Recent Advances in Recommendation Systems for Software Engineering

Robert J. Walker
University of Calgary
Calgary, Canada
walker@ucalgary.ca

Abstract. Software engineers must contend with situations in which they are exposed to an excess of information, cannot readily express the kinds of information they need, or must make decisions where computation of the unequivocally correct answer is infeasible. Recommendation systems have the potential to assist in such cases. This paper overviews some recent developments in recommendation systems for software engineering, and points out their similarities to and differences from more typical, commercial applications of recommendation systems. The paper focuses in particular on the problem of software reuse, and speculates why the recently cancelled Google Code Search project was doomed to failure as a general purpose tool.

Keywords: RSSEs, overview, classification, opportunities.

1 Introduction

At one time, the notion of a recommendation system for software engineering (RSSE) would have been anathema: computers and software were all about correctness and precision, while recommendations bespeak fuzziness and faith. Unfortunately, many tasks that face the software engineer are inherently imprecise, staggering complex, and often infeasible to compute (or even formally undecidable). The software engineer must bridge the gaps between: the messy, changing needs of the real world; the interpersonal and organizational challenges involved in collaborating with other people; and the inflexibility of the computer. While correctness and precision have their place, a willingness to revert to “good enough” answers can be the difference between success and failure in the real world with its constraints on time and money.

Robillard et al. [14] have previously defined RSSEs as follows: “An RSSE is a software application that provides information items estimated to be valuable for a software engineering task in a given context.” A key word in that definition is “estimated”, because if the correct answer can be faithfully returned in all circumstances, the application is not making recommendations but computing or finding the answer. For example, a search for all references to a particular method within a software project is not a recommendation: there is a unique, unequivocally correct set. In contrast, finding all locations in a system that
should be changed when that method is changed will require recommendations, due to the infeasibility of computing the correct answer in general.

Recommendations for a software engineer play much the same role as recommendations for any user. There is a problem or task at hand. There is (often highly structured) information available about potential solutions. Some additional information about the specific task is often available—e.g., through an integrated development environment (IDE)—as is information about what other people have done in similar circumstances. The recommendation system’s task is to infer the nature or details of the task, the needs and/or characteristics of the context of the task and of the engineer, to construct one or more recommendations according to some model of relevance, and to present these recommendations in a manner conducive to their timely and efficient usage.

An analysis of the differences and similarities of different recommendation systems can be facilitated by a classification scheme. Section 2 outlines three different schemes for classifying recommenders in general recommenders or RSSEs in particular: the paradigmatic classification for general recommenders provided by Jannach et al. [9], the multidimensional design classification that provided by Robillard et al. [14] for RSSEs, and a novel problem-space classification.

Section 3 examines a small set of recent RSSEs, to illustrate the breadth of applications that have been explored. Section 4 focuses more narrowly on software reuse, being a cornerstone of the author’s research and a fundamental goal that led to the field of software engineering. Section 5 points to some opportunities with RSSEs that the author has seen in his own work. Section 6 summarizes and points out remaining challenges and opportunities in RSSE and potential cross-fertilization with research in recommenders from other domains.

## 2 Categorizing RSSEs

Three categorization schemes are presented here: the paradigmatic classification for general recommenders provided by Jannach et al. [9], the design-oriented classification for RSSEs developed by Robillard et al. [14], and a novel problem-space classification.

Jannach et al. [9] divide recommenders into four paradigms. Their stated focus is personalized recommendations (i.e., ones reflective of the individual’s tastes, needs, habits, or characteristics) rather than generic recommendations. The archetypal example is of the online bookstore, where recommendations are made to a customer about what books may be of interest to them.

1. **Collaborative recommenders.** A typical example is, if Alice has purchased or viewed books that have been purchased or viewed by other customers, additional books purchased or viewed by those other customers are also of potential interest to Alice. Collaborative recommenders can thus be viewed as operating through analogical reasoning over different users, and extrapolation from historical data, without resorting to detailed analysis of the items being recommended.