Case-based Reuse with Partial Requirements Specifications

Hermann Kaindl
Vienna University of Technology
kaindl@ict.tuwien.ac.at

Michał Śmiałek and
Wiktor Nowakowski
Warsaw University of Technology
{smialek, nowakoww}@iem.pw.edu.pl

Abstract

A case-based approach allows reuse without the usual and significant effort for making software explicitly reusable. We even support such reuse for only partially *developed requirements, since it allows reuse already without the need to develop a "complete" specification first. The solution information (models and code) of (one of) the most similar problems can then be taken for reuse and adapted to the newly specified requirements. And even the specification of these new requirements can be facilitated, since the retrieved software case contains related requirements, which may be reused as well.*

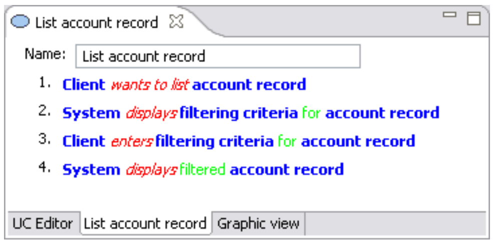
1 Introduction and Background

Several approaches to requirements reuse have been identified, e.g., in [2]. As pointed out there, extensive generalization of requirements is often not suitable. Our work[1] is based on a different approach where reuse is organized around specific software cases stored in repositories [3].

It is essential to quickly and easily find appropriate cases for reuse. Our approach employs similarity metrics for finding good candidates [5]. They calculate similarity scores for the stored cases to a given requirements specification. In fact, they can achieve good results even for *partially* developed requirements. From a set of cases with high similarity scores, one or more can be selected for reuse. Their design and code can be adapted for the new problem at hand. And even the stored requirements can be reused in this way.

2 How to Find Similar Software Cases

Let us assume that we start a new software project called "e-Banking System" with the creation of a scenario. We acquire a basic sequence of interaction for listing account records according to a filter. This sequence is entered in our scenario editor and looks as shown in Figure 1.



Name: List account record

1. Client wants to list account record
2. System displays filtering criteria for account record
3. Client enters filtering criteria for account record
4. System displays filtered account record

Figure 1: Scenario used for query.

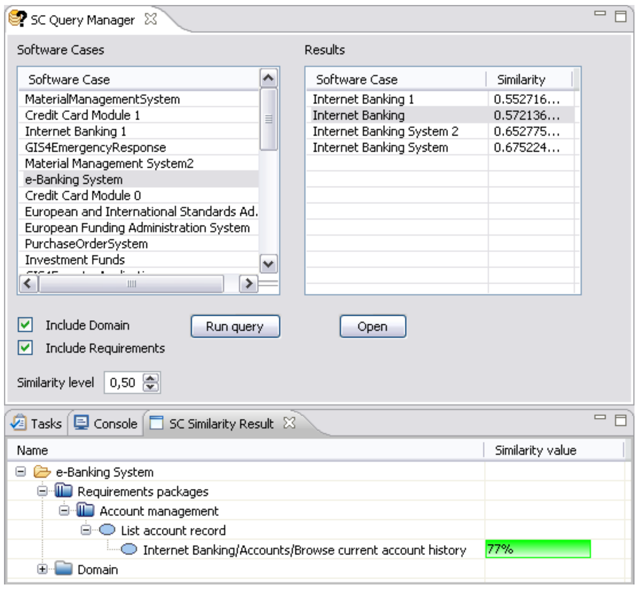
Actually, this text made of SVO (subject – verb – object) sentences is already hyperlinked with a domain specification in this screen shot. Let us assume that such a domain specification is already available, since this is not the first banking application in our organization. If we could not directly make such a reuse, we may still reuse terminology from *WordNet*[2] as described in [4]. This would involve adding concepts like "account record" to the specific application domain. The linking has to be done manually, but it is supported by the tool.

Normally, we would continue with specifying other scenarios and elaborating a fully-fledged requirements specification. Based on it, we would design the software and implement it, all according to an iterative and incremental life cycle approach.

Assume that our organization has done many banking applications previously and stored them as software cases in our tool, i.e., collections of requirements, design models and code. So, we may want to reuse one (or more) of them. This does not involve manual effort for preparing anything in the software cases for reuse, but it requires to find the most appropriate ones for reuse, presumably the most similar ones. Assuming that the various banking ap-

[1]This research has been carried out in the ReDSeeDS project and partially funded by the EU (contract number IST-33596 under the 6th framework programme), see `http://www.redseeds.eu`.

[2]WordNet is a semantic lexicon that was developed at the Cognitive Science Laboratory at Princeton University, see `http://wordnet.princeton.edu/`.



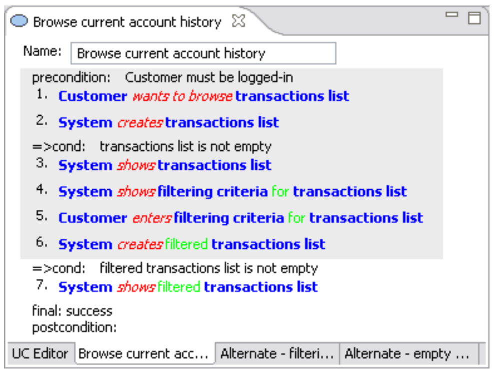
Software Case
MaterialManagementSystem
Credit Card Module 1
Internet Banking 1
GIS4EmergencyResponse
Material Management System2
e-Banking System
Credit Card Module 0
European and International Standards Ad.
European Funding Administration System
PurchaseOrderSystem
Investment Funds

Software Case	Similarity
Internet Banking 1	0.552716...
Internet Banking	0.572136...
Internet Banking System 2	0.652775...
Internet Banking System	0.675224...

✓ Include Domain
✓ Include Requirements
Similarity level 0,50

Name	Similarity value
e-Banking System	
Requirements packages	
Account management	
List account record	
Internet Banking/Accounts/Browse current account history	77%
Domain	

Figure 2: Querying for similar cases with the tool.



Name: Browse current account history
precondition: Customer must be logged-in
1. Customer *wants to browse* transactions list
2. System *creates* transactions list
=>cond: transactions list is not empty
3. System *shows* transactions list
4. System *shows* filtering criteria for transactions list
5. Customer *enters* filtering criteria for transactions list
6. System *creates* filtered transactions list
=>cond: filtered transactions list is not empty
7. System *shows* filtered transactions list
final: success
postcondition:

Figure 3: Most similar scenario matching query.

plications had several different developers, no one would have an overview sufficient to tell the most similar ones.

Therefore, it is useful to ask the tool for the most similar cases. One might think that a more or less "complete" requirements specification would be needed for such a query. However, the partial specification containing only the *single* scenario linked to a domain specification can already be used for a query, resulting in a list of the most similar software cases as shown in the top right of Figure 2.

According to the given threshold, only a few software cases with the highest similarity scores are listed. When these scores are close, the similarity metrics used for calculating them, see [5], could not tell the most appropriate case for reuse, of course. Still, the preselection is useful for finding similar cases for manual inspection, e.g., by a requirements engineer and a software designer. They should make their decision to choose a software case (even with a lower similarity score than others), e.g., because of its architectural style, depending on non-functional requirements for our new project. In this way, they may judge the software case called "Internet Banking" as the most appropriate one for reuse. It includes the scenario called "Browse current account history" as shown in Figure 3.

The similarity of this scenario and the one used for the query is calculated as 77 percent, as shown at the bottom of Figure 2. In fact, this similarity score influences the similarity score with the whole software case. But what makes these two scenarios highly similar as judged by our metrics? First, it is their similar structure of sentences. This part of the similarity metric is actually calculated based on underlying graph representations, measuring graph similarity [1]. Every sentence and sentence part is a node in such a graph, and these nodes are connected corresponding to the text structure. In addition, the "semantic" similarity of the notions used is taken into account. The verbs and the objects in the sentences of our similar scenarios are close in their meaning, as measured trough WordNet distance. This is to be preferred over measuring similarity of words. For more details on these metrics, we refer the reader to [5].

3 Conclusion

Our case-based approach allows reuse without the usual and significant effort for making software explicitly reusable. Supporting the reuse for only *partially* developed requirements is important, since it allows reuse already without the need to develop a "complete" specification first, and since it even facilitates the reuse of requirements. In our example, even a single new scenario is sufficient for finding relevant cases automatically.

References

[1] D. Bildhauer, T. Horn, and J. Ebert. Similarity-driven software reuse. In *Proc. International Workshop on Comparison and Versioning of Software Models (CVSM 2009)*, pages 31–36, 2009.

[2] W. Lam, T. A. McDermid, and A. J. Vickers. Ten steps towards systematic requirements reuse. In *Proc. 3rd Int. Symposium on Requirements Engineering*, pages 6–15, 1997.

[3] M. Śmiałek, A. Kalnins, A. Ambroziewicz, T. Straszak, and K. Wolter. Comprehensive system for systematic case-driven software reuse. *Lecture Notes in Computer Science*, 5901:697–708, 2010. SOFSEM'10.

[4] K. Wolter, M. Smialek, D. Bildhauer, and H. Kaindl. Reusing terminology for requirements specifications from WordNet. In *Proceeding of the 16th IEEE International Requirements Engineering Conference*, pages 325–326, 2008.

[5] K. Wolter, M. Śmiałek, L. Hotz, S. Knab, J. Bojarski, and W. Nowakowski. Mapping MOF-based requirements representations to ontologies for software reuse. In *CEUR Workshop Proceedings (TWOMDE'09)*, volume 531, 2009.