Construction of Ontology-Based Software Repositories by Text Mining

Yan Wu, Harvey Siy, Mansour Zand, and Victor Winter

College of Information Science and Technology,
University of Nebraska at Omaha,
Omaha, Nebraska 68182, USA
{ywu,hsiy,mzand,vwinter}@mail.unomaha.edu

Abstract. Software document repositories store artifacts produced in the course of developing software products. But most repositories are simply archives of documents. It is not unusual to find projects where different software artifacts are scattered in unrelated repositories with varying levels of granularity and without a centralized management system. This makes the information available in existing repositories difficult to reuse. In this paper, a methodology for constructing an ontology-based repository of reusable knowledge is presented. The information in the repository is extracted from specification documents using text mining. Ontologies are used to guide the extraction process and organize the extracted information. The methodology is being used to develop a repository of recurring and crosscutting aspects in software specification documents.

Keywords: Software Repository, Early Aspects, Text Mining, Ontology.

1   Introduction

Efficient reuse requires software artifacts to be stored, presented and organized in such a way that makes them easy to retrieve and use. Potentially reusable artifacts include code (ranging in size from modules to applications), project documents, test cases, software architecture, design patterns, etc. Most existing software repositories are nothing more than archival storage of documents. In order to facilitate the storage and retrieval of reusable assets, the information in existing repositories need to be structured in a way that takes into account the meaning (semantics) of the documents.

In this paper, a framework to construct a searchable repository of crosscutting aspects in software specification documents is presented. The organization of this paper is as follows: Section 2 introduces the problem and existing techniques. Section 3 is the description of the framework that represents the structure and processes in constructing and retrieving, utilizing the early aspects repositories with the aid of other relevant technologies and some guidelines on how to realize this framework. Section 4 provides conclusions and discusses ongoing and future work.

2   Background

2.1   Software Repositories

Due to the effort and cost of developing new software, it is desirable to reuse existing artifacts from previous software development efforts [1]. Software repositories provide a potential source of reusable artifacts. Software repositories store artifacts used in the analysis, design, implementation, testing and maintenance of software systems. They provide mechanisms to organize and retrieve required artifacts.

A repository may store artifacts that belong to one project, or to several projects within a given application domain. We are interested in constructing domain-level repositories that use domain knowledge to organize and manage artifacts. Creating domain-specific repositories is a crucial factor for successful reuse of artifacts [2]. However, most artifacts are stored in scattered project-level repositories, which often contain overlapping information. Although the existence of scattered overlapping software repositories allows each repository to tailor its content and service [3], they are inefficient when searching for reusable artifacts.

Aside from scattered and overlapping repositories, a cursory examination of existing public software repositories uncovered several other problems. These include the the lack of systematic design of the software repository management system (especially for the representation model for the artifacts) and inadequate search and retrieval mechanisms.

2.2   Text Mining and Ontologies

Text Mining [4] focuses on the discovery of new or previously unknown information by automatically extracting information from different written resources. A key element is the integration of the extracted information to form new facts or new hypotheses to be explored further by conventional experimentation.

Ontology is a concept from philosophy for understanding reality. To explain simply, an ontology consists of concepts and the relationships between them.

We propose the application of text mining for clustering artifacts into domain-specific groups and for retrieving concepts and relationships from the groups of artifacts to construct ontologies to organize the structure of software repositories. It can also be applied to extract users search patterns.

2.3   Related Work

There are many techniques and tools for organizing reusable repositories. Most of them are geared towards code reuse. Among the ones that work with unstructured documents, we focus on several techniques based on text mining to construct ontologies for representing and classifying knowledge from these documents.

Seeling, et al. use text mining combined with metadata-based information brokering to discover and analyze metadata from weakly structured text documents

[5]. This approach uses a tool called DocMINER to cluster similar documents. Jiang, et al. constructed a system called Concept Relation Concept Tuple based Ontology Learning (CRCTOL) to mine semantic knowledge to construct ontology from domain-specific documents [6]. The ontology is constructed through natural language processing, where parsed text is analyzed using statistical and lexico-syntactic methods to accurately identify key concepts and semantic relationships.

3   A Framework for Constructing Ontology-Based Repositories Using Text Mining

In this section, the proposed framework and guidelines for constructing a domain-specific repository is presented. While the proposal can be applied to construct repositories for any artifact, we illustrate its use in creating repositories of specification documents and focus on the retrieval of *early aspects* from these documents.

Early aspects are crosscutting concerns identified in the early stage of software development such as requirement specifications and architecture [7]. The identification of these aspects enables them to be modularized. This makes specifications and architectures more maintainable through localizing changes. It also enables downstream activities such as design and implementation to localize components related to certain concerns.

The goal of early aspect repositories is to construct a flexible and maintainable repository that reflects current understanding of aspect oriented software development (AOSD), and to use this knowledge to predict and plan for future AOSD projects. Previous research in early aspects repositories mostly focus on the development of technologies and tools to support mining knowledge from them. Here we focus on the retrieval and reuse of early aspects stored in the repositories. A proposed framework is shown in Figure 1.

The intention of constructing this framework was to provide a knowledge base to help developers define and locate early aspects effectively and efficiently so that in future AOSD projects, crosscutting concerns can be identified as early as possible. And when developers analyze future requirements in certain domains, this knowledge base can provide clues to identify the possible aspects.

The rest of this section describes the framework. The input into this process is a collection of software requirements specification documents from different application domains.

Clustering stage divides requirement specifications collected from different software repositories into domain-specific groups. Beck et al. provides a useful tool named DocMINER (Document Maps for Information Elicitation and Retrieval) which produces a document map to represent the inter-document similarity for a document corpus [8]. In this research, domain-specific terms that appear in the requirements specifications are the links among them for the same domain. The important issue is that the classification according to the similarity largely depends on the quality of the requirements specifications. Large amount
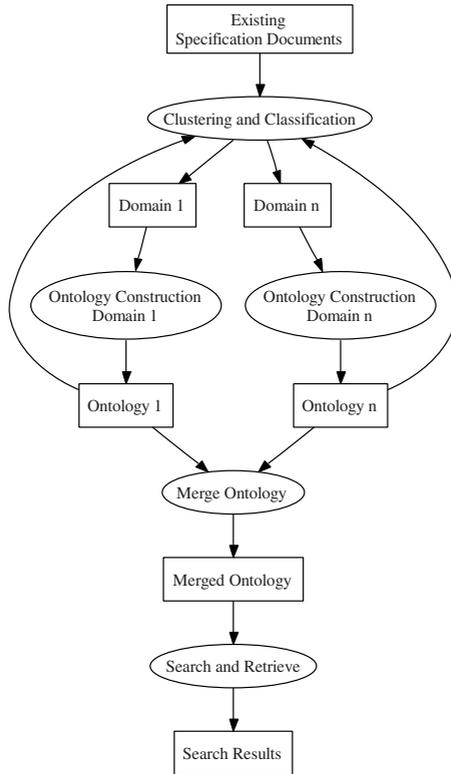
Fig. 1. Framework to manage and retrieve early aspects from requirements specification

of semantically domain rich terms can help differentiate the requirements clearly. The richer, the better, but at the same time the exactness and accuracy are important too. The other issue is that in the clustering process, a fuzzy cluster with some overlap will be a better choice than the exact cluster result.

Ontology construction is guided by domain experts. An ontology is constructed by defining concepts, properties, relationships, instances and axioms for a given domain. With ontology editors, domain experts can organize the domain knowledge and relationships well based on their knowledge. Morshed et al. concluded that in the three ontology construction tools they investigated, Protégé can be used to construct small-sized ontologies and OntoEdit, with its full functionality, can be used to develop medium to large-sized ontologies. Metis is good for enterprise architecture modeling [9].

Ontology construction consist of the following basic activities: first, the concepts, properties and relationships of any candidate early aspects from requirements specifications in a domain are identified using information retrieval

techniques with the help of domain experts; then a hierarchy is constructed to abstract the commonalities from clusters of similar early aspects into classes and subclasses. The commonalities, variabilities and relationships between different classes become the properties of the classes; then the concrete early aspects are represented by instances that belong to specific classes. Existing early aspects ontologies can also be added into this framework.

To construct ontologies that support the organization of early aspects repository, redundancy and overlap are necessary. As in the clustering activity, there will always be early aspects that cannot be clearly clustered into only one domain. For example, a security requirement can appear in different areas such as bank system, website design and so on. They have their own features but commonalities make them share information in the ontology construction. Hence, an aspect can be classified in several domains. This reduancy has the further advantage of being able to describe the aspect from different perspectives.

The constructed ontologies can be **merged** together. There can be intermediate ontologies between the upper ontologies such as SUMO and the low level domain-specific ontologies [10]. And for future use, concepts and relationships stored in the merged ontology and initial ontologies can be regarded as dictionaries to aid the clustering stage in a supervised manner. Subsequent clustering activities can improve their accuracy with the help of existing ontologies. Ontology is increasingly important in domain engineering with the development of domain analysis and design. The features of ontology such as the structure of knowledge and efficient retrieval are potentially useful in domain engineering.

The **search and retrieval interface** is designed to communicate with users to help them retrieve required early aspects from the merged early aspects repositories. According to the mismatch between the user's mental model and the system data model, a supervised retrieval method is recommended. There are ontology query languages such as HP's SPAQL for Jena which potentially can be the backbone of the search and retrieval interface. In addition to exact attribute match or wildcard match, natural language processing ability is a potentially useful aid for users to identify their real meanings. User usually cannot describe what they need exactly, so the search criteria should be allowed to have imprecision. These search criteria can be processed through natural language understanding to provide the alternative artifacts to choose. Even for the novices, a wizard that guides them to narrow down the requirement will be helpful.

3.1   Evolving and Refining the Repository

The basic framework presented can be extended in several ways in order to refine the ontologies in the repository.

Term addition. In practice, the mental model of the user will be different from that of the domain expert who designed the repository. This leads to a disparity between the user's understanding and the way the ontologies are organized. This makes it difficult for the user to retrieve the relevant early aspect. One solution is to allow users to evolve the ontology by adding their own concepts and terms. This had been applied in earlier reuse repositories such as [11]. This

is based on the idea that as programmers gain experience, their understanding of the knowledge in a particular domain gravitates toward a common structure [12]. The users will try their own keywords and feature descriptions to search for required early aspects, sometimes their words just do not fit the searchable terms that system defined but still useful to describe and locate the early aspects and could be more suitable for user's mental model.

Users are not domain experts, but from the perspective of system usage, users are people who really utilize the system to fit for their own requirement. Further, domain experts can make mistakes in identification of concepts and relationships among them. The other motivation for this stage is that the environment is dynamically changing and the terms and their relationships are changing. So the idea that let users to help us maintain the ontology structure is practical and useful. To get this capability, a database that records the usage scenarios of users is needed. User-defined terms and descriptions are emphasized and collected, analyzed to identify the suitable ones that could be added to the defined structures of existing ontology. The results are evaluated by the domain experts to decide whether or not revise the structure of the ontology to reflect the user's attitude and terms. A Bellcore study of people choosing terms to describe objects revealed that the probability of two people choosing the same keyword for these objects is between 10 and 20%, using 15 aliases will achieve 60-80% agreement, and 30 aliases can get up to 90% agreement [13]. Then revised ontologies with redundancy are used to facilitate the artifacts retrieval. This is an evolving process to help the system continuously improve performance and accuracy.

Trend detection. At the same time, the collected user terms can be used as the source for identifying the retrieval patterns of users with the help of text mining. The users' search criteria should be collected to construct the patterns of searching. The possible basis for this function is Google Zeitgeist[1] which pulls together search trends and patterns. It accumulates search statistics based on the millions of searches conducted on Google in a given period of time - weekly, monthly, and annually then produces a summary report. The trends and patterns identified in this stage could be used to guide the users' future search activities. Users' search terms will expose the retrieval habits of the users. For example, users who retrieve early aspects with feature set A also retrieve early aspects with feature set B. This kind of information can potentially provide user-specific intelligence to the retrieval stage.

3.2 Issues

Artifact representation. Creating suitable representations of the artifacts is essential to the overall management process. In this paper, the artifacts are early aspects with their containers – requirements specifications which are text documents in nature. But for some of other artifacts, the representation can be an issue. The accuracy and exactness of the representation may eliminate the need for clustering.

---

[1] http://www.google.com/press/zeitgeist.html

Repository structure. A scalable and credible structure of the software repository will always be the foundation of the whole system and till now it seems that ontologies provide the best choice.

Retrieval technology. Smart retrieval interfaces incorporating term addition and trend detection are good suggestions but there will be trade-offs between simplicity and cost. Also, the space to store the user-specific data will accumulate and could potentially become to the system. Therefore there might be some storage issues as well.

4   Conclusions and Future Work

In this paper, a framework and a series of guidelines for the specific early aspects repository management system with the help of text mining and ontology are provided. Although this is only a high level description of a specific kind of artifact, the benefits are still obvious: to guide the construction and organization of repositories of reusable artifacts which can be easily retrieved. When completed, a unified, standard software repository will greatly assist the domain-specific software system development and reuse process.

Our ongoing work consists of: a further refinement of the framework, an extension of this framework to other kinds of software artifacts, an implementation of this software repositories management system, and, an implementation and further refinement of the guidelines in practice.

Practically, we are working to accomplish the following tasks:

1. Obtain access to the software repositories.
2. Ensure the existence of unified description documents;
3. Ensure the reliability of the collected artifacts;
4. Choose the algorithm to classify the artifacts;
5. The efficiency and accuracy of user-defined terms.

References

1. Li, J.: A survey on reuse: Research fields and challenges (2003) Term paper for DIF8901.
2. Petro, J., Fotta, M., Weisman, D.: Model-based reuse repositories-concepts and experience. In: Proceedings, Seventh International Workshop on Computer-Aided Software Engineering. (July 10-14 1995) 60–69
3. Browne, S., Moore, J.: Reuse library interoperability and the world wide web. In: Proceedings of the 19th International Conference on Software Engineering. (1997) 684–691
4. Hearst, M.:   What is text mining? `http://www.sims.berkeley.edu/~hearst/text-mining.html`, updated Oct. 17, 2003, accessed Sept. 1, 2006.
5. Seeling, C., Becks, A.: Exploiting metadata for ontology-based visual exploration of weakly structured text documents. In: Proceedings of the Seventh International Conference on Information Visualization (IV'03). (2003)

6. Jiang, X., Tan, A.H.: Mining ontological knowledge from domain-specific text documents. In: Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05). (2005)
7. Araújo, J., Baniassad, E., Clements, P., Moreira, A., Rashid, A., Tekinerdoğan, B.: Early aspects: The current landscape. Technical Report CMU-SEI-2005-TN-xxx, Carnegie Mellon University, Pittsburg, PA (2005)
8. Becks, A.: Visual knowledge management with adaptable document maps (2001) GMD research series 15.
9. Murshed, A., Singh, R.: Evaluation and ranking of ontology construction tools. Technical Report DIT-05-013, University of Trento (March 2005)
10. Niles, I., Pease, A.: Towards a standard upper ontology. In: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS'01). (Oct. 2001)
11. Henninger, S.: An evolutionary approach to constructing effective software reuse repositories. ACM Transactions on Software Engineering and Methodology **6**(2) (April 1997) 111–140
12. Curtis, B.: Cognitive issues in reusing software artifacts. In Biggerstaff, T., Perlis, A., eds.: Software Reusability, Vol. 2, Applications and Experience. Addison-Wesley, Reading, Mass. (1989) 269–287
13. Furnas, G., Landauer, T., Gomez, L., Dumais, S.: The vocabulary problem in human-system communication. Communications of the ACM **30**(11) (Nov. 1987) 964–971