Debugging and maintenance

Identifying a software bug (or a code smell) and then debugging (or refactoring) the software is largely a manual and labor-intensive endeavor, though the process is supported by a number of tools. One objective of SBSE is to automatically identify bugs (for example via mutation testing), then automatically fix them.

Genetic programming, a biologically-inspired technique which involves evolving programs through the use of crossover and mutation, has been used to search for repairs to programs by altering a few lines of source code. The GenProg Evolutionary Program Repair software was shown to be able to repair 55 out of 105 bugs for approximately $8 each.

Coevolution has also been used as an approach. It follows a predator and prey metaphor where a population of programs and a population of unit tests evolve together and influence each other.

Testing

Search-based software engineering has been applied to software testing, including automatic generation of test cases (test data), test case minimization and test case prioritization. Regression testing has also received some attention.

Optimizing software

The use of SBSE in program optimization, or modifying a piece of software to make it more efficient in terms of speed and resource use, has been the object of developing research interest and success. Genetic programming has been used to improve programs. In one instance, a 50,000 line program was genetically improved, resulting in a program 70 times faster on average.

Project management

A number of decisions which are normally made by a project manager can be done automatically, for example, project scheduling.

Tools

There are a number of tools available for SBSE approaches. These include tools like OpenPAT and Evosuite and a code coverage measurement for Python

Methods and techniques

There are a number of methods and techniques available. A non-exhaustive list of these tools includes

One method of profiling is via instrumentation in order to monitor certain parts of a program as it is executed.

A different approach is obtaining an abstract syntax tree associated with the program, which can be automatically examined to gain insights into the structure of a program.

Applications of program slicing relevant to SBSE include software maintenance, optimization, program analysis.

Code coverage allows measuring how much of the code is executed with a given set of input data.

Static program analysis

Industry acceptance

As a relatively new area of research, SBSE does not yet benefit from broad industry acceptance. One issue is that software engineers are reluctant to adopt tools over which they have little control or that generate solutions that are quite different from the ones humans would produce. In the context of SBSE use in fixing or improving programs, developers need to be confident that any automatically produced modification does not generate unexpected behavior outside the scope of a system's requirements and testing environment. Considering that fully automated programming has yet to be achieved, a desirable property of such modifications would be that they need to be easily understood by humans to favor program maintainability.

Another concern is that SBSE might make the software engineer redundant. Researchers have argued that, on the contrary, the motivation for SBSE is to enhance the relationship between the engineer and the program.

4.4 USER EXPERIENCE User Experience (**UX**) involves a person's behaviors, attitudes, and emotions about using a particular product, system or service. User Experience includes the practical, experiential, affective, meaningful and valuable aspects of human–computer interaction and product ownership. Additionally, it includes a person's perceptions of system aspects such as utility, ease of use and efficiency. User Experience may be considered subjective in nature to the degree that it is about individual perception and thought with respect to the system. User Experience is dynamic as it is constantly modified over time due to changing usage circumstances and changes to individual systems as well as the wider usage context in which they can be found.

Definitions

The international standard on *ergonomics of human system interaction*, ISO 9241-210, defines user experience as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service". According to the ISO definition, user experience includes all the users' emotions, beliefs, preferences, perceptions, physical and psychological responses, behaviors and accomplishments that occur before, during and after use. The ISO also list three factors that influence user experience: system, user and the context of use.

Note 3 of the standard hints that usability addresses aspects of user experience, e.g. "usability criteria can be used to assess aspects of user experience". The standard does not go further in clarifying the relation between user experience and usability. Clearly, the two are overlapping concepts, with usability including pragmatic aspects (getting a task done) and user experience focusing on users' feelings stemming both from pragmatic and hedonic aspects of the system. Many practitioners use the terms interchangeably.

The term usability pre-dates the term user experience. Part of the reason the terms are often used interchangeably is that, as a practical matter, a user will at minimum require sufficient usability to accomplish a task, while the feelings of the user may be less important, even to the user herself. Since usability is about getting a task done, aspects of user experience like information architecture and user interface can help or hinder a user's experience. If a website has "bad" information architecture and a user has a difficult time finding what they are looking for, then a user will not have an effective, efficient and satisfying search. Usability can essentially be considered a subfield of user experience. Put another way, the feelings of the user generally depend upon having a successful experience in accomplishing the task at hand. Users have been known to use systems with sub-optimal user experiences and even "bad" usability, in order to accomplish their task goals.

In addition to the ISO standard, there exist several other definitions for user experience. Some of them have been studied by Law et al.

According to Jim Miller, principal of Miramontes Computing, user experience "encompasses much more than the traditional 'user interface' issues, such as screen design and command structure. Rather, it's a broad collection of user-centric issues that cut through the full extent of a project."[4]

History

The term *user experience* was brought to wider knowledge by Donald Norman in the mid-1990s. He never intended the term "user experience" to be applied only to the affective aspects of usage. A review of his earlier work] suggests that the term "user experience" was used to signal a shift to include affective factors, along with the pre-requisite behavioral concerns, which had been traditionally considered in the field. Many usability practitioners continue to research and attend to affective factors associated with end-users, and have been doing so for years, long before the term "user experience" was introduced in the mid-1990s. In an interview in 2007, Norman discusses the widespread use of the term "user experience" and its imprecise meaning as a consequence thereof.

Several developments affected the rise of interest in the user experience:

1. Recent advances in mobile, ubiquitous, social, and tangible computing technologies have moved human-computer interaction into practically all areas of human activity. This has led to a shift away from usability engineering to a much richer scope of user experience, where users' feelings, motivations, and values are given as much, if not more, attention than efficiency, effectiveness and basic subjective satisfaction (i.e. the three traditional usability metrics).
2. In website design, it was important to combine the interests of different stakeholders: marketing, branding, visual design, and usability. Marketing and branding people needed to enter the interactive world where usability was important. Usability people needed to take marketing, branding, and aesthetic needs into account when designing websites. User experience provided a platform to cover the interests of all stakeholders: making web sites easy to use, valuable, and effective for visitors. This is why several early user experience publications focus on website user experience.

The field of user experience represents an expansion and extension of the field of usability, to include the holistic perspective of how a person feels about using a system. The focus is on pleasure and value as well as on performance. The exact definition, framework, and elements of user experience are still evolving.

User Experience of an interactive product or a web site is usually measured by a number of methods, including questionnaires, focus groups, and other methods. A freely available questionnaire (available in several languages) is the User Experience Questionnaire (UEQ). The development and validation of this questionnaire is described in Google Ngram Viewer shows wide use of the term starting in the 1930s., "He suggested that more follow-up in the field would be welcomed by the user, and would be a means of incorporating the results of user's experience into the design of new machines." Use of the term in relation to computer software also pre-dates Norman.

Influences on user experience

Many factors can influence a user's experience with a system. To address the variety, factors influencing user experience have been classified into three main categories: user's state and previous experience, system properties, and the usage context (situation). Studying typical users, contexts, interactions and resulting emotions help in designing the system.

Momentary emotion or overall user experience

Single experiences influence the overall user experience: the experience of a key click affects the experience of typing a text message, the experience of typing a message affects the experience of text messaging, and the experience of text messaging affects the overall user experience with the phone. The overall user experience is not simply a sum of smaller interaction experiences, because some experiences are more salient than others. Overall user experience is also influenced by factors outside the actual interaction episode: brand, pricing, friends' opinions, reports in media, etc.

One branch in user experience research focuses on emotions. This includes momentary experiences during interaction: designing affective interaction and evaluating emotions. Another branch is interested in understanding the long-term relation between user experience and product appreciation. The industry sees good overall user experience with a company's products as critical for securing brand loyalty and enhancing the growth of customer base. All temporal levels of user experience (momentary, episodic, and long-term) are important, but the methods to design and evaluate these levels can be very different.