ONTOLOGY-BASED QUALITY ATTRIBUTES PREDICTION IN COMPONENT-BASED DEVELOPMENT

Chengpu Li[1], Rob Pooley[2] and Xiaodong Liu[1]

[1]School of Computing, Edinburgh Napier University, Edinburgh, UK
{c.li, x.liu}@napier.ac.uk
[2]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
r.j.pooley@hw.ac.uk

ABSTRACT

Despite the success that Component-Based Development (CBD) has achieved so far, component mismatch remains as a big obstacle for wider and smoother component reuse. Mismatch refers that the selected component does not satisfy the functional requirements, or that it fails the user's expectation in terms of the Quality Attributes (QAs) of the component-based system. This allows us the potential to predict the quality attributes of a software system by analysing the result of component retrieval. In this paper, applicable quality attributes for prediction are selected by investigating existing software quality attributes. A novel ontology-based approach was proposed to achieve precise component retrieval and quality attribute prediction. The approach contains three steps: the first is to develop a Quality Attributes Oriented Component Specification ontology model (QAOCS), where applicable QAs related knowledge of application domains were integrated into the ontology. The second is to establish an ontology-based QAs oriented component retrieval method to retrieve components according to the reuse requirements. The third is to predict the quality attributes of component-based system on the basis of the matching information. Based on these three steps, a prototype tool with an example component repository was built to verify and scale up the approach.

KEYWORDS

System Quality Attributes, Quality Attributes Prediction, Component-based Development, ontology.

1. INTRODUCTION

Component-Based Development (CBD) is an approach to developing a software system by assembling and composing already built software components. Numerous advantages of component-based development have been identified such as shortened development life cycle [7], reduced time-to-market [17][21], and reduced development costs [7][21]. However, component-based development has still not reached its full potential due to a few major hurdles. One of the problems is that CBD focused on the functional aspects of software and omitted the specification and evaluation of Quality Attributes (QA). Attention to quality attributes are usually exercised until the time of integration or testing phases. This results in enormous redesign efforts to fine-tune the software or hardware to meet the quality requirements.

Till now, there have been several approaches available to support reasoning about system quality attributes from component properties [18][8]. They can be employed to support the prediction by building or extending a component model. However, it is not viable in practice if an approach requires huge investment of human and financial resources to accomplish the necessary technical support. Further more, there is no widely accepted method to verify the measurement of the result of prediction. In fact, when the capacity of a component repository is large enough, there will be many functionally similar components available for selection to

satisfy the requirements (functional and non-functional) of an application. Meanwhile, these functionally similar components have their individual varieties of QAs. There is one proven theory: "the better the found components match the users requirements, the higher quality of system composed by these components can be achieved" [9][10]. From this point of view, by analyzing the retrieval of these components, we can source out the most suitable components for CBD, which may satisfy the user's requirements not only in functions but also in QAs. That is, we can predict the quality attributes of a software system by analyzing the result of component retrieval.

To achieve the above objectives, the following four research questions should be considered:

1. In accordance with system quality attributes prediction, which classification of QAs is more accurate? With the classification being defined, which system quality attributes are more likely to be predicted from the component quality attributes?

2. For the most suitable QAs, which factors could be used to determine their levels? And how to use the related factors?

3. Which methods can improve the search precision of the component retrieval, in order to support the system QAs prediction?

4. If a prediction is achieved, how can we verify its validity?

To answer the first question, this paper will compare the existing methods of QA classification, so as to establish a viable method of QA prediction, with which the suitable QAs will be selected according to the four criteria. The details are presented in Section 2.

To answer the second question, a Quality Attribute Oriented Component Specification ontology model is developed to describe the determined factors of QAs. And this ontology will be used to decide the level of the QAs in the proposed ontology-based component retrieval method. As a possible answer, Section 3 proposes a framework to define the process in general. And the Chapter 4 introduces the QAOCS ontology and its construction method.

To answer the third question, Section 5 presents the quality attributes oriented component retrieval method and how this method supports the system QA prediction.

For the last question, a case study is given to illustrate how the prediction of QAs is achieved. And then a questionnaire and an empirical method are used for the validation. The details are shown in Section 6 and 7.

With the foregoing questions answered, conclusion and further work are mentioned in Section 8.

2. ANALYSIS OF QUALITY ATTRIBUTES

A system has quality attributes that emerge from the combination of its parts. The qualities are properties or characteristics of the system that its stakeholders care about and hence will affect their degree of satisfaction with the system [1][2][26]. The quality of a software system can be assessed by a number of quality attributes. Many of these QAs are considered to be systemic, i.e., they are applicable to the entire software system or they are spanning across parts of it. What constitutes the importance of quality attributes is dependent on the stakeholder set perspective. For instance, while an end-user may desire performance and usability, the development management may want a high degree of maintainability and reusability.

A great number of quality attributes are encountered in software engineering. They are classified in many different ways, frequently in a non-orthogonal manner. The first example of classification [15] is related to the system lifecycle: run-time properties (visible and measurable during the program execution) and lifecycle properties (those that characterize different phases in a development and maintenance process). The second example is the quality model defined in ISO/EIC 9126-1 "Software engineering product quality" standard [16], which classifies quality attributes as external and internal. Quality attributes that refer to the internal quality are typically applied to intermediate deliverables at certain development stages (e.g. attributes of a design specification, source code, etc.). Internal therefore has the connotation of "development internal view". The relation between internal and external quality attributes is not unambiguous though; an internal quality attribute may have impact on different external quality attributes and of course an external quality attribute is a result of combination of internal attributes. The third one [6] is related to composability. The attributes are classified according to the principles applied in deriving the system attributes from the attributes of the components involved. They distinguish the following types of attributes: i) Directly composable attributes, which is a function of, and only of, the same type of attribute of the components involved; ii) Architecture-related attributes, which is a function of the same type of attribute of the components and of the software architecture; iii) Derived attributes, which depends on several different attributes of the components; iv) Usage-depended attributes, which is determined by its usage profile; v) System environment context attributes, which is determined by other attributes and by the state of the system environment.

The above existing quality attributes classifications are not helpful for the quality attributes prediction on the basis of users satisfactions of the component retrieval results. The satisfactions here are decided by the searching precision between of users' QA requirements and the result components. So we proposed a classification method base on functional and non-functional requirements [2][9]. This classification helps identify whether a requirement will affect the functionality of the system (functional) or whether it will constrain the system (non-functional). It is probably the most beneficial in that it helps define what system functions are being considered. It is known that the definition of quality attributes is put forward as in [2]: the quality attributes are often called properties or non-functional or extra-functional attributes because they describe something about the quality of the component and not explicitly about the component functionality.

It means that functionality cannot be identified as a quality attributes. Nevertheless, we consider functionality as one special sort of QAs in this paper and refer it as the Functional Attributes. Other QAs are thus referred to as Non-functional Attributes. Such arrangement is necessary due to the fact that the systems QAs are predicted in the proposed approach via analyzing the results of component retrieval. While the functional requirements are at the core of the user's requirements, it would not be valid to satisfy the non-functional requirements only without due consideration to the functional requirements in the search.

Functional attributes describe all kinds of functions performed by a component-based system. They are related to functional requirements provided by the user. The functional attribute refers to functionality, which is the capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

Non-functional attributes are affected by the non-functional users requirements. It is usually some form of constraint or restriction that must be considered when designing the component-based system. The non-functional attributes of systems have several sub attributes including security, availability, portability, maintainability, integratability, performance, reliability, usability and cost. The definition of each attributes is as follows [15][16]:

Availability
Definition: the measure of time that the system is up and running correctly; the length of time between failures and the length of time needed to resume operation after a failure.

Portability
Definition: the ability of a system to run under different computing environments. The environment types can be either hardware or software, but is usually a combination of the two.

Maintainability
Definition: the capability of the software product to be modified. Modifications may include corrections, improvements or adaptations of the software to changes in the environment and in the requirements and functional specifications (the effort needed to be modified)

Integratability
Definition: the ability to make the separately developed components of the system work correctly together.

Performance
Definition: the response time, utilization, and throughput behaviour of the system. Not to be confused with human performance or system delivery time.

Reliability
Definition: the continuity for correct service.

Usability
Definition: the ease of use and of training the end users of the system.

Some of above quality attributes can not be predicted during the component retrieval process in accordance with the following reasons [18]: Some quality attributes could not be derived directly from the component properties and might require a complex model, related to the component model and the system architecture; Some quality attributes do not exist at the component level and might be the result of a complex combination of the system interaction with its environment, system architecture and component model. In our research, we focus on functionality, availability, portability, maintainability and integratability, which comply with the above criteria.

3. PROJECT FRAMEWORK

After identifying the suitable quality attributes, a Quality Attribute Oriented Component Specification ontology model (QAOCS) is built first to describe the factors, which affect the value of QAs. The QAOCS supports quality attributes prediction by improving the precision of component retrieval. And then an algorithm is set up to calculate the precision between the users QA requirements (functional and non functional) and search result. The precision will be further used as the basic data to predict the component-based system quality attributes. To address the whole approach, a project framework is established as shown in figure 1. The framework consists of four key parts, namely query requirement collection, component ontology construction, component retrieval and system quality attribute prediction.

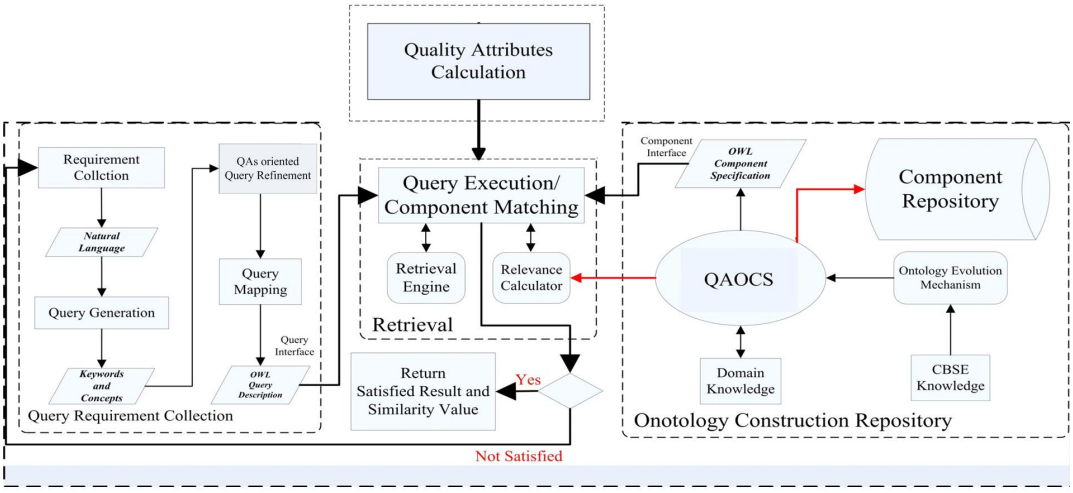


Figure 1. A framework for ontology-based component specification and selection

3.1 Query Requirement Collection

To retrieve a component with the ontology-based approach for the prediction, the first step is to collect the QAs requirement correctly and then to represent it with the ontology semantics. This task can be accomplished in two steps: initial query generation and QAs oriented query refinement.

Initial Query Generation:
The user specifies the requirements for the necessary components in a natural language which employs imperative or nominal sentences. A heuristic-based approach [5] will be used to identify keywords and concepts expressed by the user and to generate an initial query. Subsequently related terms (synonyms) of the keywords and concepts are also identified for query expansion.

QAs oriented Query Refinement:
The keywords and concepts identified in the previous step are mapped against the QAOCS ontology model to ensure that correct terms (keywords and concepts) for each QAs are used in the query. And then all the terms involved will be expressed in OWL, which is favoured here in terms of its capacity in the description of semantics. In this query refinement step, query requirements in natural languages are translated into OWL format without misunderstanding finally.

3.2 Ontology Model Construction

The Quality Attribute Oriented Component Specification ontology model (QAOCS) has a key role in the approach. It is the computer-recognisable ontological representation of the semantics of component specification. The QAOCS is developed on the basis of CBSE knowledge and application domain knowledge. It is managed by the ontology evolution mechanism.

A component repository will be built based on the QAOCS. Available components were specified according to QAOCS and populate the repository. The QAOCS and component

specification were defined in an extended version of OWL. A component specification creation mechanism has been developed to facilitate the specification process.

3.3 Component Retrieval

Having the above two phases in place, the component retrieval is thus regarded as the matching between the user QAs oriented query and the ontological component specification. Both of these two parts are described in the OWL. The matching was identified automatically. The retrieval engine returns detailed search results, including not only the matching components but also their precision. The precision gauges the level of relevance of the found components with a precision calculation algorithm which is described in following section.

3.4 System Quality Attributes Prediction

The precision obtained in the component retrieval part will be used as basic data to predict the system quality attributes. As we mentioned in section 2, the quality attributes we refer to are functionality, availability, portability, maintainability and integratability. In this part, the search result components and their precision will be transformed to the degree of every quality attribute by the quality attributes prediction algorithm.

Query requirement collection, ontology model construction, component retrieval and system quality attributes prediction are four main parts of this project. The query requirement collection part is based on the Natural Language Processing and Artificial Intelligence areas, so it is not the focus of this project. In this research, we paid more attentions to QAOCS, precision calculation algorithm and quality attribute prediction algorithm. The details of these three parts will be introduced in the following sections.

4. QUALITY ATTRIBUTE ORIENTED COMPONENT SPECIFICATION ONTOLOGY MODEL

The Quality Attribute Oriented Component Specification ontology model (QAOCS) is a component specification ontology, which is used to represent the component specification related to system quality attributes functionality, availability, portability, maintainability and integratability. It is built especially for component retrieval and quality attributes prediction. The QAOCS is the core of this approach, and is developed on the basis of CBSE knowledge and application domain knowledge. The range of its classes is depending on the content of components located in the repository. The QAOCS is the foundation of ontology-based component retrieval and the subsequently quality attribute prediction.

QAOCS has tree type architecture of the following three aspects, the function aspect, the basic specification aspect and the environment aspect. Each of them can be seen as a sub-ontology to describe one aspect of the component specification. The function aspect corresponds to quality attributes functionality, the basic specification aspect refers to availability and integratability, and the environment aspect refers to attributes portability. Among these three aspects, some adaptive information is covered, which is used to predict the attributes maintainability. The three sub ontologies are built following the Component Description Schema. In the following sections, we introduce the Component Description Schema first, and then we talk about how to build each aspect of QAOCS on the basis of the component specification taxonomy schema.

4.1 Component Specification Taxonomy Schema

Here we present the common content of a component specification in the format of three schemas, as shown in Figure 2. Each schema is related to one quality attribute. The schema illustrates the abstraction hierarchy of component specifications which affects the attribute.

Clearly the schema is not an ontology and the semantics of its content are not recognisable by a computer. However it shows what information a component specification should cover; hence it provides input to the development of the more advanced QAOCS.

The functionality taxonomy includes the function of the component and its application domain. It can be further classified into specific component functions as shown in figure 2 a). It can serve as an effective way of searching for an appropriate component. The classification of a component may include graph, communication, mathematics, database, UI, system, driver, adapter, container, business logic, agent procedure, collaborate, and word process, etc.

Availability refers to the only attribute component evaluation. The component evaluation is provided by the component vender, from the company test and previous user feedback.

Portability is affected in connection with the software implementation environment. It could be further classified into more specific categories such as operating system, compiler, database, and others (figure 2 b)).

Maintainability is related to component adaptation information. Component adaptation is a popular means to alter the functionality and quality features of selected components [3][4]. Some components, whose function and QoS may vary via the application of adaptation. So the information of adaptation assets can be used to judged the maintainability of composed system.

Integratability refers to the information of the component type and component model. Component type contains deployed form and builds technology, as shown in figure 2 c). Deployed form is also an important attribute of a reusable component. It could be used to represent the deployed form of a component. The forms are listed as follows: DLL, EXE, source code, graph and so on. Build technology refers to the technology, such as COM/DCOM, ActiveX, .NET, VCL/CLX, Java bean, EJB, CORBA, etc., which is used to build the component.

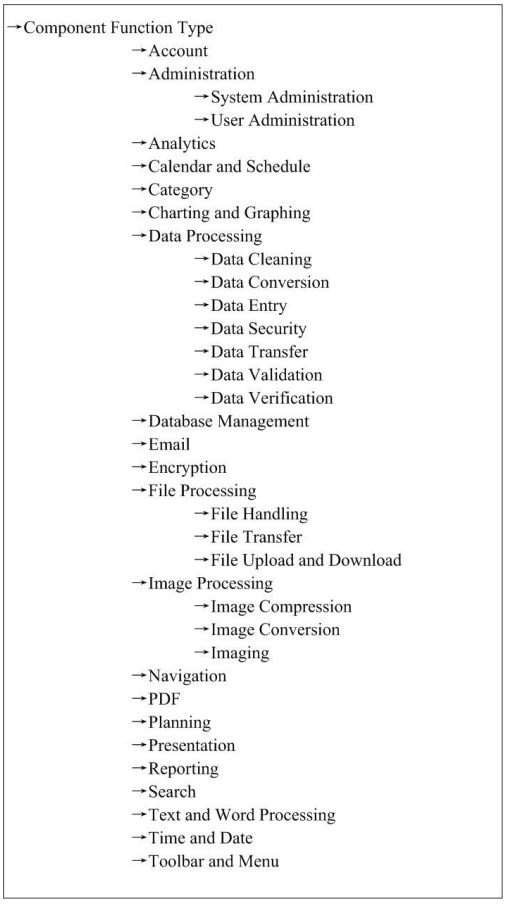
4.2 The Translation of Component Specification Schema to QAOCS

This translation has two steps: First, we put the top level attributes related to the five quality attributes as classes, and then their sub-attributes, sub-sub-attributes, and so forth. Second step, for these classes and their sub classes, we connect classes with isA relationships.

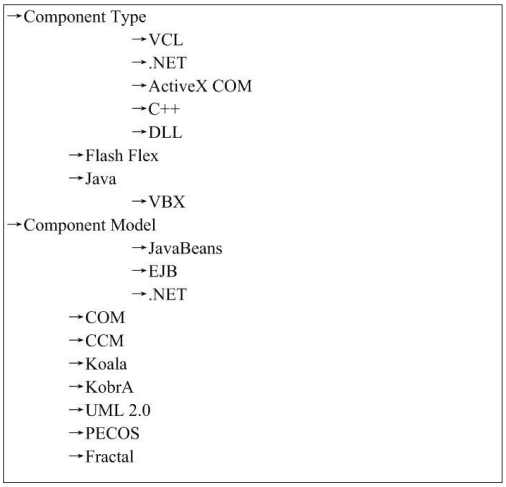
In order to support the search engine to search the more suitable components, some basic information of component specifications should to add into QAOCS, such as component name, component version, component vender and hardware and software requirements. However, if we build a large ontology to cover all relevant information, it has drawbacks, since it is monolithic, which incurs too many complications in the ontological component specification and makes it difficult to understand and use. It is also very difficult to update and manage. We therefore divide this comprehensive component ontology into three aspects.

4.3 QAOCS Sub-aspects

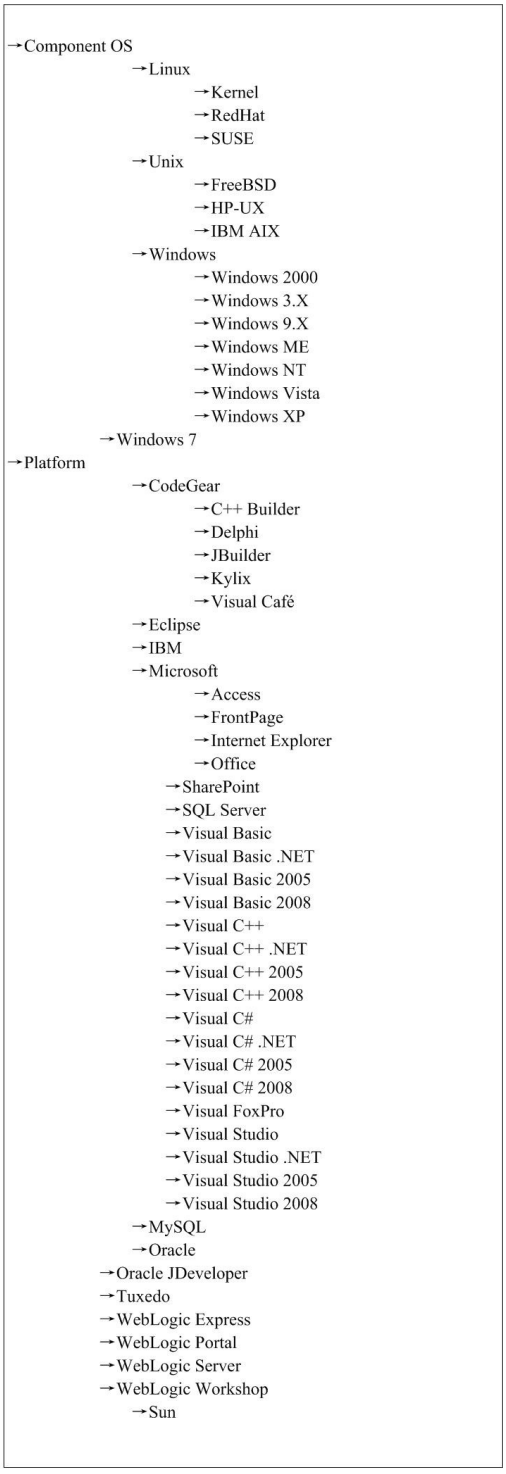
QAOCS is a component specification ontology for the quality attributes prediction. Its primary action is to describe all the attributes of a component which are related with selected system quality attributes. In addition, other component specifications which are useful for the component search are also covered. It has three aspect including basic specification aspect, environment aspect and function aspect.



a) *Functionality Related Component Specification Schema*



c) *Integratability Related Component Specification Schema*



b) *Portability Related Component Specification Schema*

Figure 2: Component Specification Schema

4.3.1 Basic Specification Aspect

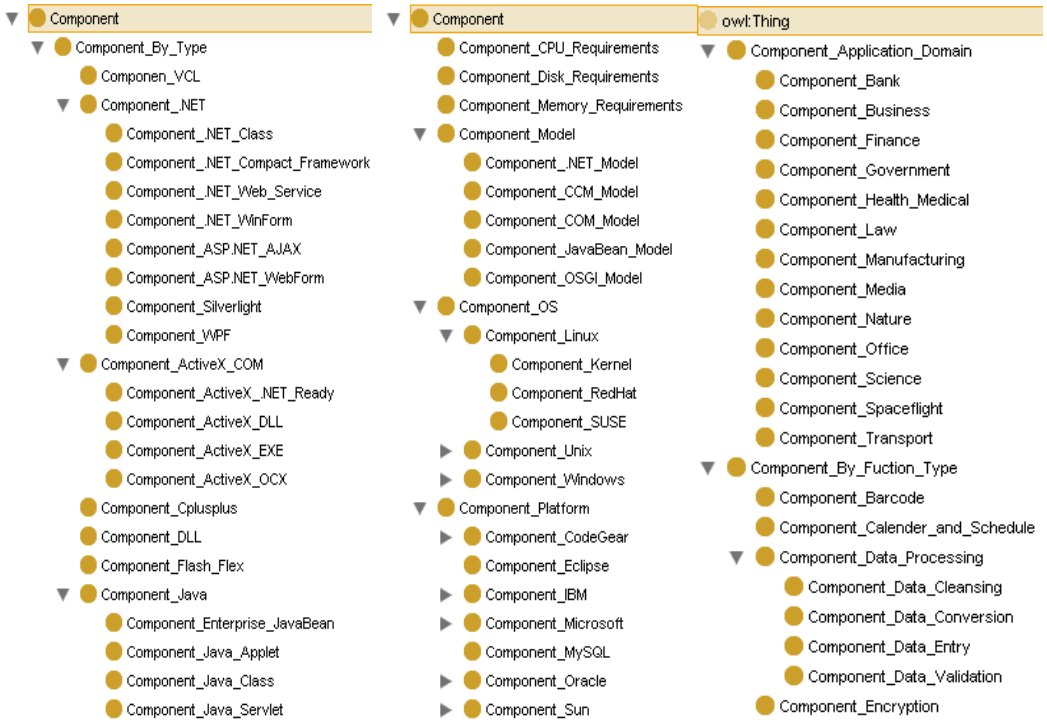
Basic specification aspect describes the basic information of a component and its related adaptation information, such as its name, version, vender, type and so on. The classes of this model are shown in figure 3 a). The classes, its subclasses, sub-subclasses and so forth as viewed are edited in the ontology tool Protégé. The basic specification aspect is used to predict the attribute availability, integratability and parts of maintainability.

4.3.2 Environment Aspect

Environment aspect is used to represent the reuse context information of the components and its related adaptation information, including but not limited to the application environment, hardware and software platform, required resources and dependency on other components, if any. Figure 3 b) shows the top level classes of this model, its subclasses, sub-subclasses and so forth, again using the ontology tool Protégé. The information included in the environment aspect is applied to attribute portability, integratability and parts of maintainability.

4.3.3 Function Aspect

Function aspect refers to the functionality and its related adaptation information. It presents component function technology and relevant information. The classes of this model are shown in c) of figure 3. We should note that the class Function AppDomain is used to connect the other domain ontology to QAOCS.



a) b) c)

Figure 3: Sub levels classes of the three aspects in QAOCS

4.4 OWL Format

As mentioned above, we use OWL [12] to record all ontology information for ontology-based component search, and protégé is used to edit QAOCS. Protégé is being developed at Stanford University in collaboration with the University of Manchester. It is a free, open source ontology editor and a knowledge acquisition system. Like Eclipse, Protégé is a framework for which various other projects suggest plugins. This application is written in Java and heavily uses Swing to create the rather complex user interface. Protégé recently has over 100,000 registered users.

Our specification in three sub-aspects of QAOCS is translated to OWL, and the QAOCS information is stored as OWL documents. The following section is an example to show one part of component type attributes, their subclasses and instances in OWL format.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/Ontology1230334782.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1230334782.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Java">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="ByType"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Component"/>
  <owl:Class rdf:ID="JavaClass">
    <rdfs:subClassOf rdf:resource="#Java"/>
  </owl:Class>
  <owl:Class rdf:about="#ByType">
    <rdfs:subClassOf rdf:resource="#Component"/>
  </owl:Class>
  <JavaClass rdf:ID="CryptoXpress_SDK"/>
  <JavaClass rdf:ID="pinUpload"/>
  <JavaClass rdf:ID="DEXTUploadJ"/>
  <JavaClass rdf:ID="CryptoXpress_CF"/>
  <JavaClass rdf:ID="CryptoXpress_LT"/>
</rdf:RDF>
<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430)  http://protege.stanford.edu -->
```

5. QUALITY ATTRIBUTES PREDICTION METHOD

The prediction method estimates the QAs of composed system by analyzing the matching result of factors that impact on the component QAs. Different from the exiting system QAs prediction methods, our approach can improve the composed system QAs by selecting the more suitable components. From this point of the view, the QAs of the composed system can be mapped by calculating the precision between QAs oriented requirements and result components.

As mentioned in the section 3, the QAOCS model supports both the QAs oriented requirements refinement and component registration. It builds a bridge between the user requirements and the components in the repository. Based on QAOCS, a search precision calculation method is developed by counting the search paths of the result components. Subsequently the search precision will be further decomposed as the value of each QA.

As the foundation of the search precision calculation method, in each aspect of QAOCS, we give every class a weight on the basis of component-based software engineering knowledge and user feedback. The class weight assignment algorithm are defined in our foregoing ontology-based component search project [19][20]. The rules of weight assignment are:

(1) In one aspect, the lower the layer is, the heavier the class weight;

(2) In different aspects, the same layer function aspect class is heavier than basic specification aspect class and then environment aspect class.

To summarise, Let N represent the occurring times of the keywords in a facet, the subscript f, b, e indicate which facet is related, namely function, basic Specification and environment. the weight assignment rules are defined with the following formulae:

Weight of function aspect class (Wfc):

$W_{fc}= (1+ \frac{N_f}{N_f + N_b + N_e})^n$ (n is the level of the layer in which the class is located)

Weight of basic specification aspect class (W_{bc}):

$W_{bc}= (1+ \frac{N_b}{N_f + N_b + N_e})^n$ (n is the level of the layer in which the class is located)

Weight of environment aspect class (W_{ec}):

$W_{ec}= (1+ \frac{N_e}{N_f + N_b + N_e})^n$ (n is the level of the layer in which the class is located)

When a user inputs requirements by groups of keywords, the keywords are searched one by one in the QAOCS model. Meanwhile, the search path of each keyword is recorded from result class to top class. And then the weight of the search path is calculated by summing up every class weight in this path. The component related to the result class will be indentified as the result component. If a result component has n paths connect to QAs requirements, the precision will be total weight of the n paths.

The search precision of each result component is further decomposed to the values of five QAs, including functionality, availability, portability, maintainability and integratability. The prediction rate of functionality is the sum of precision of the search paths which are obtained by matching with the functionality related classes in the function aspect. The method to calculate prediction rate of portability, integratability and maintainability are the same as the way to functionality. The only difference is portability and integratability related class are located in the different aspects, and maintainability is determined by the classes “adaptation information” all over the three aspects. The method to predict rate of availability is different from other QAs. It is calculated on the basis of the class “component evaluation” in the basic specification aspect. All the components populated in the repository have a specification which contains the component evaluation. The component evaluation is provided by the component vender, from the company test and previous user feedback. The component evaluation is a specific value which indicates the predicted rate of the availability directly.

6. CASE STUDY

A quality attributes oriented component search prototype tool with an example component repository has been developed, as a means of automating the proposed approach. To evaluate both the approach and the prototype tool, a case study of quality attributes prediction has been done with a number of use scenarios of component search.

6.1 The Prototype Tool

In QAOCS prototype tool, the QAOCS ontology model is translated into OWL format. The information of the functional aspect, the basic specification aspect and the environment aspect was stored in three different files. And each class from top to bottom in these three aspects is given a weight for calculating search precision. Two hundred components (actually component specifications) are selected from open source component libraries, e.g., Componentsources, Componentplanet and Allfreeware, and are used to populate a corresponding repository.

The prototype tool has a sample user interface as shown in figure 4. On the top left, it's a text area for the user to fill in search keywords. On the bottom left, these is a column of option buttons, it lists the available five quality attributes, which can be predicted during the search. If user would like one or more QAs to predict, just selects the appropriate option before the search. Otherwise the tool will reason about all the QAs for each result component. On the right hand side of the UI is a black panel for showing the search results.

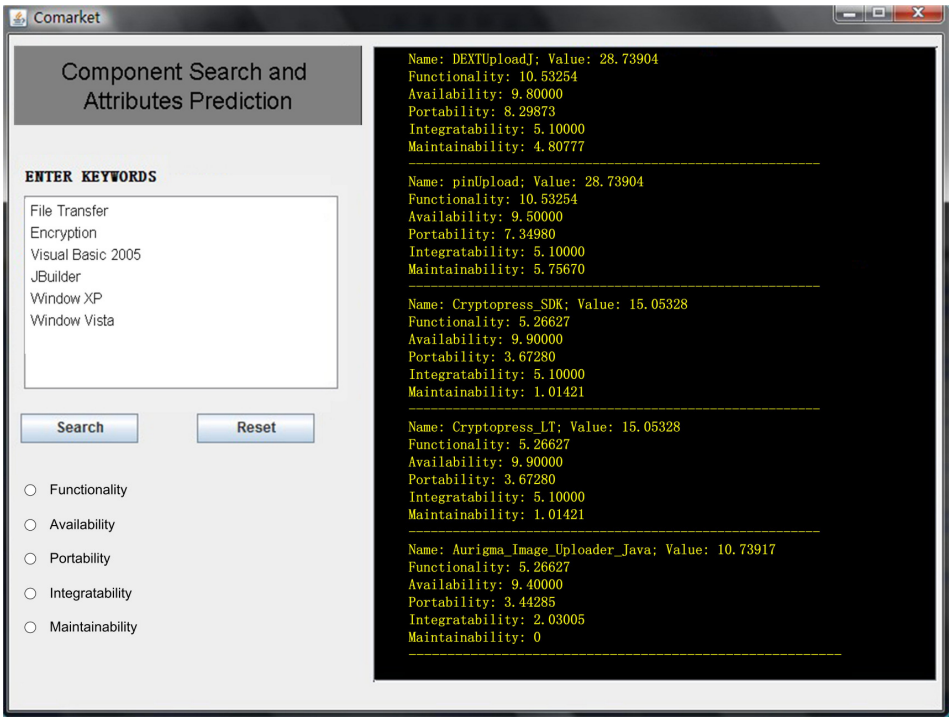


Figure 4: UI of the prototype tool

6.2 Case study

To facilitate users exemplify the prototype tool, several search scenarios with corresponding search results are offered. Here we take a scenario of developing an encrypted file transfer systems with user-friendly interface to illustrate how the tool works for predicting and comparing the QAs of result component.

Table 1 Keywords of the search

Keywords Type:	Keywords:
Function	*File Transfer,* encryption
Component Platform	*Windows XP, Windows Vista*
Component Container	*Visual Basic2005 , JBuilder*

After the user query refinement, the scenario is specified as requirements in table 1. User fills in the keywords into the text area and leave the quality attributes option button blank for receiving all the QAs information. The components relevant to the matched classes are identified as the result components for the search, and the precision for each result component is calculated on the basis of their searched paths. And then the precision are separated into the values of each quality attributes. Take functionality as an example, two functionality related search path are obtained by searching the keywords "*File Transfer*" and "*Encryption*". Meanwhile, five components are found from the repository (as shown in figure 4). For the component "DEXTUploadJ" and "pinUpload", they match with all the functionality related keywords, their prediction value of functionality is the sum of path weight "*File Transfer*" and "*Encryption*". For the others result components, they only match with one of them. So the prediction value of functionality is the weight of the matched path. The way to calculate others QAs are same as the functionality. Finally, the search results list on the black panel from the precision values high to low as shown in figure 4. When a certain QA option button are selected, the result component has the highest value of this QA will be list on the top.

7. VALIDATION

Two methods are used to test the validity of the approach, statistic testing and application testing. The statistic testing is intended for more users to use prototype tool and traditional key words search tool to search component in the process of CBD, respectively. Users can opt for the scenarios either provided by us or established by their own for their testing, and compare the result components to experience the facility of the QAs information made available in our new approach in the later component composition. The application testing is intended to test its functionality and practicability in the actual applications of software companies, on basis of future cooperation with a few software companies.

7.1 Statistic Testing

The following testing procedures are recommended: i) to understand the component specifications from its introduction; ii) to select a scenario from the given list or to set up a scenario from scratch; iii) to search with the SQL database search tool (traditional approach) and record the results (R1); iv) to search with the QAOCS prototype tool according to the same requirements and return another set of results (R2), including their relevant five quality attributes; v) Comparing R1with R2; vi) to fill in the given a questionnaire with comments.

The last two procedures are crucial for the testing. The comparison between R1 and R2 made it clear that R1 is obtained through the matching of keywords and result components in syntax, which results in low search precision and gives no consideration to the impact of the result component on the QAs of composed system. Software developers can only make the selection on basis of their experiences in the specifications of result components. When the samples of result component become larger and larger, time used for selection will increase substantially as well as the probability of error. On the contrary, R2 is achieved through the matching of keywords and result components in semantics, which helps avoid the mismatch in the search process and satisfy the QAs requirements. Software developer could easily identify the required

components in accordance with the sequential order of the QAs from high to low. A better precision and high effectiveness are thus in place.

The questionnaire is also very important as it helps us gain further understanding of the users in terms of their professional background, their satisfaction with the result components and the attributes prediction rate provided by the QAOCS search tool, their opinion in whether our precision index indicated the similarity between your query and the result components, their satisfaction with the functionality, availability, portability, maintainability and integratability provided by the QAOCS search tool, and their input in uncovered areas. All these will provide more evidence on the validity of our approach and give the chance for further improvement.

So far, 30 users tested the tool in practice. The values of prediction for the five quality attributes are shown in the figure 5. It can be seen that the maintainability achieved the best satisfaction, because its accuracy is just related to one class adaptation information in the QAOCS, which is normally clearly specified in the component specification. The functionality, portability, and integratability also are considered as highly satisfactory, because they are related to more classes in the ontology. While more information is available, more accurate the quality attributes are predicted. The availability has the lowest satisfaction due to the least information of the component evaluation.

The assessment of search performance had been done in the foregoing project [20], as shown in figure 6. The results were assessed according the criteria of Recall (R) and Precision (P).

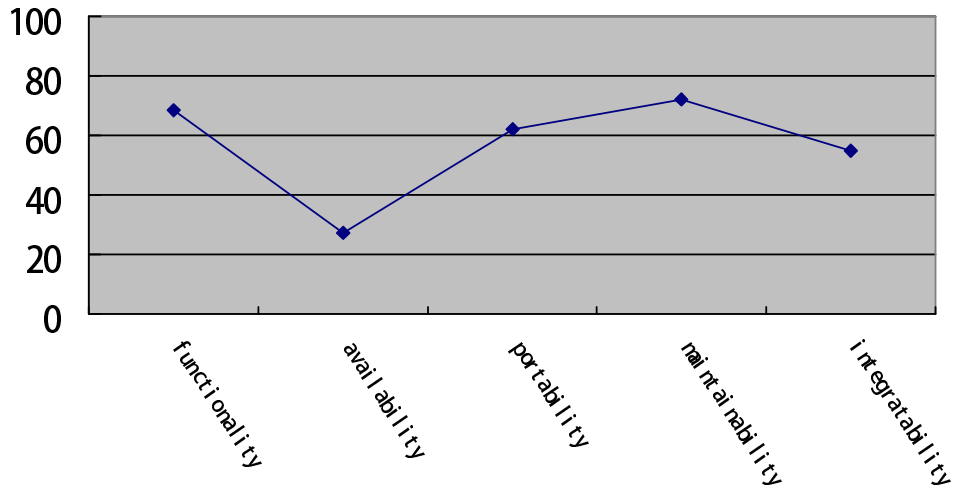


Figure 5. The degree of satisfaction of quality attributes prediction

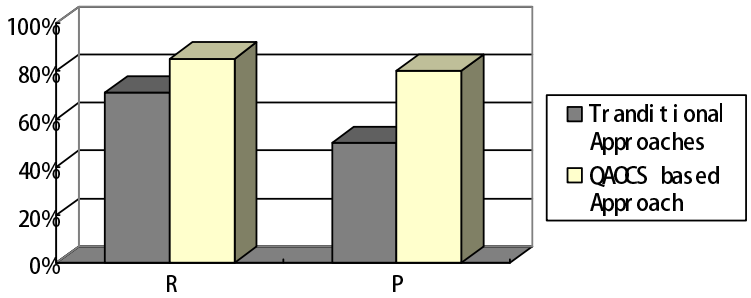


Figure 6. The level of satisfaction of QAOCS prototype tool and traditional search tools

7.2 Application Testing

To test the approach in the application environment, we cooperated with UFIDA Software Co., Ltd [13] in its application of T6 Small and Medium Enterprises (SMEs) Management Software for Opening clothing Co., Ltd [14]. UFIDA is a major provider of management software solutions and e-business services and its T6 SMEs is developed in CBD. The main features of T6 SMEs comprise Production Management (Production Data, Material Requirements Planning Management, MRP, Manufacturing Orders and Progress Management), Supply Chain Management (Sales Management, Purchasing Management, Outsourcing Management, Inventory Management, Inventory Accounting, Financial Management) and Account Management (Receivables Management, Payables Management, Cost Management).

In the process of integrating T6 with existing software of the clothing company, an incompatible problem occurred due to different development platforms. As some components (PM java edition and OM java edition) in the T6 are of low level portability and integratability, the current integration and later update will require the recoding of relevant components. In fact, there are some components with high portability and integratability available in the repository of UFIDA, which can implement both in java platform and .NET platform. However without due consideration the keyword search method did not identify the components of better precision and more clear description, the high level QAs component PM+ and OM+ were not selected. With our ontology-based QAs oriented approach, software developers can get access to the components of higher portability and integratability with least efforts. This example evidenced the viability and effectiveness of our approach in the application environment, though it needs time to construct the QAOCS and to register components.

8. RELATED WORK

This chapter introduces selected related work to the QAOCS project. In the area of QAs prediction in CBD, the research involves three main issues, which include available QAs selection, prediction mechanism and validation method. In the existing projects, PECT and Eskenazi are systematic, involving all the three issues. The PECT project was conducted by SEI, which focus on investigating predictable assembly from certifiable components [18]. This project first analyzes the possibility to develop component technologies that provide prediction mechanisms for quality attributes of assemblies, given quality attributes of the components. And then, a method was built that can be used to build prediction enabled component technologies and its corresponding validation procedure. Difference with PECT, Eskenazi's [8] project refers to software architects, which can be utilized to estimate the certain QAs. Furthermore, it provides software architects with practical and feasible approaches to be applied in an industrial context.

In addition, several representative work just focus on one or two above issues. Schmidt's project [22] analyzes the possibility of using component technologies to build or support prediction. Shepard [25] proposes a model to provide software system QAs estimates from the QAs of software components and their usage profiles. Wohlin [29] develops component models as well, and his work further targets how the models certify software components. These projects test functionality of models when building a certification framework for prediction theories or prediction enabled component technologies.

Currently, most work focus on certain specific attributes prediction. Reliability is addressed in [22][23][24], which use parameterized contractual specifications on the basis of state machines. An e-commerce example is taken to verify the effect and an experiments report is used to record the result. Other research of reliability prediction in CBD includes Stafford and McGregor's project [27]. This work targets on investigating how to use the software reliability theories to component based software. Scalability and performance prediction for component-based systems is addressed in [30], where several COTS middleware products are conducted as the object by an empirical method. This research presents scalability metrics depending on the performance of the system. The prediction of Memory consumption and the suitability are presented by using the Koala component model [28]. The Koala is used to experiment with prediction of the memory demand from component compositions. Static memory evaluation techniques are used and a method is proposed that allows estimating the memory consumption.

9. CONCLUSIONS AND FURTHER DIRECTIONS

The objectives of the approach are to develop an ontology-based approach to predict quality attributes for CBD by analyzing precision of component retrieval. At the same time, it eliminates the component mismatch problem via ontology-based retrieval. The approach has its unique contribution to the current art of state of quality attributes prediction in the following parts: i) an ontology-based framework for QA oriented component specification and prediction are proposed; ii) a Quality Attribute Oriented Component Specification Ontology Model (QAOCS) are built for describing the factors that impact on the QAs; iii) a system quality attributes prediction method was provided to calculate the value of the functionality, portability, availability, maintainability and integratibility by composing the search precision. iv) an empirical method is used for the approach validation.

As the discussions above, it is concluded that the approach has good effect on attributes prediction and component search. The resulting tool implements the approach and is consistent with the approach. Some possible extensions will be explored of the present work in the future: 1) on the basis of testing results and users feedback, the weight assignment will be refined to get closer to practical applications and achieve wider accepted. ii) The retrieval algorithm will be optimized to response the refinement both in QAOCS and precision calculate method. iii) An intensive system quality attributes research and test will be carried out. More available and strict quality attributes will be added to the approach.

ACKNOWLEDGEMENTS

I am heartily thankful to my supervisors, Prof. Rob Pooley and Dr. Xiaodong Liu, whose encouragement, guidance and support enabled me to finish the project and write up this paper.

REFERENCES

[1] Alvaro, A., Almeida, S. and Meira, L. (2005) "Component Quality Information Provided by Software Component Markets and a Brazilian Software Factory", Submitted to *the 5th International Conference on Quality Software (QSIC)*.

[2] Bertoa, M and Vallecillo, A. (2002) "Quality Attributes for COTS Components", In the Proceedings of *the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE).*

[3] Bosch, J. (1999) "Superimposition: A Component Adaptation Technique", *Information and Software Technology*, Vol.41 (5).

[4] Bracciali, A., Brogi, A. and Canal, C. (2005) "A formal approach to component adaptation", *Journal of Systems and Software*, Vol 74, Iss 1, pp(s): 45-54.

[5] Ceria, S., Nobili, P. and Sassano, A. (2005) "A Lagrangian-based heuristic for large-scale set covering problems", *Mathematical Programming* Vol 81, No 2.

[6] Crnkovic, I., larsson, M. and Preiss, O (2005) "Concerning Predictability in Dependable Component-Based Systems: Classification of Quality attributes", *Architecting dependable systems III, LNCS* 3549, pp. 257-278.

[7] Due, R. (2000) "The Economics of Component-Based Development", *Information Systems Management*, Vol. 17, No. 1, pp. 92-95.

[8] Eskenazi, E. and Fioukov, A. (2004) "Quantitative prediction of quality attributes for component based software architectures", *doctoral thesis,* Technische Universititeit Eindhoven.

[9] Goulao, M. and Brito, F. (2002) "The Quest for Software Components Quality", In the Proceedings of *the 26th IEEE Annual International Computer Software and Applications Conference (COMPSAC)*, England, pp. 313-318.

[10] Heineman, G. and Councill, W. (2001) "Component-Based Software Engineering", *Addison-Wesley*.

[11] http://www.softwarearchitectures.com/go/Discipline/DesigningArchitecture/QualityAttributes/tabid 64/default.aspx. SoftwareArchitectures.com.

[12] http://en.wikipedia.org/wiki/Web Ontology Language

[13] http://www.ufida.com/

[14] http://www.ouwang.com/

[15] ISO 9126 http://en.wikipedia.org/wiki/ISO_9126

[16] ISO/IEC, "Software engineering – Product quality – Part 1: Quality model",ISO/IEC, International Standerd 9126-1:2001

[17] Kim, Y. and Stohr, E.A. (1998) "Software Reuse: Survey and Research Directions", *Journal of Management Information Systems*, Vol. 14, No. 4, pp. 113-147.

[18] Larsson, M. (2004) "Predicting Quality Attributes in Component-based Software Systems", *PhD Thesis*, Mälardalen University.

[19] Li, C., Liu, X.and Kennedy, J. (2010) "Achieve Semantic-based Precise Component Selection via an Ontology Model Interlinking Application Domain and MVICS", In *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE'10).*

[20] Li, C., Liu, X.and Kennedy, J. (2009) "A Multiple Viewed Interrelated Ontology Model for Holistic Component Specification and Retrieval" In *Springer-Verlag's LNCS*, pp. 50-69.

[21] Patrizio, A. (2000) "The new developer portals", *Information week*, No. 799, Aug 14, pp. 81-86.

[22] Reussner, R.H., Schmidt, H. and Poernomo, I. H. (2003) "Reliability Prediction for Component-based Software Architectures", In *Journal of Systems & Software*, Vol 66, Iss 3, pp. 241-252.

[23] Schmidt, H. (2003) "Trustworthy components: compositionality and prediction", In *Journal of Systems & Software*, Vol 65, Iss 3, pp. 215-225.

[24] Schmidt, H. and Reussner, R. H. (2001) "Parametrized Comtracts and Adapter Synthesis", In Proceedings of *5th ICSE workshop on CBSE*.

[25] Shepard, T. and Dolbec, J. (1995) "A component based reliability model", In Proceedings of *Conference of the centre for advanced studies on collaborative research (CASCON), ACM*.

[26] Simao, S. and Belchior, A. (2003) "Quality Characteristics for Software Components: Hierarchy and Quality Guides", *Component-Based Software Quality: Methods and Techniques, Lecture Notes in Computer Science (LNCS) Springer-Verlag*, Vol. 2693, pp. 188-211.

[27] Stafford, J. and McGregor, J. (2002) "Issues in Predicting the Reliability of Composed Components", In Proceedings of *5th workshop on component based software engineering*.

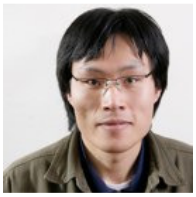
[28] Van Ommering, R., (1998) "The Koala Component Model", in *Building Reliable Component-Based Software Systems", ISBN 1-58053-327-2, Artech House.6*, pp. 53-59.

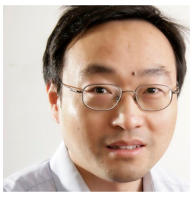
[29] Wohlin, C. and Runeson, P. (1994) "Certification of Software Components", In *IEEE Transaction on Software Engineering*, Vol 20, Iss 6, pp. 494-500.

[30] Yan, L., Gorton, I., Liu, A. and Chen, S. (2002) "Evaluating the scalability of enterprise javabeans technology", In Proceedings of *9th Asia-Pacific Software Engineering Conference*, pp. 74-83.

Authors

Chengpu Li is a third year PhD researcher in centre for information and software systems of Edinburgh Napier University. His research interests include Software Reuse, Component-based Software Engineering and Service-oriented Systems.



Rob Pooley is a Professor of Computer Science at Heriot-Watt University, Edinburgh. His research interests include Performance Engineering, Software Engineering and ad hoc and pervasive networking. He is the author of some one hundred papers in these and other areas, as well as two textbooks on UML and Software Engineering.

Xiaodong Liu received his PhD in Computer Science from De Montfort University, England. He is an academic member (Reader) of the School of Computing, Edinburgh Napier University. As an active researcher, his current research focuses on software reuse, software evolution, component-based development and service-based systems. He has published over 50 referred papers and book chapters, and led several externally funded projects.