A Holistic Semantic Based Approach to Component Specification and Retrieval

Chengpu Li

A thesis submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Edinburgh Napier University
School of Computing

July 2012

Abstract

Component-Based Development (CBD) has been broadly used in software development as it enhances the productivity and reduces the costs and risks involved in systems development. It has become a well-understood and widely used technology for developing not only large enterprise applications, but also a whole spectrum of software applications, as it offers fast and flexible development. However, driven by the continuous expansions of software applications, the increase in component varieties and sizes and the evolution from local to global component repositories, the so-called component mismatch problem has become an even more severe hurdle for component specification and retrieval. This problem not only prevents CBD from reaching its full potential, but also hinders the acceptance of many existing component repository.

To overcome the above problem, existing approaches engaged a variety of technologies to support better component specification and retrieval. The existing approaches range from the early syntax-based (traditional) approaches to the recent semantic-based approaches. Although the different technologies are proposed to achieve accurate description of the component specification and/or user query in their specification and retrieval, the existing semantic-based approaches still fail to achieve the following goals which are desired for present component reuse: precise, automated, semantic-based and domain capable.

This thesis proposes an approach, namely MVICS-based approach, aimed at achieving holistic, semantic-based and adaptation-aware component specification and retrieval. As the foundation, a Multiple-Viewed and Interrelated Component Specification ontology model (MVICS) is first developed for component specification and repository building. The MVICS model provides an ontology-based architecture to specify components from a range of perspectives; it integrates the knowledge of Component-Based Software Engineering (CBSE), and supports ontology evolution to reflect the continuous developments in CBD and components. A formal definition of the MVICS model is presented, which ensures the rigorousness of the model and supports the high level of automation of the retrieval. Furthermore, the MVICS model has a smooth mechanism to integrate with domain related software system ontology. Such integration enhances the function and application scope of the MVICS model by bringing more domain semantics into component specification and retrieval. Another improved feature of the proposed approach is that the effect of possible component adaptation is extended to the related components. Finally a comprehensive profile of the result components shows the search results to the user from a summary to satisfied and unsatisfied discrepancy details. The above features of the approach are well integrated, which enables a holistic view in semantic-based component specification and retrieval.

A prototype tool was developed to exert the power of the MVICS model in expressing semantics and process automation in component specification and retrieval. The tool implements the complete process of component search. Three case studies have been undertaken to illustrate and evaluate the usability and correctness of the approach, in terms of supporting accurate component specification and retrieval, seamless linkage with a domain ontology, adaptive component suggestion and comprehensive result component profile.

A conclusion is drawn based on an analysis of the feedback from the case studies, which shows that the proposed approach can be deployed in real life industrial development. The benefits of MVICS include not only the improvement of the component search precision and recall, reducing the development time and the repository maintenance effort, but also the decrease of human intervention on CBD.

Acknowledgments

I have so many people to thank for the help of my PhD study. Firstly, it is hard to overstate my gratitude to my first supervisor Dr. Xiaodong Liu, second supervisor Professor Jessie Kennedy and penal chair Professor Ben Paechter. Without their inspirational guidance, constant encouragements and unselfish help, I could never finish my doctoral work. They are not only my teachers, but also lifetime friends and advisors.

Secondly, I wish to thank my colleagues at the Centre of Information and Software Systems in Edinburgh Napier University for their support and feedback, and for providing me with such a stimulating and friendly working atmosphere.

Thirdly, I would like to thank my case study partners Mr. Jia Ding (SE) in Beijing Mysoft Technology Co., Ltd, Mr. Xikun Yang (SSE) and Mr. Jie Li (SE) in Mobile Game Development Department of Gameloft Co., Ltd and other anonymous participants whose test results and valuable comments validate and further improve my research outcome.

Fourthly, I would acknowledge Agilent Technologies Foundations for the research grant that partly sponsored my PhD project.

I would give the last but most appreciated thanks to my wife and my parents for their unceasing moral and daily life support throughout the three and half years of my PhD study.

Declaration

I declare that the work described within this thesis was originally taken by me between the dates of registration fro the degree of Doctor of Philosophy at Edinburgh Napier University, October 2007 to August 2011.

The thesis is written by me and has been produced using Microsoft Word.

Publications from the PhD Work

[1] Li, C., Liu, X., 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)*. San Francisco, USA: Knowledge Systems Institute.

[2] Li, C., Pooley, R., Liu, X. (2010). Ontology-Based Quality Attributes Prediction in Component-Based Development. In: *International Journal of Computer Science & Information Technology (IJCSIT)*, *2*(5).

[3] Li, C., Liu, X., Kennedy, J. (2009). A Multiple Viewed Interrelated Ontology Model for Holistic Component Specification and Retrieval. In: *Proceedings of the 1st International Conference on Advanced Software Engineering & Its Applications (ASEA2009), Springer-Verlag's LNCS*. Volume 59, (pp. 50-69). Springer-Verlag.

[4] Li, C., Liu, X., Kennedy, J. (2008). Semantics-Based Component Repository: State of Arts and a Calculation Rating Factor-based Framework. In: *Proceedings of Computer Software and Applications Conference*. (pp. 751-756) IEEE Computer Society Press.

Table of Contents

1. INTRODUCTION .... - 1 -

1.1 Problem Statement .... - 1 -

1.2 Objectives of the Proposed PhD Research .... - 3 -

1.3 Contributions to Knowledge .... - 5 -

1.4 The Structure of the Thesis .... - 3 -

2. LITERATURE REVIEW .... - 10 -

2.1 Software Reuse .... - 10 -

2.1.1 The Process of Software Reuse .... - 11 -

2.1.2 Software Reuse Approaches .... - 13 -

2.1.2.1 Design Patterns .... - 13 -

2.1.2.2 Generator-Based Reuse .... - 14 -

2.1.2.3 Application Framework .... - 15 -

2.1.2.4 COTS Product Reuse .... - 15 -

2.1.2.5 Software Product Line .... - 16 -

2.1.2.6 Component-Based Development .... - 17 -

2.2 Component-Based Software Engineering (CBSE) .... - 17 -

2.2.1 Component Certification .... - 20 -

2.2.2 Component Specification .... - 22 -

2.2.2.1 Characterisation of Components .... - 22 -

2.2.2.2 Component Specification Approaches .... - 25 -

2.2.3 Component Retrieval .... - 34 -

2.2.4 Component Adaptation .... - 36 -

2.3 Ontology .... - 41 -

2.3.1 Ontology Language .... - 42 -

2.3.1.1 Traditional Ontology Language .... - 43 -

2.3.1.2 Web Standards .... - 48 -

2.3.1.3 Web-based Ontology Language .... - 50 -

2.3.2 Ontology Tools .... - 55 -

2.3.2.1 Ontology Editors .... - 55 -

2.3.2.2 Ontology Repositories .... - 56 -

2.3.2.3 Ontology Language Processors .... - 58 -

2.4 Analysis and Conclusion .... - 60 -

3. RELATED WORK.. - 64 -

3.1 Traditional Component Specification and Retrieval Projects... - 65 -

3.2 Ontology-based Component Specification and Retrieval Projects.. - 69 -

3.3 Limitations of Related Work and Conclusion.. - 75 -

4. THE APPROACH .... - 77 -

4.1 Overview.. - 77 -

4.2 The MVICS Framework .... - 79 -

4.2.1 Query Requirement Collection .... - 80 -
4.2.2 Ontology Model Construction .... - 81 -
4.2.3 Component Repository .... - 82 -
4.2.4 Component Retrieval .... - 84 -

5. MULTIPLE-VIEWED INTERRELATED COMPONENT SPECIFICATION ONTOLOGY MODEL (MVICS)... - 85 -

5.1 Function Model... - 93 -

5.2 Context Model... - 86 -

5.3 Intrinsic Model.. - 90 -

5.4 Meta-relationship Model.. - 95 -

5.4.1 Matching Propagation Relationship... - 95 -
5.4.2 Conditional Matching Propagation Relationship .... - 97 -
5.4.3 Matching Negation Relationship .... - 99 -
5.4.4 Supersedure Relationship .... - 99 -

5.5 Linkage between Domain Related Software System Ontology and MVICS .... - 101 -

5.5.1 Association Class and Association Link .... - 102 -
5.5.2 Aggregation and Aggregation Link .... - 103 -
5.5.3 Linkage Definition... - 105 -

6. HOLISTIC AND PRECISE COMPONENT RETRIEVAL METHOD.... - 107 -

6.1 MVICS-based Component Retrieval .... - 107 -

6.1.1 MVICS Component Search Path .... - 108 -
6.1.2 AssL and AggL Search Path.. - 109 -
6.1.3 Adaptive Search Path... - 111 -

6.2 Result Component Precision Calculation Method... - 113 -

6.2.1 Weight of Class.. - 114 -
6.2.2 Weight of Search Path .... - 116 -

6.3 Precision Calculation Method .... - 117 -

6.4 Result Component Profile .... **- 118 -**

7. REALISATION OF THE MVICS-BASED COMPONENT SPECIFICATION AND RETRIEVAL .... - 120 -

7.1 Realisation of MVICS-Based Component Repository .... **- 121 -**

7.1.1 OWL Files .... - 122 -

7.1.2 Component Repository .... - 123 -

7.2 Financial Domain Related Ontology .... **- 123 -**

7.3 User Interface .... **- 125 -**

7.4 User Query Refinement .... **- 126 -**

7.5 MVICS Component Search .... **- 127 -**

7.5.1 Refined User Keywords Parser .... - 128 -

7.5.2 MVICS Component Search Path .... - 129 -

7.5.3 Adaptive Component Search Scratch Storage and Adaptive Suggestion Processing .... - 129 -

7.5.4 Result Component Oriented Data Conversion .... - 130 -

7.5.5 Precision Calculation .... - 131 -

7.5.6 Dynamic Weight Assignment .... - 131 -

7.5.7 QAs-based Result Component Ordering .... - 132 -

7.5.8 Result Component Profile .... - 132 -

7.5.9 Search Time Recorder .... - 133 -

7.6 Result Display .... **- 133 -**

7.7 The MVICS Project Website .... **- 134 -**

8. CASE STUDIES .... - 137 -

8.1 Introduction .... **- 137 -**

8.2 Essential Validation and Evaluation of the MVICS Prototype Tool .... **- 138 -**

8.2.1 Background .... - 138 -

8.2.2 The Process .... - 139 -

8.2.3 Analysis .... - 142 -

8.3 Large-scale Two-stage Metadata Management Software System Case Study .... **- 147 -**

8.3.1 Background .... - 148 -

8.3.2 Mysoft's Existing Component Specification and Retrieval .... - 149 -

8.3.2.1 Introduction .... - 149 -

8.3.2.2 Search Scenario .... - 152 -

8.3.2.3 Process .... - 155 -

8.3.3 MVICS Component Specification and Retrieval .... - 157 -

8.3.3.1 Introduction .... - 157 -

8.3.3.2 Process .... - 158 -

8.3.3.3 Analysis .... - 159 -

8.4 Racing Game Case Study - 161 -
8.4.1 Background - 162 -
8.4.2 MVICS-Based Gameloft Component Specification and Retrieval - 163 -
8.4.2.1 RAG Domain Ontology - 163 -
8.4.2.2 Linkage Construction - 166 -
8.4.2.3 MVICS-Based RAG Component Specification and Retrieval - 168 -
8.4.2.4 Analysis - 170 -
8.4.3 Extended Game Domain Ontology Model - 172 -
8.5 Summary - 161 -

9. CONCLUSIONS AND FUTURE WORK - 177 -

9.1 Conclusions and Novel Contributions - 177 -
9.1.1 Conclusions - 177 -
9.1.2 Contributions - 180 -

9.2 Limitations and Future Work - 183 -

REFERENCE - 186 -

APPENDIX A: THE OWL DOC OF THE MVICS MODEL - 199 -

A.1 Function Model - 219 -

A.2 Context Model - 199 -

A.3 Intrinsic Model - 208 -

APPENDIX B: THE OWL DOC OF THE FINANCIAL DOMAIN ONTOLOGY - 229 -

APPENDIX C: THE MVICS PROJECT HOMEPAGE - 255 -

C.1 The Homepage - 255 -

C.2 The Online Prototype Tool - 256 -

C.3 The MVICS-based Component Specification Form - 257 -

APPENDIX D: THE OWL DOC OF THE RAG ONTOLOGY - 258 -

APPENDIX E: THE OWL DOC OF THE GAMELOFT ONTOLOGY - 268 -

List of Figures

Figure 2.1: Process of CBSE…………………………………………………………………..…….-19-

Figure 2.2: Classification of ontology languages……………………………………………..……-43-

Figure 4.1: The framework of the MVICS approach……………………………………..…..-79-

Figure 4.2: The architecture of the query requirement collection…………………………..-80-

Figure 4.3: The architecture of the query requirement collection…………………………..-82-

Figure 4.4: The architecture of the repository construction……..……………………………-83-

Figure 5.1: Multiple-Viewed and Interrelated Component Specification Ontology Model.........-86-

Figure 5.2: Top three level classes of a) function model, b) context model, c) intrinsic model...-88-

Figure 5.3: The impact on search path: a) Matching Propagation Relationship; b) Conditional Matching Propagation Relationship; and c) Supersedure Relationship…..……………………..-97-

Figure 5.4: An example of Association Link………………………..………………………-103-

Figure 5.5: An example of Aggregation Link………………………………………………-104-

Figure 6.1: Example of the original search path……………………..………………………..-108-

Figure 6.2: Example of domain AssL search path…………………………………………….-109-

Figure 6.3: Example of domain AggL search path...…………………………………………-110-

Figure 6.4: Example of adaptive component search path……………..…………………….…-113-

Figure 6.5: The Result Component Profile………………………………………………......-119-

Figure 7.1: The main interface of the prototype tool…………………………………………-125-

Figure 7.2: The structure of the user query refinement…………..…………………………..-127-

Figure 7.3: The workflow of the MVICS component search…………………………..…..-128-

Figure 8.1: The level of satisfaction of the MVICS component search tool, existing ontological domain specific search tools and traditional search tools………………………………….....-145-

Figure 8.2: Specification levels and specification aspects……………………………………-150-

Figure 8.3: Facets of search parameters…………………………………………………..…-152-

Figure 8.4: Two-stage functional requirements………………………………………………-154-

Figure 8.5: Three sizes of result components compares with required system functions scope……………………………………………………………………………..…………..- 133 -

Figure 8.6: Feedback analysis of the MVIVS-based approach and the Mysoft approach…....-160-

Figure 8.7: The top level classes of RAG domain ontology……………………………………..-164-

Figure 8.8: RAG domain ontology…………………………………………………………...-165-

Figure 8.9: User satisfaction of the MVICS-based approach and the Gameloft approach…...-171-

Figure 8.10:The statistics of working days for each phase………………………………………-171-

Figure 8.11: The top two level classes of the Gameloft ontology……………………………….-173-

List of Tables

Table 8.1: Experience level of testers……………………………………………………………- 143 -

Table 8.2: professional background of testers………….………………………………………….- 143 -

Table 8.3: Example of component specification document…………………………………….- 151 -

1. Introduction

1.1 Problem Statement

Component-Based Development (CBD) has been widely used to develop software systems by assembling and composing already built software components. Numerous advantages of CBD have been identified such as shortened development life cycle [36], reduced time-to-market [119][146], and reduced development costs [36][146]. However, at present CBD still fails to reach its full potential and has not been as widely accepted as it should be due to a few unsolved major hurdles, one of which is the lack of effective and automated methods for holistically and semantically specifying and retrieving existing components that precisely match user reuse requirements [126].

The above problem is basically caused by the lack of semantic-based component specification/repository and retrieval technologies. In order to solve the problem, four research questions need to be further investigated.

- **Is it possible to construct an ontology to cover all component specification?**

First of all, in the semantic-based approach, the ontology plays a key role in supporting the component specification and retrieval. Up to now, there is still no suitably defined ontology that is able to cover the complete characteristics of component specification[1].

- **How to design the architecture of an ontology so that it can organise all characteristics via various relationships?**

Secondly, although a few approaches have started to use an ontological domain model in the component retrieval process, to date it is clear that the ontology in these approaches has a too simple and monolithic structure and few relationships to deal with the specification and retrieval of modern components [171][190][191].

- **How to link component specification ontology with domain ontology?**

Thirdly, even if a hypothetical component specification ontology exists, the range of application is limited by the lack of a method to link the specification ontology to different domain specification ontologies.

- **How to design a technique based on component specification ontology to calculate the search precision?**

[1] According to the development of component based software engineering, the characteristics mentioned here update dynamically

Moreover and as part of the consequence, the existing approaches also failed to rank the components found (result components) with accurate relevance rating and clear unsatisfied discrepancy to reuse requirements.

To summarise the *research hypothesis*, is it possible to develop an effective and automated technique for semantically specifying and retrieving existing components that precisely match user reuse requirements by solving the four research questions.

1.2 Objectives of the Proposed PhD Research

To overcome the limitations in semantic-based component specification and retrieval mentioned in section 1.1, a novel ontology-based approach is proposed to achieve holistic and semantic-based component specification and automatic and precise component retrieval.

The objectives of the research include:

- **To develop a holistic ontology-based approach**

  "Holistic" means that the ontology-based approach has been systematically structured as a spectrum of different aspects in component specification and retrieval, such as domain specific component information, adaptive assets information and the way to present result components comprehensively.

"Ontology-based" means that the component specification and retrieval approach is based on a component specification ontology model suitably defined with ontology languages.

With the help of this model, the characteristics of the component may be specified from multiple perspectives with semantic expression in the repository. The scope of the specification ontology may thus be extended by links between many specific domains in a smooth manner.

- **To assess the search precision of the result components**

  Whereas the existing component specification and retrieval methods only generally differentiate the similar components, a method of search precision calculation algorithm that can assess the search precision numerically may reflect more accurately how the search results match with the user requirements.

- **To build a prototype tool to illustrate and scale up the approach.**

  A prototype tool with an example component repository may help verify and automate the approach, particularly if it can implement the complete process of component search, starting from filling the initial query and ending up with receiving the search results (i.e. to list in the result component profiles).

- **To do case studies to evaluate the approach.**

  Properly designed case studies may help evaluate the practibility and efficiency of the approach in both theoretical and practical level, if the participants do use our prototype tool or put the approach into their own practices.

1.3 Contributions to Knowledge

The proposed ontology-based approach has benefits over existing work in component specification and retrieval, in terms of i) the Multiple-Viewed Interrelated Component Specification(MVICS) ontology model; ii) the method of linking domain ontology with the MVICS; iii) the improved component retrieval method; v) the result component profile. The project originally contributes to the current state of the art by producing the following key technologies, which are integrally linked in a holistic and semantic-based specification and retrieval framework:

- **In correction of Multiple-Viewed and Interrelated Component Specification ontology model (MVICS)**

  The MVICS model provides an ontology-based architecture to specify components from a range of perspectives, it accommodates domain knowledge of CBSE and application domains, and supports ontology evolution to reflect the continuous developments in CBD and components.

Another unique feature is that the effect of possible component adaptation assets/methods are also described under the MVICS model, which enables a more systematic and holistic view in component specification and selection. OWL-DL is adopted to define the classes, individuals, relationships and constraints of the MVICS. Via these formal definitions in OWL-DL, the automatic semantic-based component search and validation are achieved with the support of an ontology reasoner.

- **To take into account domain ontology linkage method**

  To extend the scope of the component specification ontology into specific domains, a domain ontology linkage method is developed. The method links the domain specific ontology with the MVICS by two mechanisms: *Association Link* (*AssL*) and *Aggregation Link* (*AggL*). These two links can be used to connect all the classes of the required domain ontology with their related classes in the MVICS, which enhances the function and application scope of the MVICS-based component specification and retrieval.

- **To improve semantic-based component retrieval method**

  The existing semantic-based component retrieval method is improved based on the MVICS model in the following aspects: 1) the relevance rating for the result component is calculated on the basis of the search paths. The rating is more accurate than the existing approaches as it is produced with the help of the tree structure of the MVICS. The result component are obtained by matching with the lower level classes, with higher precision; 2) the relevance rating method is updated dynamically by changing the factors of the search precision calculation algorithm, according to user feedback; 3) the domain specific components are gained through the domain linkage paths (*AggL* and *AssL*); 4) the adaptive components assets/methods can be retrieved through the adaptive search paths, which provide more choices for the user.

- **Find comprehensive result component profile**

A result component profile is designed to present the comprehensive search result of each result component. In addition to the usual contents of the result, such as the component name and search precision, the profile also presents the new features offered by the MVICS-based approach, such as the percentage of matched keywords, search paths, optional adaptive assets/methods and QAs suggestions. All the above search results are classified and shown in a specially designed display panel, including summary, result representation by facet of the MVICS and result representation by search path.

1.4 The Structure of the Thesis

This thesis is organised as follows:

Chapter 1 gives the background, motivation, scope and original contribution of the thesis.

Chapter 2 provides an overview of the current state of software reuse, component-based development and ontology.

Chapter 3 presents a critical evaluation of existing representative related work in semantic-based component specification and retrieval, which triggers the motivation of the research.

Chapter 4 proposes a framework for the holistic, semantic-based, adaptation-aware component specification and retrieval approach.

Chapter 5 describes the Multiple-Viewed Interrelated Component Specification ontology model, the domain specific ontology linkage method and their formal definition.

Chapter 6 focuses on presenting the key parts of the holistic and precise component retrieval method.

Chapter 7 explores the realisation of the proposed approach in the form of a prototype tool. It describes the tool's system architecture, as well as the activities of each function module.

Chapter 8 provides laboratorial and industrial experiments as the evidence for the evaluation of the proposed approach and tool with four case studies according to a set of criteria.

In chapter 9, conclusions are drawn based on the case studies and evaluation. Potential future work is identified for further development of the proposed approach and tool.

2. Literature Review

In the literature review, a broad survey was completed which investigates the current state of the art of software reuse, component-based development and ontology. These technologies are the foundations of the development of the proposed approach. Through the range of the investigation from broad to detail, the merits of the existing techniques are recognized as well as the demerits, which are then addressed as research topics in the project.

2.1 Software Reuse

Software reuse is the use of existing software, or software knowledge, to build new software. It is the process whereby an organization defines a set of systematic operating procedures to specify, produce, classify, retrieve, and adapt reusable assets for the purpose of using them in its development activities. Reusable assets can be from any part of the software development life cycle including software components, objects, software requirement analysis and design models, domain architectures, database schemas, code documentation, test scenarios, and plans[54][111][156]. According to the survey from the ESDS Software Reuse Working Group [86] in 2006, up to 92% of a new application can be developed by reusing existing software, about 68% that is domain specific and about 24% that is domain independent. And in recent years, the proportion continued to expand. Software reuse is widely used, because it has a lot of the expected benefits including: lower development costs, higher productivity and better use of resources, reduced cycle time and quicker development, lower training costs, easier maintenance, higher quality, lower risk, better interoperability [167].

2.1.1 The Process of Software Reuse

Software reuse involves three steps: abstraction of the assets, storage of the assets and recontextualisation of the assets [54]. Abstraction focuses on designing a reusable asset. Software reuse advocates drawing a strong distinction between designing reusable code as opposed to salvaging code from current systems [43]. The designed reusable assets can be roughly classified into three groups of different sizes as follows [167].

- Object and function reuse. Software components that implement single functions, such as a database connection or a class, may be reused. This type of reuse, based on function libraries or class libraries, has been commonly used for the past 40 years. Many libraries of functions and classes for different types of application and platform are available. These can be easily used by invoking them with other application code.

- Component reuse. All components of an application may be reused. For example, GUI components developed as part of a word-processing system may be reused in a spreadsheet system.

- Application system reuse. The whole of an application system may be reused by customizing the application for different users or by developing application families that have the same architecture but are tailored for specific users.

The second step of software reuse is to make these assets available for others to use. Software developers often encounter this problem in the process of storing and retrieving components. Even in simple cases, storing and retrieving assets proves to be difficult due to the multiple ways that software can be indexed [52]. Particularly for larger software development, many reusable assets and their related diverse software, design diagrams, software processes, or associated development tools are needed; a simple storage and retrieval mechanism is not possible.

The final challenge for software reuse is recontextualisation. This involves making what is reused understandable to those who will incorporate it into their systems [54]. At this point, software reuse advocates it is necessary to motivate others to use reusable assets and ensure that the assets are technically compatible. Reuse of software components raises not only cognitive issues of understanding but also raises issues of incentives that managers often need to address along with the technical issues of compatibility[159][175].

2.1.2 Software Reuse Approaches

To meet the challenges in the process, many approaches have been developed to support software reuse over the past 20 years. The question which is the most appropriate approach in the development depends on the requirements of the system. The key factors that should be considered include: the development schedule, the expected software lifetime, the background, skills and expertise of the development team, the criticality of the software and its non-functional requirements, the application domain and the platform [167]. The representative approaches of software reuse are introduced as follows.

2.1.2.1 Design Patterns

When executable components are reused, they are inevitably constrained by detailed design decisions that have been made by the implementers of these components [38]. These decisions range from the particular algorithms that have been used to implement the components to the objects and types in the component interfaces [63]. If these design decisions conflict with the particular requirements then reusing the component is either impossible or introduces significant inefficiencies into the system.

One way around this is to reuse more abstract designs that do not include implementation details. These are then implemented specifically to fit the application requirements. This approach to reuse has been embodied in the notion of design patterns [131][179][182]. A “pattern” is a description of the problem and the essence of its solution so that it may be reused in a different setting. A pattern is not a detailed specification. Rather, it can be thought of as a description of accumulated wisdom and experience [13][28][64]. It is a well-tried solution to a common problem.

2.1.2.2 Generator-Based Reuse

The concept of reuse through patterns relies on describing the concept in an abstract way and leaving it up to the software developer to create an implementation. It was suggested that an alternative approach to this was generator-based reuse [154]. In the approach, reusable knowledge is captured in a program generator system that can be programmed by the domain experts using either a domain oriented language or an interactive CASE tool.

Generator-based reuse takes advantage of the fact that applications in the same domain, such as the business systems, have common architectures and carry out comparable functions. Generator-based reuse is cost effective for applications such as business data processing. Furthermore, it is much easier for end users to develop programs. There are inefficiencies in generated programs. In other words, it may not be possible to use this approach in systems with high performance or throughput requirements. Moreover, generator-based reuse is limited to specific domains.

2.1.2.3 Application Framework

In the early stages of Object Oriented Programming (OOP), objects were regarded as the most appropriate abstraction for reuse. However, experience has shown that objects are often too fine-grained and too specialised to a particular requirement. The larger-grain abstractions called Frameworks provide a better solution for object-oriented reuse [167].

An Application Framework is a system built by a collection of various classes and interfaces between them [39][137][187]. Applications developed using a framework has the great potential for further reuse through software product line technologies. Consequently, the maintenance of these systems such as modifying family members to create new family members is simplified [167].

2.1.2.4 COTS Product Reuse

A commercial-off-the-shelf (COTS) product [6][7][17][100][167] is a software that can be used directly by its buyer without any modifications. As COTS product is developed for general purpose, such as word-processing, database management, etc., it usually has many features that can be reused in many different applications. Although there can be problems with this approach to system construction [176], COTS is widely used across government and enterprises because they offer significant savings, in terms of costs and development time.

Some types of COTS products have been very popular for many years, such as database management and GUI. Very few developers want to implement their own database system. However, until the mid-1990s, integrating these large systems and making them work together was a big challenge because most large systems were designed as standalone systems [150][173].

At present, well-defined Application Programming Interfaces (APIs) that allow program access to system functions is always available in COTS product. Consequently, constructing a large system by integrating a range of COTS product is a popular approach. In this way, the costs of development and delivery times are reduced. Furthermore, risk may be reduced as the mature COTS products are already available.

2.1.2.5 Software Product Line

A product line is a related set of applications that has a common domain-specific architecture [11][172], which allows the implementation of planned reuse of components within an organization.

Such architecture supports the structured assembly of product-line components, whereby various components to be assembled following the implicit rules of planned architecture. This approach has proven to be very successful for industry that produces many variants of products with similar functionality. Based on the future plans of product development, the components are developed according to guidelines of product-line architecture and thus enables them to be reused (i.e. assembled) under the same guidelines [172].

2.1.2.6 Component-Based Development

Component-Based Development (CBD) is a representative approach of software reuse. It focuses on reduction of development time through technical facilities that enable the easy assembly and upgrading of systems with the pre-defined components. Because we propose to solve the existing problems in CBD by using the semantic-based method, an intensive literature review was completed to find the current state of art of Component-based Software Engineering (CBSE) and ontology in the following sections.

2.2 Component-Based Software Engineering (CBSE)

Nowadays, software is widely used in daily life of almost every sector. To satisfy the demand of various users in terms of efficiency, flexibility, reliability and security, to name a few, software has become more powerful in function, and more complicated in structure. As a result, software programs have become larger and more complex, and thus more time-consuming for the programmers to deliver. Having been perceived as a means to be able to help deliver a more user-friendly system within a tight timeframe, Component-based Software Engineering began to develop in the late 1980s.

By using a component-based development approach, i.e. developing software systems by assembling and composing components which have already been built, it is apparent that component-based development has contributed tremendously to the software industry in the following aspects [112][158][167]:

- Increase developer's productivity
- Reduce skills requirement
- Shorten development life cycle
- Reduce time to market
- Increase quality of the developed system
- Decrease development costs

In spite of these foregoing benefits, it carries the following restraints in software development [167]:

- Time and effort required for development of components,
- Unclear and ambiguous requirements,
- Conflict between usability and reusability,
- Component maintenance costs.

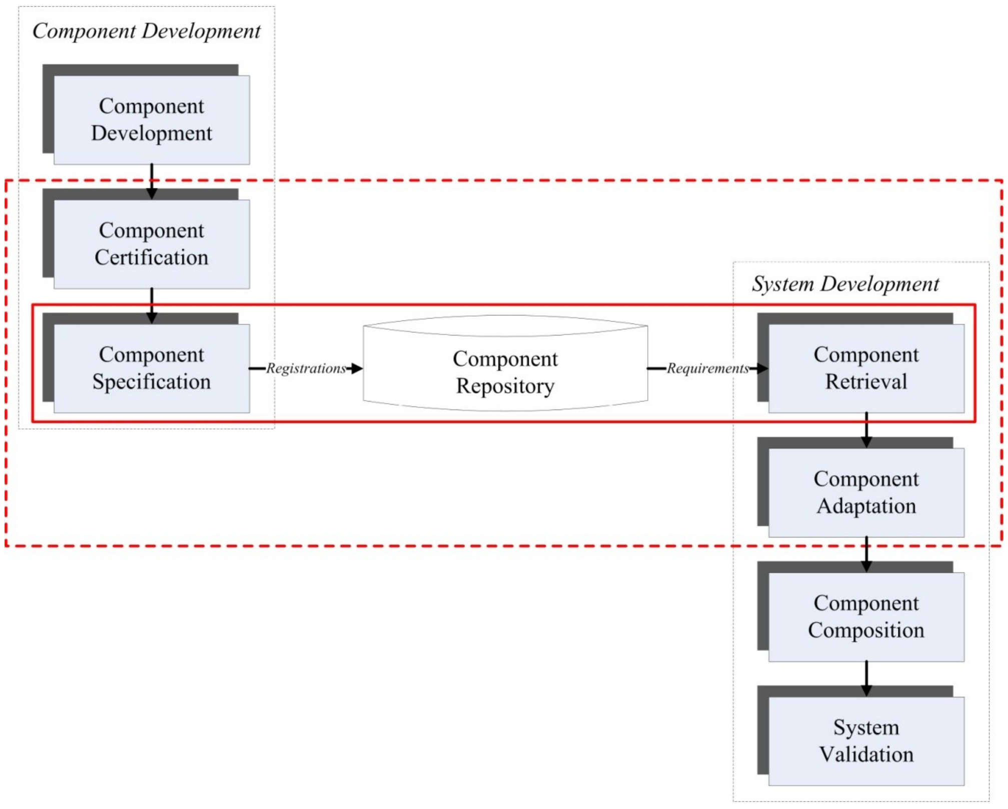


Fig. 2.1. Process of CBSE

To understand these restraints in further detail, we need to have a closer look into the basic process of CBSE. A very important characteristic that distinguishes component-based development from other types of development is the separated development processes for system development and component development, which is illustrated in Figure 2.1. In system development, specialized products are built through reuse. While in component development, general components are built for reuse [2].

The focus to develop reusable components is stressed through the separate component-development process, which ends with delivery to a common component repository where reusable components are stored. Mili [137] provides an elegant definition of reusability of a software components as the aggregation of the quality attributes usability, and usefulness. These attributes must be first class goals for the component-development in order to maximize reusability of the components.

In a component-based system development process, the development team focuses on efficient development of a system dedicated for a certain purpose. The goal of the system development process can thus be compared to the goal of a traditional software systems development process, but the major difference is the method to integrate the selected components (Component Composition).

The steps inside the red dotted lines are the core steps of CBD, including component certification, component specification, component retrieval and component composition, among which component specification and retrieval are the focus of this research.

2.2.1 Component Certification

With the objective of evaluating the current necessity for software components, Carnegie Mellon University's Software Engineering Institute (CMU/SEI) has studied industry trends in the use of software components [9]. The study examined the existing components from both technical and business perspectives. One of inhibitors for adopting software component technology is the "Lack of certified components".

To solve the above limitation, Trass and Hillegersberg [174] discuss the conditions for the growth of component markets. Some of which are intrinsic characteristics of certification processes, such as the documentation and component quality and well-defined services. More evidence [25][66][67][183] also points to certification as the key precondition for CBSE to be successfully adopted in the large. Moreover, certified components used during development will have predetermined and well-established criteria in order to reduce the risks of system failure. However, there is no consensus of what component certification really means within the CBSE. In [29], Councill established a satisfactory definition about what software component certification is:

"Third-party certification is a method to ensure that software components conform to well-defined standards; based on this certification, trusted assemblies of components can be constructed."

To show that a component conforms to well-defined standards, the certification process must provide a certificate that evidences that it fulfils a given set of requirements. Thus, trusted assembly, application development based on third-party composition, may be performed based on previously established quality levels.

The existing component certification approaches can be divided into two stages: from 1993 to 2001, the approaches appearing in this stage were mainly focused on mathematical and test-based models; and after 2001, the focus of the techniques and models shifted gradually to predicted quality requirements. The representative approaches in the early stage are listed in the order of appearance, including Poore et al. [151], Wohlin & Runeson [188], Rohde et al. [155], TCI [134], Voas [180], Voas & Payne [181] and Morris et al. [139]. The approaches focusing on reuse level degree, reliability degree, among other properties in the second stage, include Stafford & Wallnau [169], Councill [29], Hissam et al. [68], McGregor [133]. However, component certification is still an open research area. The effort to develop a component certification standard is still ongoing.

2.2.2 Component Specification

In CBSE research, much effort has gone into developing specification techniques for software components. The component specification initially concentrated on syntactic description of interfaces, and eventually moved to adding some semantic information, namely as pre- and post-conditions on the operations defined in the interfaces. To date, current research has extended the semantic information to include a little more operational descriptions.

2.2.2.1 Characterisation of Components

As more and more systems are built from existing components, it has become increasingly important to have a proper characterisation of components [16]. Without such a characterisation, much effort is wasted on the comprehension and adaptation of components each time they are used.

In a way, a characterisation of components is similar to but extends the characterisation of object technology. It is used to provide a basis for the development, management and use of components. Therefore, it can be referred to as object-oriented characterisation of components [62].

The structural elements are essential aspects of a component in the sense that they constitute the component. Following the general convention, these structural elements are called “attributes”. Among these elements are those that are relevant to the interface of the component i.e., the elements that form part of the component’s external view. These observable elements are particularly important from the viewpoint of component management use and form the basis of other aspects of component characterisation.

Another aspect of a component is the operations with which other system components interact with it. These operations capture the dynamic behaviour of the component and represent the service/functionality that the component provides.

As in the case of objects, attributes and operations are the most essential aspects of a component. In addition, the structural composition of the component from attributes is usually subject to further constraints i.e., compositions of attribute instances may not be allowed. These constraints are called “structural constraints”. Some examples are the constraints about the abstract state of the component.

In terms of the dynamic behaviour of a component, not all possible sequences of operation invocation on the component are allowed. That is, there are operational constraints that specify the permissible operation patterns. This aspect is similar to that of an object captured by the state transition diagram.

Besides proactive control, which is usually in the form of explicit operation invocation, another form of control used to capture system behaviour is reactive control, which is usually in the form of an event-driven implicit operation invocation. It is often the case that certain aspects of a system are better captured through proactive control, while other aspects of the system are better captured in the form of reactive control. To facilitate reactive control, a component may generate events from time to time, which other components in the system may choose to respond to. The Java component model i.e., JavaBeans, provides such support.

A component may interact with a number of other components from specific perspectives (or playing specific roles). The interactions between the component concerned and these other components may differ depending on the components and their related perspectives. The port specification goes some way towards this direction. In general, this suggests the need for defining perspective/role-oriented interaction protocols for a given component i.e., multi interfaces. A component may be used, and play different roles, in different circumstances. The circumstances provide the contexts of use for the component.

Another aspect of a component is its non-functional properties such as reliability, performance and deployability [8][26]. In the context of building systems from existing components, the characterisation of the components' non-functional properties and their impact on enclosing systems are particularly important. However, not much work has been done in this area.

2.2.2.2 Component Specification Approaches

The mess of work are proposed in the area of component specification, the representative approaches are categorized in terms of early approaches, design-by-contract methods, formal methods, and framework-based approaches.

- *Early Approaches*

  Object Management Groups' CORBA and Microsoft's COM/COM+ component frameworks relied entirely on interface definition language (IDL) to specify a component. The IDL specification contained a description of the operations and types that the component used in its interfaces, including the exact signature of the operation with its parameters.

With the introduction of the CORBA Component Model (CCM) by the Object Management Group and the .NET framework by Microsoft, both groups moved away from having IDL as the only method of specification. CCM is extended, including provides and requires interfaces, event sources and sinks, and attributes. The .NET framework went away from explicit specification of an IDL choosing instead to rely on the components themselves to supply the necessary information (extracted via .NET's Common Language Runtime).

Another popular method for writing specifications is free-text. This is a carryover from more traditional programming methods, from the days of libraries of procedures and, more recently, of standard template libraries. The Swing library of GUI components available as part of Sun Microsystems JavaTM releases is specified using JavaDoc, a web-based specification method that is the de facto standard of learning about Java components (as well as the Java library itself). Free-text-based specification is still popular because of two reasons. It is human readable and understandable, and with the advent of web hyperlinks, even easy to use. Free-text allows the component implementer to specify just about anything they wish about the component, to any level of precision.

- *Design-by-Contract*

Another category of component specifications is the one made popular by the Cheesman and Daniels component specification process [24] as well as by the

Catalysis Approach of D’Souza et al. [34]. Both approaches employ design-by-contract methods, which are characterized by the assertion that a component can be precisely specified solely by the specification of operations in the interfaces offered by that component. These methods define the contract as a collection of assertions that describe what a component does and does not do. Not only do these assertions precisely describe the behavior of the operations in the component, they are also logical expressions of the entities in the information model of the component. The key assertions of the contracts are invariants, pre-conditions, and post-conditions.

Design by contract methods generally specifies the design of the component using a graphical language, most often UML. UML can be used, because almost all of the specification methods that fall in this category assume that the component developer will use object-oriented methods to design and implement component functionality. The processes described in [152] and [155] focus almost entirely on the development of the component. The extension mechanism of stereotypes in UML is used to build component specification architectures and component interaction diagrams. These models describe using UML, the interfaces provided by the components in the model, and in the case of [152], the interfaces required by the component. Some design by contract methods may develop behavioral diagrams, such as collaboration diagrams, interaction diagrams, or even state machines that have been applied at the component, rather than object level. The heart of this UML specification is a model containing a set of class elements, stereotyped as components, each naming an interface, and each containing a list of operation signatures. These are the provided interfaces of the component.

The resultant component specification is specified in UML, with the contract constraints specified in some formal language; typically, they are expressed in OCL [185]. The results in a component specification using a graphical model with textual attributes attached to specific operations in the model.

In the description of their specification method, Liu and Cunningham [128] state that their method provides two complementary purposes: (1) to specify expected functionality required by a component to fill a specific need; and (2) to specify the actual functionality implemented in an available component. They, however, provide no reason for this position.

The Catalysis Approach offered by D'Souza et al. [34] employs an action language to specify operational constraints instead of using OCL. The claim here is that the action language offers fully operational specification capabilities.

Kim et al. [118] offers a methodology which, like D'Souza's, also employs an action system, instead of static OCL constraints. They state that the difference between their approach and others in this category is that they use this action system to express dynamic configuration properties of the component. In their method, components are again modeled in UML, defining object relationships within the component. Once interfaces are developed, the dynamic operational specification of each operation in the interface is specified in simple statements in an action language based on existence dependencies2. To obtain the rigor desired by the authors, these statements are then translated into Petri nets [111] to take advantage of formal analysis techniques. These Petri net specifications are then used to build an execution model. They state that their method offers theoretical means for describing and analyzing both structural and behavioral aspects of components.

- *Formal Methods*

  Unlike the design by contract approaches discussed previously, component specification via formal methods tends to focus more on verification of the specified model and component reuse. Formal specifications provide precise descriptions of problem requirements, component function, and component structure [147]. Formal inference defines a mechanism for reliably and formally comparing problem requirements and component specifications. However, most specification methods concentrate their efforts on the structural and operational aspects of the interfaces of the component.

Penix and Alexander [148] concentrate on formal specification of the component inputs and outputs using axiomatic expressions specifying the pre-conditions and post-conditions on the inputs to and outputs of, respectively, the component interface operations in terms of the components' modeled domain and range. They also allow for constraints to be specified not only for specific operations, but for the component as a whole. Properties describing the environment in which the component will execute are given only cursory attention, and there is no discussion of the dependencies between the specified component and other system elements.

Morel and Alexander [138] also offer an approach formalizing the pre- and post-conditions of component interfaces. Their motivation is the development of a component adaptation framework, stating that matches found between components in the selection process will never likely meet all the requirements. Thus, closely matching components will need to be adapted in order to be assembled into the target component system. They have extended the ideas in [168] to include system level properties. The component specifications are written in Rosetta, which is a systems level design language for modeling heterogeneous systems. A facet specifies a particular aspect of a system or component. Facets can describe behaviors and more general requirements. A facet operates in a declared domain (system environment), which defines semantics available to the facet.

- *Frameworks*

Frameworks offer a foundation for methods of component specification. They direct the contents and format of specifications, but do not dictate a process or prescribe methods to obtain the information that makes up the contents of the specifications. Frameworks are far more comprehensive in defining the amount of information required for a component specification because they focus on generalizing component operation in a specific environment.

Three prominent commercially available frameworks (.NET, CCM, and EJB) focus on implementation and development of components that are constrained to the environment for which they were developed. Even though they provide a good set of practical solutions to many of the issues facing component development, there is a lack of support for component selection.

For specifically targeted components, the burden of selection should be easier in these environments, by the reason of the environments share a common set of environment and system assumptions, and configuration properties. However, that since these component frameworks have been implemented by different vendors (excepting COM/.NET), the burden of selection was not apparent, as component "experts" were able to "select" components based on experience and advice from others.

Within their own environments, reusability of components is fairly good, due to the set of common assumptions on the environment. Even between different implementations of the same framework (e.g., CORBA), component reuse here is better than the general case.

Let us look at the following two frameworks to see what they offer in the context of specifying components for selection and reuse.

Han [62] offers an approach to software component specification containing four specific sets of information. First, the properties, operation, and events of the component form the signature of the component interface. Second, constraints further restrict and precisely define component interface. Han states that the signature and the constraints together characterize the component capability. The third set of information is component usage in context. Configurations are defined based on the component usage scenarios. A configuration identifies the roles and defines the role-based interfaces of the component in a given use context. Lastly, the component's non-functional properties are useful in assessing the component's usability in given situations and in analyzing properties of the enclosing systems.

A framework whose primary focus is the architectural connectivity between components, proposed by Tracz [175], is named SOFA, for SOFtware Appliances. Self described as a platform for components, SOFA views applications as hierarchies of nested components, wherein each component is either primitive or composed of primitive components. As a result of hierarchy, all functionality is in primitive components. There is no direct mechanism for inheritance of components. The SOFA component model specifies several framework-level features. First, SOFA specifies interfaces, much in the way as has been done in other methods and approaches. It then defines Frames, which are black-box views of the component, containing a view of the provided and required interfaces plus some component configuration parameters. Next is the Architectures, more of a gray-box view, which define the ties between the subcomponents of the modeled component. SOFA defines four types of ties: binding, wherein a requires-interface is bound to a provides-interface; delegating, wherein a provides-interface of component is bound to a subcomponent's provides-interface; subsuming, wherein a subcomponent's requires-interface is bound to a requires interface of component; and exempting, wherein the interface of a subcomponent is exempted from any ties. Non-exempted ties are realized via a connector, which models the interconnection between components, implements interaction semantics, and takes account of the deployment details.

In addition to these structural specifications, SOFA utilizes behavior protocols. These protocols represent the formal capture of communication between components, modeled as events (specifically emit method call, accept call, emit return, accept return). Sequences of events are called a trace. SOFA then defines that the behavior of an entity (an interface, frame, or architecture) is the set of all traces which can be produced by that entity. All the information is captured using SOFA's Component Description Language, or CDL. CDL is like the IDLs we have seen above, but has extensions to support frames, architectures, and the behavior protocols. These extensions make CDL a rather complex specification language.

2.2.3 Component Retrieval

A major limitation of software reuse is the lack of efficient means to search and retrieve reusable artefacts from the repository. The same problem is even more severe in the CBD.

There are several factors that impact the search and retrieval process including scope of the repository, query representation, asset representation, storage structure, navigation scheme, relevance and matching criteria [136]. Most retrieval methods focus on one or two of these factors to suit the specific domain they are working in. Consequently, these methods do not provide an effective overall retrieval solution.

As mentioned above, one of the big hurdles in component-based development is how to retrieve suitable components. Among the bulk of solutions proposed so far, none of them satisfactorily surpassed the known hurdles in searching and retrieving relevant components from a repository.

The existing approaches can be classified into four different types:

a) *Simple keyword and string search* [137]

Simple keyword searching is most widely used by search engines, whereby a user can locate the target object by specifying a set of keywords. The search engine will compare them with the labels of the objects and retrieve those matched. This is the most simplified approach without giving due consideration to any additional information, such as relationships among objects or synonymous labels.

b) *Faceted classification and retrieval* [143][153]

The faceted classification approach attempts to classify the objects in the repository based on predefined taxonomies, e.g. subjects. Although this approach is useful for objects that can clearly fall into such categories, it is less useful for those without explicit classification.

c) *Signature matching* [192]

Signature matching focuses on matching of function types and argument types to the query specified by the user. Signature matching could be one either at the function level or module level. But sometimes it is difficult to map user requirements with function and module signatures. And it does not work for multiple components.

d) *Behavioural matching* [59][193]

Behavioural matching is the most involved approach with consideration of the functional behaviour of objects, in which objects are provided with input vectors, and the outputs generated are compared to the expected outputs. Objects that exhibit certain behaviour are retrieved and presented to the user.

The last two approaches are considered cumbersome and inefficient without giving due consideration to the domain and search context information. It is apparent that all the search methods mentioned above are lacking accurate semantics of the component and the user query.

2.2.4 Component Adaptation

Component adaptation refers to the process of changing the component for use in a particular application. While the component may only partially satisfy the user requirement, it is essential to modify the behaviors of the component [116], which may create further difficulties for adaptation when the original components are not suitable for additional requirements [56][64].

Since the early 1990s, a number of techniques have been developed for adapting components. These component adaptation techniques can be categorised into white-box and black-box adaptation techniques. White-box techniques (e.g. inheritance) require the software engineer to adapt a reused component either by changing its internal specification or by overriding and excluding parts of the internal specification. Black-box techniques (e.g. wrapping) reuse the component as it is, but adapt the interface of the component. Black-box adaptation only requires the software engineer to understand the interface of the component, not the internals.

Many requirements for component adaptation have been identified in the prior research [18][116]:

The adapted component should be used in the same way as the original component would have been used, even when the adapted component may behave differently; No additional effort should be required to integrate the adapted component to the target system, as well as to maintain the client-side view; The original component should be open to any future adaptation while maintaining its original identity.

This suggested that all phases of software engineering need to be considered during the component adaptation [65][129][132], including:

- The design phase, when the component designer specifies the interfaces, interoperability, and relationship between components;

- The implementation phase, when the components is constructed from design specifications, including writing the code, obtaining the source code for all the dependent components, implementations of interfaces and database tables, compilation and linking of source files;

- The deployment phase, when the component is deployed into a component infrastructure.

Four representative component adaptation techniques are presented as follows:

- *Inheritance*

  Inheritance makes the state and behaviour of the reused component available to the reusing component [116]. Depending on the language model, all or part of the internal aspects become available to the reusing component.

  One of the important advantages of inheritance is that the code remains in just one location. However, it is disadvantaged by the fact that the software engineer concerned must have detailed understanding of the internal functionality of a superclass when overriding superclass methods and when defining new behaviour using behaviour defined in the superclass.

- *Wrapping*

Wrapping is a process to encapsulate a component to forward the functionality of the wrapped component with minor changes based on the client's request [44].There is no clear boundary between wrapping and *aggregation*. While wrapping is used to adapt the behaviour of the enclosed component, *aggregation* is used to compose new functionality out of existing components providing relevant functionality. A major disadvantage of wrapping is that it may result in considerable implementation overhead since the complete interface of the wrapped component needs to be handled by the wrapper, including those interface elements that need not be adapted. In addition, wrapping may lead to excessive amounts of adaptation code and serious performance reductions [69].

- *Superimposition*

Superimposition is a technique based on the concept that the entire functionality of a component (rather than that of a single method) should be superimposed by certain behavior [18]. The superimposition of a behavior B over a component C is the additional overriding behavior of B over the whole component C. The principle underlying superimposition is that the component C and the overriding behavior B are separate entities which may be reused independently. This overriding behavior is encapsulated as an adaptation entity.

Superimposition is implemented as a language construct in the layered object model (LayOM) through the notion of layers. LayOM is an extended component object model that, next to instance variables and methods, contains parts such as states, categories and layers. The extended expressiveness of LayOM furnishes the software developer with powerful component adaptation types through superimposition.

Superimposition attempts to preserve the transparency of the composition of components with adaptation types - a component and its clients should be unaware of the presence of other adaptation entities that are active on the component.

Superimposition uses nested component adaptation types to compose multiple adaptation behaviours for a single component. However, lack of component information, limits modification to a simple level, such as conversion of parameters, and refinement of operations. Moreover, with more layers of code imposed on original code, the overhead of the adapted component increases heavily. This degrades system efficiency.

- *SAGA*

  The Scenario-based dynamic component Adaptation and GenerAtion (SAGA) [127][184] project has developed a component adaptation approach with little code overhead through XML-based component specification, interrelated adaptation scenarios and corresponding component adaptation and generation. In this project, a Component Definition Language (CDL) is used to record the design configuration of components to be reused in specific applications. Scenarios are established to capture adaptation requirements.

  SAGA is more suitable for the development of traditional component-based systems where developers have access to the internal design of components and can impose intervention on the adaptation and integration process. However, automation is still a challenge in SAGA due to the complexity in generating blocks of code according to scenarios and the original component code.

2.3 Ontology

Ontology attracts many attentions nowadays in computer science. Research fields, like knowledge management, intelligent information integration, e-commerce, cooperative information system, database integration, all claim that ontology will play a major role in the foreseeable future. The reason of the popularity lies in its promise of a shared and common understanding of some domain, which can be communicated across people and computers. In general, the accepted industrial meaning of "ontology" makes it synonymous with "conceptual model" and is clearly independent of its philosophical antecedents [186]. The definitions are adopted here from Uscholod [178]: “ontology is an explicit representation of a conceptualization. This conceptualization includes a set of concepts, their definition and their inter-relationships. Preferably this conceptualization is shared or agreed.” In spite of varying interests in research and the use of ontologies, many available languages and tools are proposed to construct good ontologies.

2.3.1 Ontology Language

The categorization of the existing ontology languages is adopted from the evaluation of languages in [12], including traditional ontology languages, web standards and web-based ontology languages. Figure 2.2 depicts the candidate languages and their relations.

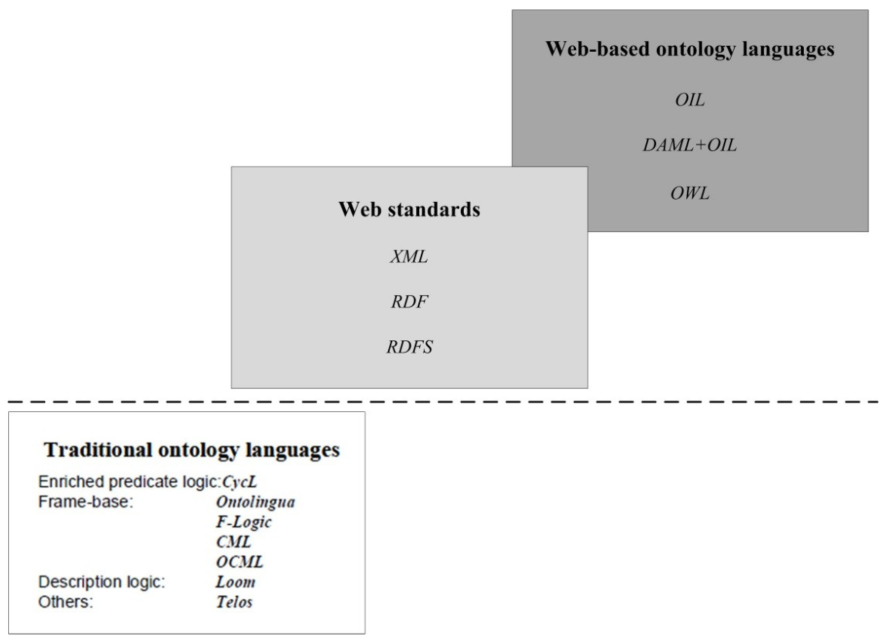


Fig. 2.2. Classification of ontology languages

2.3.1.1 Traditional Ontology Language

Traditional ontology languages are mostly rooted from AI or Knowledge engineering. The languages can be further divided into four groups as shown in Figure 2.2.

The first group is enriched first-order predicate logic. KIF and CycL whose central modeling primitive are predicates can be representative here. However KIF is eliminated, since its general purpose is for interchange. CycL is a formal language that focuses on proving a general ontology for commonsense knowledge. Its syntax derives from first-order predicate calculus, and was first developed in the Cyc project [125]. The Cyc has created and manages a large knowledge base for common-sense knowledge created with this language. To express real-world concepts the language has a vocabulary of terms (about 160 of them), which can be combined into meaningful CycL expressions. Some of the main concepts of CycL are: constants, variables, formulas, predicates and microtheories. Microtheories are sets of formulas, but they can also participate in formulas, i.e. reification. The microtheories provide a context for the truth of formulas.

The second group is frame-based languages, Ontolingua, F-logic, CML and OCML fall into that group. Classes (frames) are the central modeling primitives.

- *Ontolingua*

  The term Ontolingua is overloaded, referring both to the system and to the language. The Ontolingua language is based on KIF (Knowledge Interchange Format) [47] and the Frame Ontology [55]. KIF has a declarative semantic and is based on first-order predicate calculus. It provides definitions for object, function, relation and logical constants. KIF is a language for knowledge exchange, and is tedious to use for the development of ontologies. Thus, the Frame Ontology is built on top of KIF, and provides definitions for object-oriented and frame-language terms, like class, subclass-of, and instance-of. One good thing with the frame-based style is that it is intuitive to humans and thus can be easily understood. But axioms can't be expressed in Frame Ontology.

  Ontolingua lets the developer decide whether to use the full expressiveness of KIF, where axioms can be expressed, or to be more restricted during the specification by using only Frame ontology terms. An ontology developed with Ontolingua is typically defined by: relations, classes (treated as unary relations), functions (defined like a relation), individuals (distinguished objects) and axioms (relate these terms).

- *F-logic (Frame Logic)*

F-logic [117] was developed in the late 80s. It is a logic language integrated with object-oriented or frame-based paradigm. Some fundamental concepts from object-oriented languages have a direct representation in F-logic, for example, class, method, types and inheritance, and other secondary aspects, like polymorphism, can be easily modeled as well. One of the main problems with object oriented approach, lack of logic semantics, is overcome here by the logical foundation of F-logic. There are many similarities between F-logic and Ontolingua, since they both try to integrate frames into logical framework. But the frame-based modeling primitives are explicitly defined in the semantics of F-logic, while Ontolingua treats them as second-order terms defined with KIF axioms. Another difference is that F-logic lacks the powerful reification mechanism. Ontolingua inherits from KIF, which allows the use of formulas as terms of meta-formulas.

- *CML (Conceptual Modeling Language)*

CML [160] was developed to support the CommonKADS framework, and is basically an informal notation for knowledge modeling. CML offers privileges for domain knowledge, inference knowledge and task knowledge. It has many similarities with OCML, but lacks operability capabilities [140] and is more or less dedicated to CommonKADS. CML provides more primitives than OCML in the structural perspective, but the commitment restricts the general ontology specification capabilities. CML will not be evaluated further in this paper because OCML is similar, but more general.

- *OCML (Operational Conceptual Modeling Language)*

OCML was developed and is maintained by the Knowledge Media Institute (KMI) in context of the VITAL project [163]. Its primary purpose is to provide operational knowledge modelling facilities and to achieve this, it includes interpreters for functional and control terms. OCML provides mechanisms for defining relations, functions, classes, instances, rules and procedures. It can be viewed to some extent as "operational ontolingua", which provides theorem proving and functional evaluation mechanisms for Ontolingua constructs. The operational nature of OCML makes it possible to support quick prototyping, which is important for model validation. OCML provides a set of base ontologies that forms a rich modeling platform for building other ontologies: meta, functions, relations, sets, numbers, lists, strings, mapping, frames, inferences, environment and task-method. The functionality of the base ontologies is roughly analogous to Java Foundation Class.

The third group is Description Logic (DL) language, only of which is Loom.

- *LOOM*

  Loom [130] is a knowledge representation and reasoning system based on description logic. A distinguished feature of DL is that classes (concepts) can be defined in terms of descriptions that specify the properties or restrictions, which objects must satisfy in order to belong to the concept. In other words, this means that class membership relations can be determined by inference. One of the primary tasks of Loom is to compute subsumption relationships between descriptions, and organize them into taxonomies. To achieve automatic derivation of taxonomies, Loom offers both a language for the description of objects and relationships, and an assertion language for specifying constraints on the concepts and relations. Loom provides powerful deductive reasoning with underlying production and classification-based inference capabilities.

- *Telos*

  Telos [141] was constructed to handle several modeling aspects, and brought in ideas from knowledge representation, deductive databases and requirement languages for constructing information systems. Basically it's a knowledge representation language with an object-oriented focus. An object is either an individual (entities, concepts, nodes) or an attribute (relationships, attribute links). Telos has high expressive power, especially because individuals and attributes are treated uniformly. Integrity, deductive and constraint rules can be specified through an assertion sub-language, and constructs for temporal statements are offered. The language has hardly been maintained in the last decade, and is therefore not suitable in the interoperability and distribution sense.

2.3.1.2 Web Standards

In this part we review the works that investigate the possibility of using web standards to specify ontologies.

- *XML (Extensible Markup Language)*

  XML [21] is the universal format for structured documents and data on the Web, proposed by the W3C. The main contribution of XML is that it provides a common and communicable syntax for web documents. XML itself is not an ontology language, but XML-Schemas, which define the structure, constraints and the semantics of XML documents, be used to specify ontology. But since XML-schema is created mainly for the verification of XML document and its modeling primitives are more application oriented rather than concept oriented, it will not be viewed as ontology language.

- *RDF (Resource Description Framework)*

RDF [124] is an infrastructure for encoding, exchange and reuse of structured metadata, proposed also by W3C. RDF provides a standard form for representing metadata in XML. The RDF data model consists of three object types: resources (subjects; available or imaginable entity), properties (predicates; describing the resources) and statements (objects; assigning a value for a property in a resource). Principally, information is stored in the form of RDF statements, which are machine understandable. Search engines, intelligent agents, information brokers, browsers and human users can understand and use that semantic information.

- *RDFS (Resource Description Framework Schema)*

RDF Schema (RDFS) [107] enriches the basic RDF model, by providing a vocabulary for RDF, which is assumed to have certain semantics. Predefined properties can be used to model instance of and subclass of relationships as well as domain restrictions and range restrictions of attributes. Indeed, the RDF schema provides modeling primitives that can be used to capture basic semantics in a domain neutral way. That is, RDFS specifies metadata that is applicable to the entities and their properties in all domains. The metadata then serves as a standard model by which RDF tools can operate on specific domain models, since the RDFS meta model elements will have a fixed semantics in all domain models. RDFS provides simple but powerful modeling primitives for structuring domain knowledge into classes and sub classes, properties and sub properties, and can impose restrictions on the domain and range of properties, and defines the semantics of containers.

2.3.1.3 Web-based Ontology Language

- *OIL (Ontology Inference Layer)*

OIL [32] was developed in the On-To-Knowledge project, and is both a representation and exchange language for ontologies. The language is combined with primitives from frame-based languages, and formal semantics and reasoning services from description logics. To enable the use of OIL on the Web it is grounded on the W3C standards, XML and RDF(S). The ontology description is divided into three layers: object level (concrete instances), first meta-level (ontological definitions) and second meta-level (describing features of the ontology). OIL provides definitions for classes and slots (relations), and a limited set of axioms. Slots are treated like first-class citizens, and can be represented in hierarchies. There are several limitations related to axioms, which also limits how expressive the language is [40]. OIL has a accurate semantics that forms a necessary foundation for effective reasoning support.

- *DAML+OIL*

DAML+OIL [71] is a semantic markup language for Web resources, and is a proposed W3C standard for ontological and metadata representation. DAML (DARPA Agent Modeling Language) was transformed to DAML+OIL by including some OIL aspects in the language. DAML+OIL is built on RDF and RDF Schema, but provides richer modeling primitives, commonly found in description logics. Most of the frame-based ideals provided in OIL were removed, and assertions are made in terms of a limited set of axioms. The result is a language that works better as a delivery platform for ontologies than RDF and XTM [12], but has limitations as a language for the development of ontologies, because of the removal of the frame-based constructs.

- *OWL (Web Ontology Language)*

OWL is a family of knowledge representation languages for authoring ontologies. The languages are characterised by formal semantics and RDF/XML-based serializations for the Semantic Web. OWL is endorsed by the World Wide Web Consortium (W3C) [106] and has attracted academic, medical and commercial interest. Intuitively, OWL can represent information about categories of objects and how objects are interrelated. It can also represent information about objects themselves. More precisely, on the one side, OWL let describe classes by specifying relevant properties of objects belonging to them. This can be achieved by means of a partial (necessary) or a complete (necessary and sufficient) definition of a class as a logical combination of other classes, or as an enumeration of specified objects. OWL let also describe properties and provide domains and ranges for them. Domains can be OWL classes, whereas ranges can be either OWL classes or externally-defined datatypes (e.g. string or integer). Moreover, OWL let provide restrictions on how properties behave that are local to a class. Thus, it is possible to define classes where a particular property is restricted so that i) all the values for the property in instances of the class must belong to a certain class or datatype (universal restriction), ii) at least one value must come from a certain class or datatype (existential restriction), and iii) there must be at least or at most a certain number of distinct values (cardinality restriction). On the other side, OWL can also be used to restrict the models to meaningful ones by organizing classes in a subclass hierarchy, as well as properties in a sub property hierarchy. Other features are the possibility to declare properties as transitive, symmetric, functional or inverse of other properties, and couple of classes or properties as disjoint or equivalent. Finally, OWL represents information about individuals, providing axioms that state which objects belong to which classes, what the property values are of specific objects and whether two objects are the same or are distinguished.

OWL is quite a sophisticated language. Several different influences were mandated on its design. The most important are DLs, the frames paradigm and the Semantic Web vision of a stack of languages including XML and RDF. On the one hand, OWL semantics is formalised by means of a DL style model theory. In particular, OWL is based on the SH family of Description Logics (DL) [72], which is equivalent to the ALC DL [5] extended with transitive roles and role hierarchies. Such family of languages represents a suitable balance between expressivity requirements and computational ones. Moreover, practical decision procedures for reasoning on them are available, as well as implemented systems such as FaCT [70] and RACER [98]. On the other hand, OWL formal specification is given by an abstract syntax, which has been heavily influenced by frames and constitutes the surface structure of the language. Class axioms consist of the name of the class being described, a modality indicating whether the definition of the class is partial or complete, and a sequence of property restrictions and names of more general classes, whereas property axioms specify the name of the property and its various features. Such a frame-like syntax makes OWL easier to understand and to use.

Moreover, axioms can be directly translated into DL axioms and they can be easily expressed by means of a set of RDF triples. This property is an essential one, since OWL was also required to have RDF/XML exchange syntax, because of its connections with the Semantic Web.

Given the huge number of requirements for OWL and the difficulty of satisfying all of them in combination, three different versions of OWL have been designed:

- OWL-Lite is the simplest variant for building a basic frame system (or an object oriented database) in terms of class, property, subclass relation, and restrictions. OWL-Lite does not use the entire OWL vocabulary and some OWL terms are used under certain restrictions.

- OWL-DL is grounded on DL, and focuses on common formal semantics and inference decidability. Description logics offer additional ontology constructs (such as conjunction, disjunction, and negation) besides class and relation, and have two important inference mechanisms: subsumption and consistency. Horrocks and Sattler [71] argued that basic inference in most variations of DL is decidable with complexity between polynomial and exponential time. The strong Set Theory background makes DL suitable for capturing knowledge about a domain in which instances can be grouped into classes and relationships among classes are binary. OWL-DL uses all OWL ontology constructs with some restrictions.

- OWL-Full is the most expressive version of OWL but it does not guarantee decidability. The biggest difference between OWL-DL and OWL-Full is that class space and instance space are disjointed in OWL-DL but not in OWL-Full. That is, a class can be interpreted simultaneously as a set of individuals and as an individual belonging to another class in OWL-Full. The entire OWL vocabulary can be used in without any restrictions in OWL-Full.

2.3.2 Ontology Tools

Ontology is constructed by suitable languages with the help of effective supporting tools.

2.3.2.1 Ontology Editors

A good editor can save a significant amount of time when developing ontologies by helping ontology engineers focus on the semantics without worrying much about syntactic organization [33]. This section offers a brief introduction to some popular ontology editors for collaborative (or independent) ontology development.

Protege [48] provides a standalone ontology development environment. It is highlighted by its syntax grammar independent user interface and pluggable infrastructure. It is suitable for independent ontology development and has a large user community. SWOOP [114] takes advantage of both Protege and Ontolingua [42] and provides a convenient web-based ontology browsing, editing, debugging [145] and publishing interface.

2.3.2.2 Ontology Repositories

Although the Web improves the visibility of centralized ontology development, it is hard to achieve a universal ontology for everything (e.g. Cyc) due to huge space complexity. Hence, distributed ontology development is preferred in the Semantic Web, i.e., small ontologies are authored by different sources in an incremental fashion. To reuse existing ontologies, effective web-based tools are in great need to browse, search and navigate distributed ontologies. The popular repositories for publishing and searching ontologies on the Web are detailed below.

- *DAML Ontology Library*

  DAML ontology library [91] indexes user submitted ontologies and provides browse/search services. It organizes ontologies by their URI, users annotations supplied during ontology submission (e.g. submission date, keyword, open directory category, funding source and submission organization), the defined class/property, or the used namespace. Users can run sub-string queries over a defined class/property.

- *SchemaWeb*

  SchemaWeb [99] provides services similar to DAML ontology library with better human/machine user interface (i.e. both HTML and web service interface). It adds more services: i) for human user - it provides full text search service for indexed ontologies, and a customizable resource search interface by letting users specify triple patterns; ii) for machine agents - it searches the “official” ontology of a given namespace or the resource with the user specified triple patterns; it also navigates RDF graph through RDFS properties (i.e. sub- ClassOf, subPropertyOf, domain, range), and publishes RSS feeds about new ontology submissions.

- *W3C's Ontaria*

  W3C’s Ontaria [105] stores RDF documents (including ontologies) and provides search/navigation services in the repository. It allows a user to i) browse a RDF file as a list of triples, a list of used properties, or a list of populated classes, and ii) browse relations between RDF files.

- *SemanticWeb Search*

  SemanticWeb Search [101] provides an object oriented view of the Semantic Web, i.e. it indexes instances of well-known classes including *rdfs:Class*, *rdf:Property*, *foaf:Person*, and *rss:Item*. It partially supports ontology search by finding instances of *rdfs:Class* and *rdf:Property*; however, its search results are biased to terms from the namespace of *WordNet 1.6*.

- *Swoogle*

  Swoogle [88] indexes millions of Semantic Web documents (including tens of thousands of ontologies). It enables users to search ontologies by specifying constraints on document metadata such as document URLs, defined classes/properties, used namespaces, and RDF encoding. Moreover, it provides detailed metadata about ontologies and classes/properties in an object oriented fashion. It has an ontology dictionary that enables users to browse the vocabulary (i.e. over 150KB URIrefs of defined/used classes and properties) used by Semantic Web documents, and to navigate the Semantic Web by following links among classes/properties, namespace and RDF documents. In addition, it is powered by automatic and incremental Semantic Web document discovery mechanisms and updates statistics about the use of ontologies in the Semantic Web on a daily basis.

2.3.2.3 Ontology Language Processors

An ontology construct expresses descriptive semantics, and its actionable semantics is enforced by inference. Hence, effective tools, such as parsers, validators, and inference engines, are needed to fulfill the reasoning capability. A detailed developers' guide is available online [108], and experimental evaluation can be found in W3C's OWL Test Cases report [104].

- *OWLJessKB* [73] is the descendent of DAMLJessKB [120] and is based on the Jess Rete inference engine.

- *Java Theorem Prover (JTP)* [95], was developed by Stanford University. It supports both forward and backward chaining reasoning by using RDF/RDFS and OWL semantics [51].

- *Jena* [80] was developed by HP Labs at Bristol [22]. It is a popular open-source project, which provides sound and almost complete (except for blank node types) reasoning support for RDFS. The latest version of Jena also partially supports OWL reasoning and allows users to create customized rule engines.

- *F-OWL* [79] was developed by UMBC [195]. It is an reasoning engine which is based on Flora-218.

- *FaCT++* [83] was developed by Manchester University [177]. It is the descendent of FaCT [70] reasoning system, which provides full support for OWL-Lite. And the latest version complete support for OWL-DL reasoning and partially for OWL 2.

- *Racer* [102] is a DL-based reasoner [58]. It provides reasoning over RDFS/DAML/OWL ontologies through rules explicitly specified by the user.

- *Pellet* [96] was developed by the University of Maryland. It is a 'hybrid' DL reasoner that can deal both TBox reasoning as well as non-empty ABox reasoning [166]. It is used as the underlying OWL reasoned for SWOOP ontology editor [114] and provides in-depth ontology consistency analysis.

- *TRIPLE* [89] was developed by Sintek and Decker [165]. It is a Horn Logic based reasoning engine and employs the same features of F-logic. Different with F-logic, it does not support fixed semantics expression for classes and objects. This reasoner can be used by translating the DL-based OWL into a language (named TRIPLE) handled by the reasoner. Extensions of DL that cannot be handled by Horn logic can be supported by incorporating other reasoners, such as FaCT, to create a hybrid reasoning system.

- *SweetRules* [87] is a rule toolkit for RuleML. RuleML has highly description capability, which is based on courteous logic programs. It provides additional built-in semantics to OWL, including prioritized conflict handling and procedural attachments [87]. The SweetRules engine also provides semantics preserving translation between different languages and ontologies (implicit axioms).

2.4 Analysis and Conclusion

The literature review is conducted in respect of the theories, the technologies, the approaches and the tools that related to the semantic-based component specification and retrieval. It starts from the software reuse in which the features of the mainstream reuse approaches are introduced and are analyzed according to the criteria including the development schedule, the expected software lifetime, the criticality of the software and its non-functional requirements, the application domain and the platform. The existing software reuse approaches have their own advantages and applicable conditions. In fact, among them there are no clear boundaries. Often there are two or even many methods are used together during software reuse development.

And then, the review focuses on the component-based development in which the key steps are expressed, including component certification, component specification and retrieval, and component adaptation. Through the review, it was found that the research of CBD is centred on finding appropriate formal approaches for describing components, the architectures for composing them, and the methods for component-based software construction. Through over 20 years of continuous improvement, CBD approach has been increasingly used in the software development. The main features are: 1) the basic CBD framework has been widely accepted, in particular the emergence of a large number of component-based models; 2) according to the increasing user requirements, many new methods continue to arise and improve for each step of CBD; 3) a large number of components are provided from the gradually mature open source or commercial component markets; 4) human interaction during the CBD is decrease with the help of the development tools.

Currently, the CBD is still a key research area, one of the important reasons is the component mismatch problem is still not well addressed, even many traditional approaches are proposed for the component specification and retrieval (as mentioned in section 2.2.2 and 2.2.3).

In the recent years, the ontology is applied to improve the precision of the component specification and retrieval, which has become a new hotspot. In the third section, the review introduces the ontology and its related issues, such as ontology languages, ontology editors, ontology repositories and ontology language processors. Ontology describes the aspects of the real world that can be used multiple times in different applications. The description capability and reasoning capability can improve the search effectiveness in the process of component specification and retrieval, and reduce the impact of the human intervention by the following reasons: 1) ontology provides a mechanism to represent and store domain specific knowledge, which can be used to find the most related components; 2) the relationships of ontology can exploit the additional knowledge embedded in domain ontology to augment and correct the user requirements; 3) the formal definitions of the ontology can provide the high degree of automation with the help of ontology tools in the component specification and retrieval.

The completed literature review benefits the research in the thesis with the following conclusions:

1) Software reuse is a widely-used comprehensive approach to improve the efficiency and to reduce the costs of the development of various software systems.

2) CBD is one of the most popular approaches of the software reuse by the reason of components are the most suitable and widely used assets for software reuse.

3) The component mismatch is a persistent and difficult problem, which limits the application and development of the CBD.

4) To solve the mismatch problem, ontology is introduced to help understand the semantics of components. Ontology is useful because it provides a means of understanding what the components mean in both the domain model and reuse repository.

5) The OWL-DL has the potential to define the contents of the used ontology by the reason of its appropriate description and reasoning capability.

6) Ontology is constructed conveniently by OWL-DL with the help of relevant effective supporting tools.

3. Related Work

As mentioned in the literature review, the mismatch between requirements and selected components has existed as a rather persistent problem in Component-Based Development (CBD), and is getting increasingly severe with the emergence of modern software systems and the evolution of CBD. At an early stage, software developers modify the code of accessible components to satisfy the requirements. This approach is known as white-box reuse, which is applicable to local repositories and incurs much of the cost in making the changes. Thereafter, most local repositories extend to external or even global markets, and components are usually reused “as is”, i.e. without changes to the code. This type of reuse is known as black-box reuse. As the investigation in [10][57][137] shows, more and more component venders put their components on the Internet, which gradually forms numerous online component agencies (component repositories). The representative websites [174] include ComponentSource, Flashline, Buydirect, Brattbery, and Findcomponents. The growing component markets raise thirteen conditions [176] which affect CBD. Three of them play a crucial role in connection with overcoming the mismatch problem, including user query formulation, standard specification of components and component retrieval with search relevant rating. Because the user query formulation refers to the research area of natural language processing and Artificial Intelligence areas, etc. It is not the research topic.

Our research focuses on the last two issues, which are referred to as component specification and retrieval. The existing approaches to component specification and retrieval are classified into two types: traditional and ontology-based.

3.1 Traditional Component Specification and Retrieval Projects

The traditional project is that one or more traditional component retrieval approaches are used. The traditional approaches include keyword searching [135], faceted classification [143], signature matching [192] and behavioral matching [193]. The keyword searching approach is used to component retrieval first in the early 1990's. The search engine compares users keywords to the names of the objects and retrieves matches. It does not take into account additional information such as relationships among objects or synonymous names. In order to improve the precise of the simple keyword search, the other three approaches were developed. The faceted classification approach makes an attempt to classify the objects in the repository based on predefined taxonomies. It is useful for objects that can clearly fall into such categories. Signature matching focuses on the type and number of arguments defined for methods and in essence takes an indirect approach to identifying whether an object is relevant. Behavioral matching is the most involved approach because it takes into consideration the functional behavior of objects. In this approach, objects are provided with input vectors and the outputs generated are compared to the expected outputs. Objects that exhibit a certain behavior are retrieved and presented to the user. In the late 1990's, the traditional approaches had been developed to mature, and widely used in the process of component specification and retrieval.

Typical examples of component specification and retrieval projects using the traditional approaches includes: Agora, Morebased, Maracatu, Zhuge and Zaremski.

- *Agora*

  Agora [161] is component search research project at the Software Engineering Institute of Carnegie Mellon University. It supports two basic processes: the location and indexing of components and the search and retrieval of a component. Agora uses a JavaBeans agent and a CORBA agent for locating and indexing component information. Once a component has been identified, the interface information is decomposed into a set of tokens and is saved in a document. Agora combines introspection with Web search engines to look for components in the software marketplace with the help of keyword search and faceted classification approaches. The query keywords and options specifying the types of components, are searched against the index collected by the search agents. Each result includes meta-information e.g. the URL of the component. Moreover, application-specific lexicons are integrated in Agora, which can be used to facilitate searches in application domains such as manufacturing, healthcare, and finance.

- *Merobase*

Merobase [82] is a search and tagging engine that allows users to find, remember and share components on the Internet. In contrast with first-generation code search engines, Merobase treats source code modules first as class abstractions rather than chunks of text, and is thereby able to offer a much wider range of search options. In particular, Merobase specializes in finding components based on their interface (or API) rather than the strings in their source code. Merobase supports four basic kinds of searches. Using the Merobase Query Language (MQL) users can look for components that have a particular name or contain a given string in their source code. The user can search for components that represent a particular logical abstraction - either a function-oriented abstraction or an object-oriented abstraction. In addition, Merobase supports a wide range of constraints which allow users to narrow down their search to components with a particular set of properties.

- *Maracatu*

Maracatu [46] is developed jointly by Recife Center for Advanced Studies and Systems (C.E.S.A.R) and institute of Mathematical and Computing Sciences of

San Paulo University. It is a search engine for retrieving source code components from development repositories. The tool is structured in a client-sever architecture: the client side is a plug-in for the Eclipse IDE, while the server side is represented by a web application responsible for accessing the repositories in the Internet or Intranets. Two versions of the engine were developed, with new features being added for use in industrial practice. To demonstrate the tool, two experiments are presented for comparing the text matching mechanism (first version) with the facet mechanism implemented in the last version. The experiment showed that the facet-based mechanism alone does not have good performance but, when combined with text-based search, is a better overall solution.

- *Zhuge's project*

Zhuge [194] proposed a problem-oriented and rule-based component repository. Based on the repository, a framework was designed to describe a component's behavior by problem-solving mechanism, with which it is possible to assure that the retrieved components will fulfil the requirements specified in the query. This framework supports the component search tool in two aspects. Firstly, the reuse is developed from the component level to the problem-solving level. Users are encouraged to concentrate on the problem description and solve the problem through top-down refinement rather than plunge into the technical details at the beginning. Secondly, the environment evolution is during running. The candidate components for composing an application are simulated through case-based rule reasoning. The candidate components are dynamically checked before composing an application system.

- *Zaremski's project*

Zaremski [193] developed a component specification matching tool on the basis of analyzing component foundational definitions. It is intended to use as much information associated with the description of software components as possible by describing their signatures, behaviours and so forth. With this tool, the definitions were explored for applying specification matching to various applications. Although the idea of specification match was originally initiated by the software library retrieval application, it can also be applied in other areas of software engineering.

3.2 Ontology-based Component Specification and Retrieval Projects

Traditional component search is not effective for component selection, suffering from lower recall and precision, i.e., poor completeness and accuracy of components matching [171]. Traditional approaches are rather limited in accommodating semantics of user queries and domain knowledge. To solve this problem, ontologies have been introduced to help understand the semantics of components, from the early 2000's. Here some typical ontology-based approaches that specially focus on component specification or involve the whole process of the component specification and retrieval are analyzed as follows. The methods based only on component specification are given first.

- *Girardi's*

  Girardi's [50] project first developed GRAMO, which is a technique for the construction of domain and user models to be reused in the development of multi-agent applications. To support the GRAMO, an ontology-based meta domain model ONTODUM is built to represent the knowledge of techniques for the specification of the requirements of a family of multi-agent systems in an application domain. The goal of the project is the specification of a methodology for Agent-based Domain Engineering, and a framework for ontology-based specification of agent-based reusable artefacts, exploring both compositional and generative approaches. In the approach, domain models are being used as the main resources for the construction of Domain Specific Languages [49][162]. The advantages of using the ONTODUM for the representation of reusable products have been shown in a software development environment for Multi-agent Domain Engineering. However, it should be redesigned according to the particular knowledge of those development techniques.

- *Sugumaran's*

Sugumaran's approach [171] enables a user to execute more intelligent queries by using domain knowledge and natural language parsing techniques. The approach includes a natural language interface, a domain model and a reusable repository. This approach introduces domain ontology to exploit the additional knowledge to augment or revise a user's initial query. In this way, the ontology serves as a surrogate for the meaning of terms so that a more complete response to a user query will be produced. However, the domain ontology in this approach is too simple, which only covers quite limited semantic information; the gauge of relevance in component retrieval is not covered; No ontology-based component description method is presented to specify the component; and the prototype tool needs much manual work to operate, therefore further improvement in automation is needed.

- *Pahl's*

Pahl [144] investigates how ontology technologies can be utilised to support software component development. A link between modal logic and description logics proves that the provision of reasoning support for component behaviours, which is essentially characterised by the component's interaction processes with its environment and by the properties of the individual operations requested or provided in these interactions. The objective of the project was to provide reasoning support for semantically described components. Some shortcomings limited the realization of their objective, e.g., the architecture of the software development ontology is not as comprehensive as necessary to cover the components' attributes; the relationships provided for the reasoning are not adequate to implement the necessary reasoning; no component matching architecture is proposed.

- *Braga's*

Braga [19][20] addresses the interoperability problem between component information repositories. In this approach, an integration layer is developed to help search and identify suitable reusable components. This layer is based on mediators and domain ontologies to provide the binding of different components to their domain concepts. To assist the identification of related components and their appropriate domain organization, each mediator encloses one domain ontology and provides the mapping to their respective repository of components. The mediation layer promotes domain information integration and provides mechanisms to translate component requests across ontologies. The limitation of this approach is that it only focuses on the business components, with no consideration for other kinds of components, such as GUI components, controller components and IT function components.

- *Liu Quan's*

Liu Quan's approach [126] presents a component description scheme in the OWL language. The schema illustrates the abstraction hierarchy of a component specification, which consists of several facets [62][126], such as Basic Information, Component Type, Component Function, Application Domain and Interface information. Each facet represents a group of characteristics of the component. Clearly the schema is not ontology-based and its semantics not computer-recognizable. The component description information is associated with an ontology to provide semantic meaning, which builds the foundation for semantic reasoning in the component retrieval process. In Liu's work, the component description schema has no rules or individuals in it, hence it is not a component specific ontology yet. The gauge of relevance in component retrieval is not covered and neither is there an implementation and evaluation to prove the approach.

- *Yao's*

Yao [190] treats a software component as a service described in semantic service representation format and enhances the retrieval by semantically matching between the semantic representation of a user query and component description against domain ontology. A domain ontology-based matchmaker compares a user query in a conceptual graph with the component service descriptions in other conceptual graphs. In addition, the component search does not happen just in the local repository, it extends to the World Wide Web. A conceptual graph method is proposed to indicate the relevance between the user query and the result components. Its main drawbacks include the fact that both the details of how to analyze the user query and component specification based on semantics and the translation of the query into WSDL/RDF are not available. Similar to Liu's approach, the details of the domain ontology architecture and the linkage technique are not given. Furthermore, there is no implementation to validate the approach.

- *Yen's*

Yen [45][191] proposed an On-line Repository for Embedded Software (ORES) that uses an ontology-based approach to facilitate repository browsing and effective search. It provides effective component retrieval and facilitates the capture of component properties. An integrated mechanism was presented to facilitate efficient and cost-effective embedded software development. However, the architecture of the ontology used in this approach is monolithic, which means too many classes with less relationships were enumerated in one single facet.

3.3 Limitations of Related Work and Conclusion

The existing approaches closely related to ontology-based component specification and retrieval have been analysed in the previous sections. From this literature analysis, we conclude that existing component specification and retrieval approaches failed to have a sound semantic model as their foundation, preventing them reach an adequately sufficient level of automation and comprehension of the semantics of components. Consequently, these approaches have the following drawbacks in the delivery of the desired aims:

1) the ontology models in existing approaches are all domain specific, therefore a generic computing-oriented overview is missing, often leaving the component retrieval in a unsystematic style and within too narrow a scope;

2) the architecture of the ontology in existing approaches is monolithic and has few relationships, which limits their semantic expressiveness;

3) the existing ontology-based specification and retrieval approaches presume that the domain ontology in use already exists; the method of domain ontology retrieval is not mentioned. Furthermore, the evolution of the domain ontology is not considered;

4) the search precision calculation method of the result components is missing or not efficient in the existing approaches. Therefore, the existing approaches can not

further distinguish which one matches the user requirements more for very similarity components.

5) the search of the possible adaptive assets/methods are not considered in the existing approach. Lack of the semantic description, the possibility of using the adaptive assets/methods is overlooked.

6) the display of the result components is ineffective. Existing approaches just show the basic search result such as the name of the result component, the search precision, and the text-based result component specification. However further details of the search result are missing, such as the search paths of each user query keywords, the percentage of the matched keywords and the adaptive searching result. This information is also important for the users to make the final decision.

4. The Approach

4.1 Overview

By investigating the literature review and the related work, one critical problem of CBD is that the most suitable component in the repository can not be retrieved automatically, in particular for large complex software applications and from a huge collection of very diverse or sometimes very similar components. These problems not only prevent CBD from reaching its full potential, but also hinder the acceptance of many existing large component repositories. The repositories here refer to the enterprise-grade local component repository, mature open source and commercial component markets [10][137][174]. These repositories have the following features: 1) large size, 2) great variety and 3) large number of similar components. For example, the Componentsource commercial repository locates more than 10,000 components which are classified into more 120 function type and other categories. In some frequently-used function, it contains more 400 similar components.

To overcome the above problems, existing approaches have engaged a variety of technologies to support better component specification and retrieval, among which several recent research projects attempted to use domain models and ontologies in component retrieval [20][144][171][190][191]. Although these approaches reduce the severity of the problem to some extent, it is clear that the drawbacks mentioned in chapter 3 still exist.

To improve the state of art, a novel and complete ontology-based approach is developed and then fully realized for holistic and semantic-based component specification and follow-on automatic and accurate component retrieval. As the foundation of the proposed approach, a Multiple-Viewed and Interrelated Component Specification ontology model (MVICS) is first developed for component specification and repository building. The MVICS model provides an ontology-based architecture to specify components from a spectrum of perspectives; it integrates the knowledge of Component-Based Software Engineering (CBSE), and supports ontology evolution to reflect the continuous developments in CBD and components. A formal definition of the MVICS model is presented, which ensures the rigorousness of the model and supports a high level of automation of retrieval. Furthermore, the MVICS model has a smooth mechanism to integrate with a domain related software system ontology. Such integration enhances the function and application scope of the MVICS model by bringing more domain semantics into component specification and retrieval. Finally, based on the MVICS model and the domain model integration, a MVICS-based component repository and search tool has been developed. The MVICS approach supports semantic-based component matching and adaptive component matching; it is fully automated, and presents a comprehensive profile of the result components instead of merely a value of relevance. The result of retrieval includes not only the matching components but also accurate relevance rating and unsatisfied discrepancy, which are presented to CBD engineers in the component matching profile.

4.2 The MVICS Framework

The framework of the MVICS approach is shown in Figure 4.1, which defines the architecture of its key elements. It consists of four key parts, namely query requirements collection, ontology construction, repository construction and component retrieval.

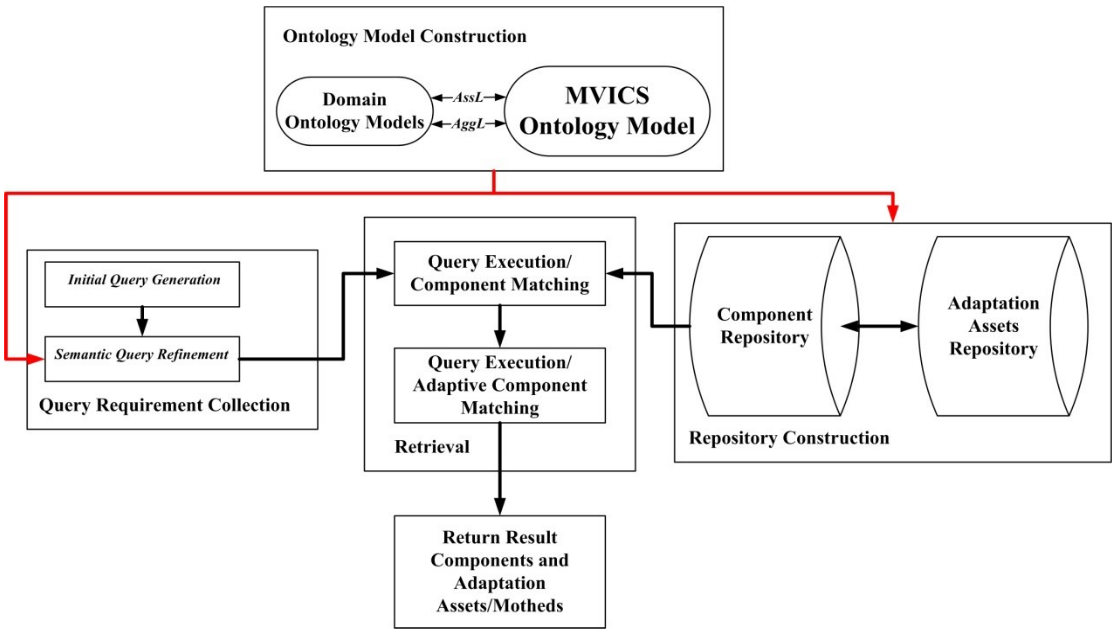

Ontology Model Construction
Domain Ontology Models
AssL
AggL
MVICS Ontology Model
Initial Query Generation
Semantic Query Refinement
Query Requirement Collection
Query Execution/ Component Matching
Query Execution/ Adaptive Component Matching
Retrieval
Component Repository
Adaptation Assets Repository
Repository Construction
Return Result Components and Adaptation Assets/Motheds

Fig. 4.1. The framework of the MVICS approach

4.2.1 Query Requirement Collection

To retrieve a component with the ontology-based approach, the first step is to collect the user requirements correctly and then to represent it with the ontology semantics. This task can be accomplished in two steps: initial query generation and semantic query refinement. The architecture of the query requirement collection is shown in figure 4.2. After the two steps processing, the user requirements in natural language are translated to OWL format.

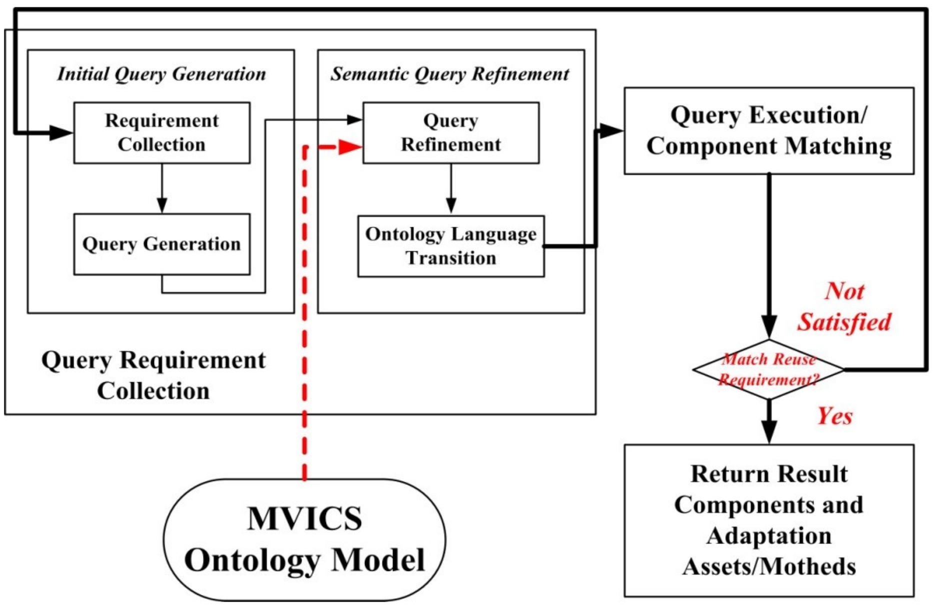

Initial Query Generation
Requirement Collection
Query Generation
Semantic Query Refinement
Query Refinement
Ontology Language Transition
Query Requirement Collection
Query Execution/ Component Matching
Not Satisfied
Match Reuse Requirement?
Yes
Return Result Components and Adaptation Assets/Motheds
MVICS Ontology Model

Fig. 4.2. The architecture of the query requirement collection

- *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 is 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. The heuristic-based approaches [30] have been researched for years, which focus on the natural language processing and Artificial Intelligence areas. In MVICS-based approach, we assume that the Initial Query Generation has been completed, as it is not the focus for our research.

- *Semantic Query Refinement*

  The keywords and concepts identified in the previous step are mapped against the MVICS ontology model to ensure that correct terms (keywords and concepts) of the requirements are used in the query. Then all the terms involved are 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 language are translated into OWL format without misunderstanding.

4.2.2 Ontology Model Construction

In the ontology construction part, the MVICS ontology model has a key role in the approach. It is the computer-recognisable ontological representation of the semantics of component specification. The MVICS is developed on the basis of CBSE knowledge and IT application domain knowledge. And it is managed by the ontology evolution mechanism. The role of the MVICS model includes supporting the semantic query refinement, conducting the component specification, connecting the domain ontology and supporting the sub-functions in the component retrieval, such as search precision calculation and adaptive component search and component QAs suggestion. The structure of the ontology model construction is shown in the figure 4.3. The domain ontology is retrieved from the ontology library and further connected to the MVICS by the *Association Link* and *Aggregation Link* when searching the relevant domain component. The details of the MVICS model and the domain ontology linkage technique are described in the Chapter 5.

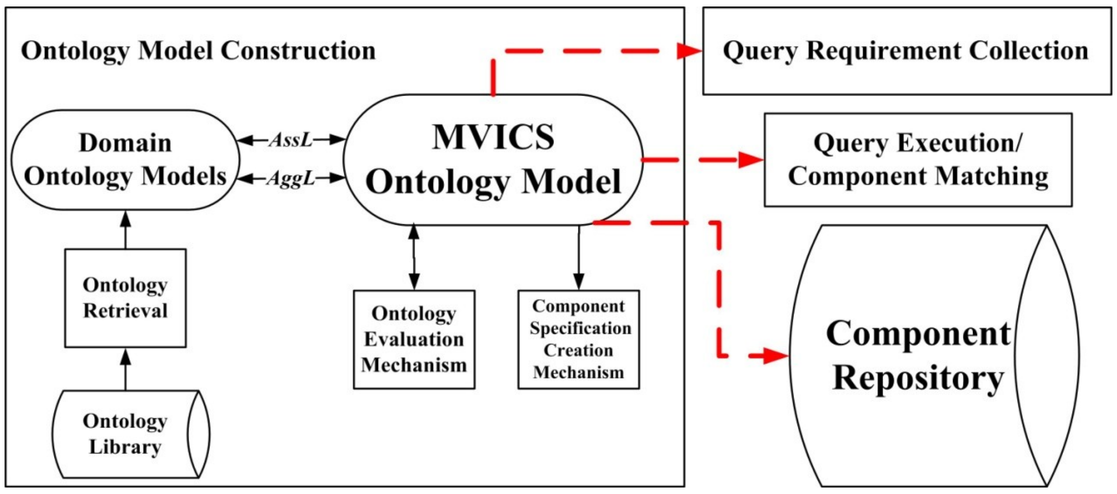

Ontology Model Construction
Query Requirement Collection
Domain Ontology Models
AssL
AggL
MVICS Ontology Model
Query Execution/ Component Matching
Ontology Retrieval
Ontology Evaluation Mechanism
Component Specification Creation Mechanism
Component Repository
Ontology Library

Fig. 4.3. The architecture of the query requirement collection

4.2.3 Component Repository

A component repository is built based on the MVICS model. Available components were specified according to MVICS model and populated into the repository. A component specification creation mechanism is developed to facilitate the specification process. For the component retrieval, the MVICS model and component specification is defined in OWL, corresponding with the OWL format user requirements. The MVICS approach also accommodates the impact of component adaptation with a new concept called “adaptive component search”. The available adaptation assets, their related adaptation method information and their impact on the targeted components are defined and stored in the component repository as an integral part of MVICS. The structure of the repository construction part shows in the figure 4.4.

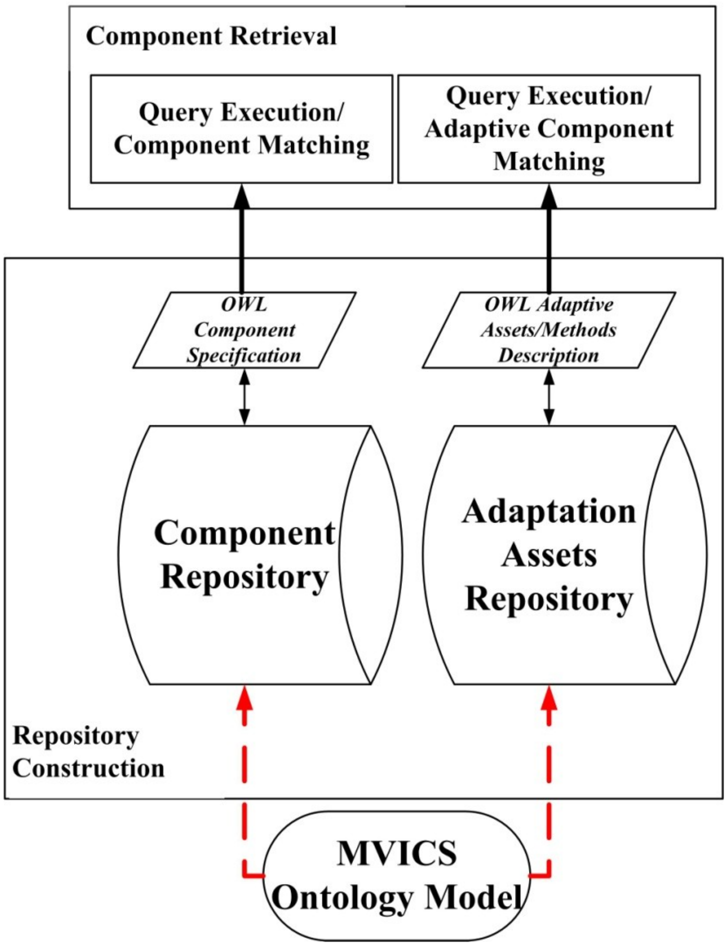

Component Retrieval
Query Execution/ Component Matching
Query Execution/ Adaptive Component Matching
OWL Component Specification
OWL Adaptive Assets/Methods Description
Component Repository
Adaptation Assets Repository
Repository Construction
MVICS Ontology Model

Fig. 4.4. The architecture of the repository construction

4.2.4 Component Retrieval

Having the above two phases in place, the component retrieval is thus regarded as the matching between the users query description and the ontological component specification in the retrieval (as shown in figure 4.1). Another unique feature of the component retrieval is the component adaptation information considered. Some of the discrepancies can be solved through the component adaptation. The available adaptive assets are stored in the Adaptation assets repository and an adaptive methods or assets suggestion mechanism is built to help the user to make the decision. With the help of the adaptation classes in the MVICS model, the components whose specification satisfies the user requirements after adaptation are indentified as part of the result as well.

The retrieval returns detailed search results, including not only the matching components but also accurate relevance rating (search precision) and unsatisfied discrepancy in descending order. Components with the same search precision will be presented in the descending order of the selected QAs. The precision indicates the level of relevance of the result components with a precision calculation algorithm and the QAs suggestion helps user avoid the neglect of the non-functional requirement.

All the results of the retrieval are presented to CBD engineers in a comprehensive component matching profile, which gives the user a full understanding of each result component and helps them to make a decision.

5. Multiple-Viewed Interrelated Component Specification Ontology Model (MVICS)

A holistic ontology model of component specification provides the foundation for effective semantic reasoning in the component retrieval and improves substantially the precision of component retrieval. The MVICS ontology model has a pyramid architecture, which contains four facets: function model, context model, intrinsic model and meta-relationship model, as shown in Figure 5.1. The first three models (function model, intrinsic model and context model) can be viewed as sub-ontology models, each of which describes one facet of component specification (locates at the three sides). The forth (meta-relationship model) is used to store four types of inter-relationships among the classes of the first three models (locates on the bottom). As a whole they construct a complete spectrum of semantic-based component specification. All the four models are ontology-based, extracted from the analysis of CBSE knowledge, and have extension slots for further upgrade according to the evolution of CBSE and components.

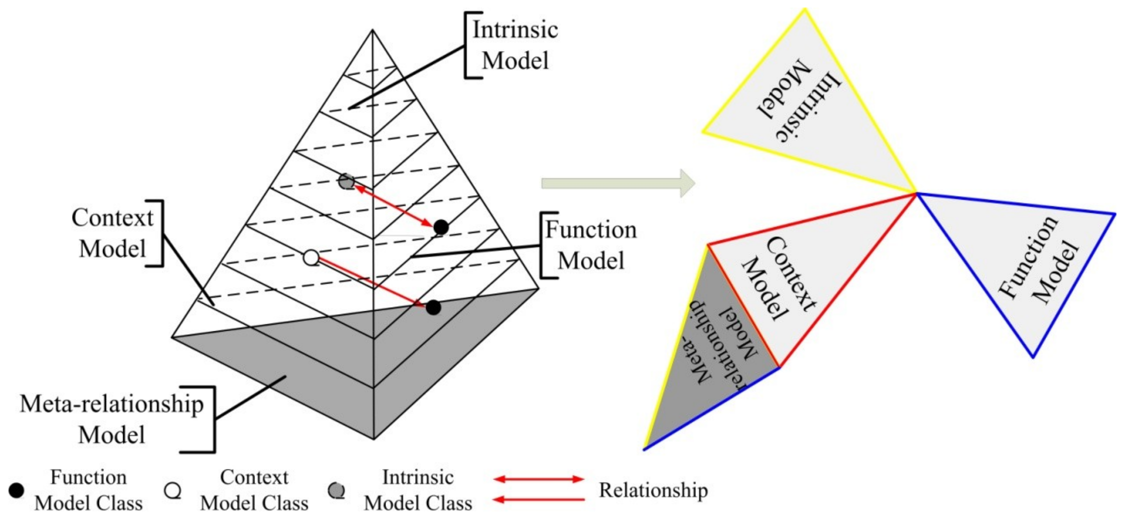


Fig. 5.1. Multiple-Viewed and Interrelated Component Specification Ontology Model

OWL-DL is adopted to define the classes, individuals and relationships of the above four sub-models. These formal definitions enable automatic ontology validation and semantic-based component search with the support of ontology reasoner.

5.1 Function Model

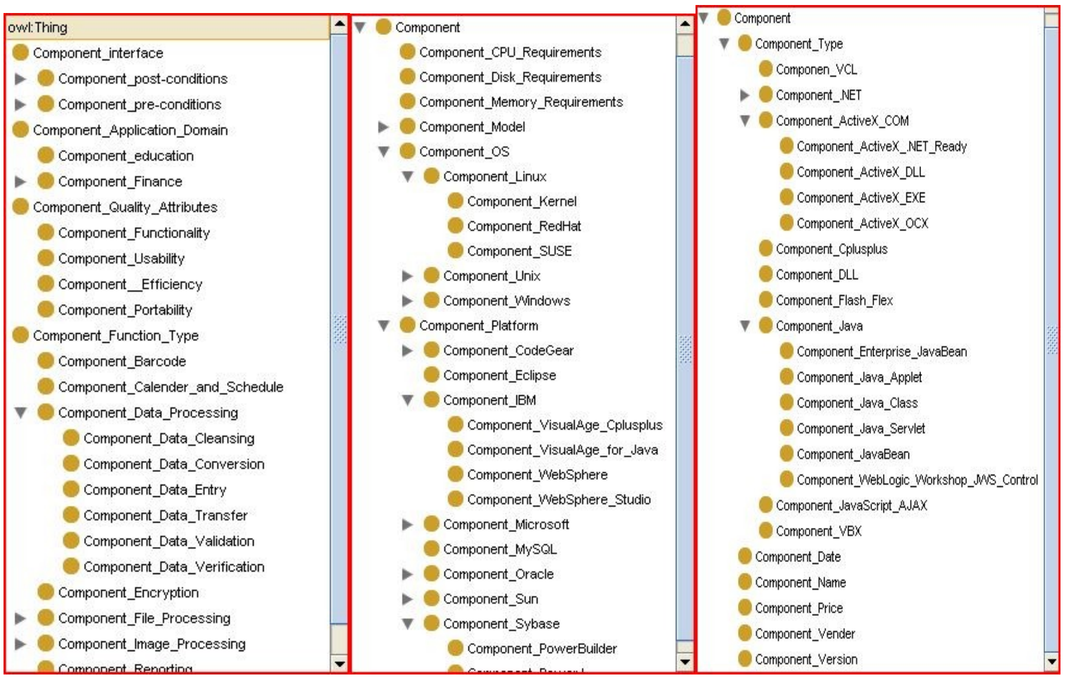
The function model specifies the function, domain information and interface. As an ontological model, the top level classes include *Function Type*, *Component Domain* and *Interface.*

Functions are performed by components which represent fundamental characteristics of software. The sub classes and sub sub-classes of *Function Type* are classified by the Computer Technology. Due to the fact that classes may overlap, the subclasses of class *Function Type* are defined in detail and are classified without any overlap (i.e., disjoint),e.g., *Data Cleaning*, *Data Conversion*, *Data Entry*, *Data Validation*, *Data Verification* and so forth.

Specific application domain ontologies can be interfaced with the function model as the subclasses of class *Component Domain*. These subclasses consist of two types *Association Link* and *Aggregation Link* which are used to link the domain ontology classes with related MVICS classes. The details of *Association Link* and *Aggregation Link* will be introduced in chapter 5.5.

Class *Interface* includes two composite sub-classes *Pre-conditions* and *Post-conditions*, each are then further composed of a set of method pre-condition and method post-condition according to the methods/procedures in the component. Take *Pre-conditions* as an example, its sub class "*Pre-conditions of method n*" has three subclasses including *Parameter Type*, *Parameter Range* and *Constraint*, in which class *Parameter Range* is to describe a range based on parameter type. To describe the parameter range accurately, number restriction constructor ($\leq$, $\geq$) in DL is required.

Figure 5.1 a) shows the top three levels of classes in the function model. The details of the function model (OWL format) list given in the appendix A.1.



```
owl:Thing
  Component_interface
  ▶ Component_post-conditions
  ▶ Component_pre-conditions
  Component_Application_Domain
    Component_education
  ▶ Component_Finance
  Component_Quality_Attributes
    Component_Functionality
    Component_Usability
    Component__Efficiency
    Component_Portability
  Component_Function_Type
    Component_Barcode
    Component_Calender_and_Schedule
  ▼ Component_Data_Processing
      Component_Data_Cleansing
      Component_Data_Conversion
      Component_Data_Entry
      Component_Data_Transfer
      Component_Data_Validation
      Component_Data_Verification
    Component_Encryption
  ▶ Component_File_Processing
  ▶ Component_Image_Processing
    Component_Reporting
```

```
▼ Component
    Component_CPU_Requirements
    Component_Disk_Requirements
    Component_Memory_Requirements
  ▶ Component_Model
  ▼ Component_OS
    ▼ Component_Linux
        Component_Kernel
        Component_RedHat
        Component_SUSE
    ▶ Component_Unix
    ▶ Component_Windows
  ▼ Component_Platform
    ▶ Component_CodeGear
      Component_Eclipse
    ▼ Component_IBM
        Component_VisualAge_Cplusplus
        Component_VisualAge_for_Java
        Component_WebSphere
        Component_WebSphere_Studio
    ▶ Component_Microsoft
      Component_MySQL
    ▶ Component_Oracle
    ▶ Component_Sun
    ▼ Component_Sybase
        Component_PowerBuilder
        Component_Power
```

```
▼ Component
  ▼ Component_Type
      Componen_VCL
    ▶ Component_.NET
    ▼ Component_ActiveX_COM
        Component_ActiveX_.NET_Ready
        Component_ActiveX_DLL
        Component_ActiveX_EXE
        Component_ActiveX_OCX
      Component_Cplusplus
      Component_DLL
      Component_Flash_Flex
    ▼ Component_Java
        Component_Enterprise_JavaBean
        Component_Java_Applet
        Component_Java_Class
        Component_Java_Servlet
        Component_JavaBean
        Component_WebLogic_Workshop_JWS_Control
      Component_JavaScript_AJAX
      Component_VBX
    Component_Date
    Component_Name
    Component_Price
    Component_Vender
    Component_Version
```

a) b) c)

Fig. 5.2. Top three level classes of a) function model, b) context model, c) intrinsic model

To define the model in OWL-DL, let $\top$ represent the top class. C_i^n, C_j^n, C_k^n represent classes in the nth level of the hierarchical architecture. The details are as follows:

$$C_i^1 \sqsubseteq \top$$

where i = *Function Type, Application Domain* or *Interface*

which defines that *Component Function Type, Component Application Domain* and *Component Interface* are the top level classes of the function model.

$C_i^n \sqsubseteq C_k^{n-1}$, which defines C_i^n is a subclass of C_k^{n-1}, for example,

$C^2_{DataConversion} \sqsubseteq C^1_{FunctionType}$, which states class *Data Conversion* is a subclass of class *Function Type*.

In the model, the subclasses of the same level of one class mutually disjoint.

If $C^n_i \sqsubseteq C^{n-1}_k$ and $C^n_j \sqsubseteq C^{n-1}_k$, then $C^n_i \sqcap C^n_j = \perp$, which defines that if C^n_i is subclass of C^{n-1}_k and C^n_j is subclass of C^{n-1}_k, then the intersection of C^n_i and C^n_j is null.

For example,

If $C^2_{DataConversion} \sqsubseteq C^1_{FunctionType}$ and $C^2_{DataEntry} \sqsubseteq C^1_{FunctionType}$,

then $C^2_{DataConversion} \sqcap C^2_{DataEntry} = \perp$,

which defines class *Data Conversion* and class *Data Entry* are disjoint on condition that they are the same level subclasses of class component type.

In the function model, all the top classes have a large tree type architecture of subclass, sub subclass and so forth. In this sub model, *isA* is the only relationship to link the classes and its subclass. In knowledge representation, *isA* is a relationship where one class *A* is a subclass of another class *B,* and *isA* is of a transitivity. The way to define the relationship with OWL-DL as follows:

For *isA* relationship, we assert that the range of the relationship is the respective class C_i^n:

$C_i^n \equiv \forall isA \, . \, C_i^{n+1}$, which defines C_i^n has an *isA* relationship with its subclass C_i^{n+1}, for example,

$C^1_{FunctionType} \equiv \forall isA \, . \, C^2_{DataConversion}$, which states the relationship *isA* links the class *Data Conversion* to the class *Function Type*.

5.2 Context Model

The context model is used to represent the reuse context information of the components, including but not limited to the application environment, hardware and software platform, required resources and possible dependency with other components. The top level classes consist of *Operating System*, *Component Container*, *Hardware Requiremen*t, *Software Requirement.* The context model is built in the same way as above two models, i.e., using isA to build ontology hierarchies of class operating system and class component container, and using isAttributeof to specify the value set of the attributes of the classes. Figure 5.1 b) shows the top three levels of classes in the context model. The details list in the appendix A.2.

The classes and relationships are defined in OWL-DL in the same way as that in the intrinsic model. The details are as follows:

$C_i^1 \sqsubseteq \top$

where i = *OS, Platform, CPU Requirements, Disk Requirements* or *Memory Requirements*.

which defines that *Component OS, Component Platform*, *Component CPU Requirements, Component Disk Requirements* and *Component Memory Requirements* are the top level classes of the context model.

$C_i^n \sqsubseteq C_k^{n-1}$, which defines C_i^n is a subclass of C_k^{n-1}, for example,

$C_{Windows}^2 \sqsubseteq C_{OS}^1$, which states class *Windows* is a subclass of class*OS.*

In the model, the subclasses of the same level of one class are mutually disjoint.

If $C_i^n \sqsubseteq C_k^{n-1}$ and $C_j^n \sqsubseteq C_k^{n-1}$, then $C_i^n \sqcap C_j^n = \perp$, which defines that if C_i^n is subclass of C_k^{n-1} and C_j^n is subclass of C_k^{n-1}, then the intersection of C_i^n and C_j^n is null.

For example,

If $C_{Windows}^2 \sqsubseteq C_{OS}^1$ and $C_{Linux}^2 \sqsubseteq C_{OS}^1$, then $C_{Windows}^2 \sqcap C_{Linux}^2 = \perp$,

which defines class *Windows* and class *Linux* are disjoint on condition that they are the same level subclasses of class component type.

Different from the function model, two types of relationships are used to show the links between the classes in different layers. *isA* relationship is used to describe super- and sub-class links between component type. *isAttributeof* defines the value set of an attribute of a class in the ontology model, e.g., *Memory Requirement* class is linked with a set of *Memory Value* classes under the "*isAttributeof*" relationship. The relationships are defined as follows:

For *isA* relationship, we assert that the range of the relationship is the respective class C_i^n:

$C_i^n \equiv \forall\, isA\ .\ C_i^{n+1}$, which defines C_i^n has an *isA* relationship with its subclass C_i^{n+1}, for example,

$C_{OS}^1 \equiv \forall\, isA\ .\ C_{Windows}^2$, which states the relationship *isA* links the class *Windows* to the class *OS*.

For the *isAttributeof* relationship, we assert that the range of the relationship is the respective attribute class $C_{attribute}^{n+1}$:

$C_i^n \equiv \forall\, isAttributeof\ .\ C_{attribute}^{n+1}$, which defines C_i^n has an *isA* relationship with its subclass $C_{attribute}^{n+1}$, for example,

$C_{MemoryReq}^1 \equiv \forall\, isattributeof\ .\ C_{MemoryValue}^2$, which defines the relationship *isAttributeof* links the class Memory Requirements to class *Memory Value*.

5.3 Intrinsic Model

The intrinsic model specifies the basic information of a component, including only its name, vendor, price, version, date and type. In the proposed approach, such information is defined as “intrinsic information” of the component. A taxonomy of the intrinsic information is developed first, which includes top level attributes such as *Component Name*, *Component Vendor*, *Component Price*, *Component Version*, *Component Dat*e and *Component Type*. All the attributes in this taxonomy are finally modelled as classes in the intrinsic ontology model of MVICS. Figure 5.1 c) shows the top three levels of classes in the intrinsic model. The details of the intrinsic model (OWL format) are listed in the appendix A.3.

The way to define the intrinsic model classes with OWL-DL is the same with the function model.

$C_i^1 \sqsubseteq \top$

where *i* = *Name, Vendor, Price, Version, Date* or *Type*,

which defines that *Component Name, Component Vendor*, *Component Price, Component Version* and *Component Type* are the top level classes of the context model.

$C_i^n \sqsubseteq C_k^{n-1}$, which defines C_i^n is a subclass of C_k^{n-1}, for example,

$C^2_{DLL} \sqsubseteq C^1_{type}$, which states class *DLL* is a subclass of class *component type*.

In the model, the subclasses of the same level of one class mutually disjoint.

If $C^n_i \sqsubseteq C^{n-1}_k$ and $C^n_j \sqsubseteq C^{n-1}_k$, then $C^n_i \sqcap C^n_j = \perp$, which defines that if C^n_i is subclass of C^{n-1}_k and C^n_j is subclass of C^{n-1}_k, then the intersection of C^n_i and C^n_j is null.

For example, If $C^2_{java} \sqsubseteq C^1_{type}$ and $C^2_{.NET} \sqsubseteq C^1_{type}$, then $C^2_{java} \sqcap C^2_{.NET} = \perp$,

which defines class *java* and class *.NET* are disjoint on condition that they are the same level subclasses of class component type.

For *isA* relationship, we assert that the range of the relationship is the respective class C^n_i:

$C^n_i \equiv \forall\, isA\, .\, C^{n+1}_i$, which defines C^n_i has an *isA* relationship with its subclass C^{n+1}_i, for example,

$C^1_{type} \equiv \forall\, isA\, .\, C^2_{DLL}$, which states the relationship *isA* links the class *DLL* to the class *component type*.

For the *isAttributeof* relationship, we assert that the range of the relationship is the respective attribute class $C^{n+1}_{attribute}$:

$C_i^n \equiv \forall$ *isAttributeof* . $C_{attribute}^{n+1}$, which defines C_i^n has an *isA* relationship with its subclass $C_{attribute}^{n+1}$, for example,

$C_{vender}^1 \equiv \forall$ *isattributeof* . $C_{vendername}^2$, which states the relationship *isAttributeof* links the class *vendor name* to class *component vendor*.

5.4 Meta-relationship Model

The Meta-relationship model provides a semantic description of the relationships among the classes in different facets (sub-models) of MVICS. Four types of relationships are identified. Let's define a relationship as $C_A \longrightarrow C_B$, where C_A and C_B are classes in different facets of the MVCIS model. To define these relationships in DL, we create a transitive rule R_{match}, which defines a matching relationship from C_A to C_B. It means that if C_A matches the requirement of a component search then C_B will match the requirement as well. The above four relationships are then defined as follows.

5.4.1 Matching Propagation Relationship

Matching Propagation Relationship, $C_A \xrightarrow{Pro} C_B$, it means that if C_A satisfies the requirement of a component search then C_B and all its subclasses will satisfy the requirement as well. In component retrieval, such a relationship will enable all the components under C_B and its subclasses to be part of the result components for a user query that is matched by C_A. The impact on the search path of this relationship is given in part a) of Figure 5.3. When the search engine identifies C_A as a match with the user search keyword K_1, it will continue to search for result components in the subclasses of C_A, and at the same time also identify C_B as a match. It would not continue the search in subclasses of C_B, because all the subclasses of C_B are deemed as matching.

For example, when users search for a component with keyword “IBM VisualAge”, the search will find the class *IBM VisualAge* in the context model. The instances linked (via its leafclass) to class *IBM VisualAge* will be recorded as result components. In the MVICS, class *IBM VisualAge* has a Matching Propagation Relationship with *Java* class and *C++* class which are subclass of *Component Type* class in the intrinsic model. With the semantic information provided by this relationship, the component type can be run by IBM VisualAge are Java and C++. This indicates that the result component obtained while the user query is matched class *Java* or class *C++* are also the result components when the user search keyword is “IBM VisualAge”. Therefore component type class *Java* and class *C++* in the intrinsic model can be seen as the subset of platform class *IBM VisualAge*.

The DL definition of matching propagation relationship is then as follows:

Relationship definitions: $C_A \equiv \sqcap \forall R_{match} . C_B$

Role assertions: $< C_A . C_B > : \quad \forall R_{match}$

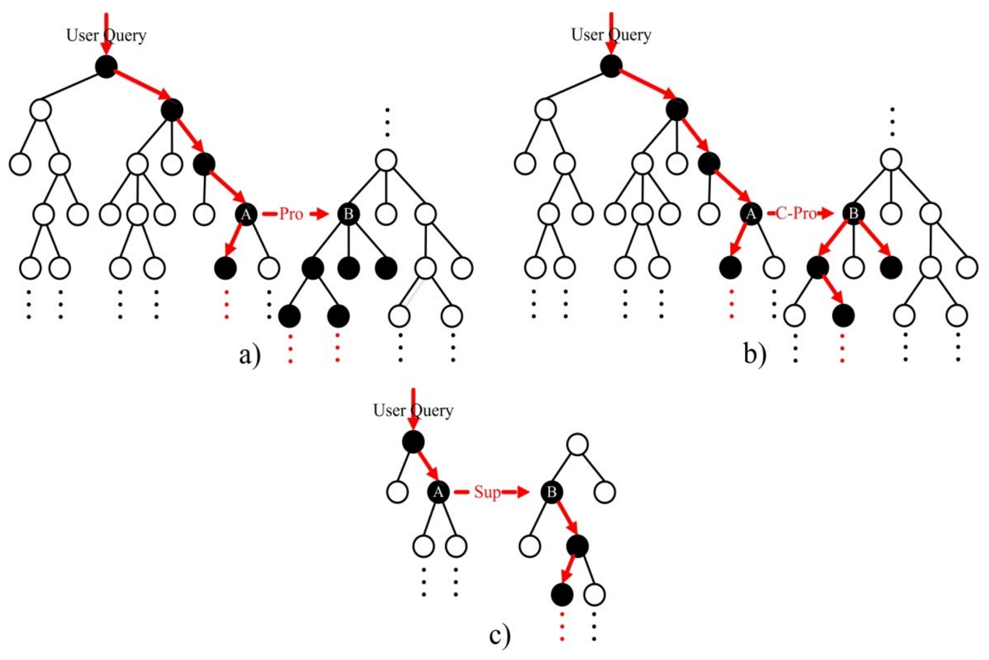


Fig. 5.3. The impact on search path: a) Matching Propagation Relationship; b) Conditional Matching Propagation Relationship; and c) Supersedure Relationship

5.4.2 Conditional Matching Propagation Relationship

Conditional Matching Propagation Relationship, $C_A \xrightarrow{C\text{-}Pro} C_B$ (attri=V), in MVICS, it means that if C_A satisfies the requirement of a component search then C_B and its subclasses may satisfy the requirement if their attribute attri has value V. In component retrieval, the relationship enables that the components under C_B are part of the result components for a user query that is matched by C_A, if their attri has value V. This relationship will impact on the search path as follows: when the search engine identifies C_A as a match with a user search keyword K_1, it will continue to search for result components in the subclasses of C_A, and at the same time search C_B and its subclasses on the condition of attri=V, as shown in b) of Figure 5.3.

For instance, when a user searches for a Silverlight type component by keyword "Silverlight", the search will find the class *Silverlight* in the intrinsic model. The instances linked (via its leafclass) to class *Silverlight* will be recorded as result components. In the MVICS, class *Silverlight* has the Conditional Matching Propagation Relationships with class *Component Type* in Context facet. With the semantic meaning specified by Conditional Matching Propagation relationship, we know that the compatible platform of Silverlight type component is Visual Studio 2008, Visual Studio 2005, Visual Basic 2008, Visual Basic 2005, Visual C# 2008 and Visual C# 2005. The compatible platform adds to class B as an attribute provide by the Conditional Matching Propagation Relationship. Then the search will continue to search the result components by the platform keywords in the Context model. Therefore the instances of class *Visual Studio 2008*, *Visual Studio 2005*, *Visual Basic 2008*, *Visual Basic 2005*, *Visual C# 2008* and *Visual C# 2005* in context model will be found as the result components of keyword" Silverlight" as well.

The DL definition of conditional matching propagation relationship is then as follows:

Relationship definitions: $C_A \equiv \sqcap \exists R_{match} . C_B$,

if $C_B \sqsubseteq C_V$, where C_V defines the classes have value V

Role assertions: $< C_A . C_B > : \exists R_{match}$

5.4.3 Matching Negation Relationship

Matching Negation Relationship, $C_A \xrightarrow{Neg} C_B$, in MVICS, this relationship means that if a result component is obtained by a keyword matching with C_A, then the result component cannot be obtained by another keyword matching with C_B. This relationship deals with problems caused by the incompatible requirements in a user query. When users input several keywords, C_A and C_B, which are matched with two different keywords respectively, they may have Matching Negation Relationship, i.e., a result component cannot belong to both classes simultaneously. To tackle this problem, the user query can be treated as two groups of keywords. One group consisting of the keyword matched with C_A, the other group consisting of the keyword matched with C_B. The Matching Negation Relationship is usually used for the user query refinement.

The matching negation relationship is defined in OWL-DL as follows:

Relationship definitions: $C_A \equiv \neg\forall R_{match} . C_B$

Role assertions: $< C_A . C_B > : \neg\forall R_{match}$

5.4.4 Supersedure Relationship

Supersedure Relationship, $C_A \xrightarrow{Sup} C_B$, in MVICS, Supersedure Relationship means that if the content of class C_B has higher priority to the content of class C_A, then the result components obtained by matching C_A will be replaced by the result components obtained by matching C_B. This relationship provides the following impact on the search path, as shown in c) of Figure 5.3: when the search engine identifies C_A as a match with the search keyword K_1, it will stop searching for result components in the subclasses of C_A, but turn to searching from C_B and its subclasses.

Up to now, this relationship is especially used to describe the relationship between class *Platform Hardware Requirements* (including *CPU* and *Memory*) and *Component Hardware Requirements*. The relationship shows the hardware requirements for platform, which need to be considered by users. It means if class *Platform* is matched with the user requirements, sometimes the relevant information of the platform hardware requirements will precede the result component hardware requirements. In this case, the component hardware requirements will be replaced by platform hardware requirements which need to be presented to users. To illustrate the Supersedure Relationship between class *Platform* and class *Hardware Requirements*, we take Microsoft Visual Studio 2008 as an example. Microsoft Visual Studio 2008 requires 384MB memory. Its memory requirements are beyond the result components'. In this case, the memory requirements of result component will change to 384MB.

The supersedure relationship is defined in OWL-DL as follows:

Relationship definitions: $C_A \equiv R_{match} . C_B$ if and only if $C_A \subseteq C_B$

Role assertions: < C_A. C_B > : R_{match} if and only if $C_A \subseteq C_B$

All the above four sub component specification ontology models are defined in OWL. These OWL documents can be seen as the paths that connect user queries and result components.

5.5 Linkage between Domain Related Software System Ontology and MVICS

The original MVICS model is a component specification ontology model based on the IT specific functions, rather than the application domain related functions and other features. Therefore this MVICS model does not support domain oriented component specification and retrieval. To extend the MVICS-based component search into a specific domain, two mechanisms, namely *Association Link* (*AssL*) and *Aggregation Link* (*AggL*), are developed to integrate the domain related software system ontology into MVICS. With such integration, the domain ontology is linked to MVICS effectively and thus extends the application scope of MVICS without changing the architecture of the model. Different from traditional ontology mediation, the connection between MVICS and the domain related ontology neither uses the method of ontology merging to create a new ontology, nor uses the method of ontology mapping to make the same or similar ontologies to establish contacts. The two mechanisms *AssL* and *AggL* are used to link two different kinds of classes of domain ontology to the MVICS. And then, the *AssL* and *AggL* are defined formally and to automate the component search and repository building. In the function model of MVICS, the class *Component Domain* is set to interface domain ontology with MVICS. The *AssL* and *AggL* generated from the integration will be stored under the class *Component Domain*.

5.5.1 Association Class and Association Link

Those classes in the domain ontology which can be viewed as sub classes of a MVICS class are named as "*Association class*". The *Association classes* in the domain ontology represent specific operations, as a specialization of their MVICS super classes in the relevant domain. *Association Link* is used to link these classes with their super class counterparts in MVICS.

Figure 5.4 shows an example of *AssL*. The class *Document Log* in financial domain ontology can be viewed as a sub-class of the class *Document Processing* in MVICS, because logging is a specific of document processing in the financial sector, therefore the *Document log* link to *Document Processing* with *AssL*.

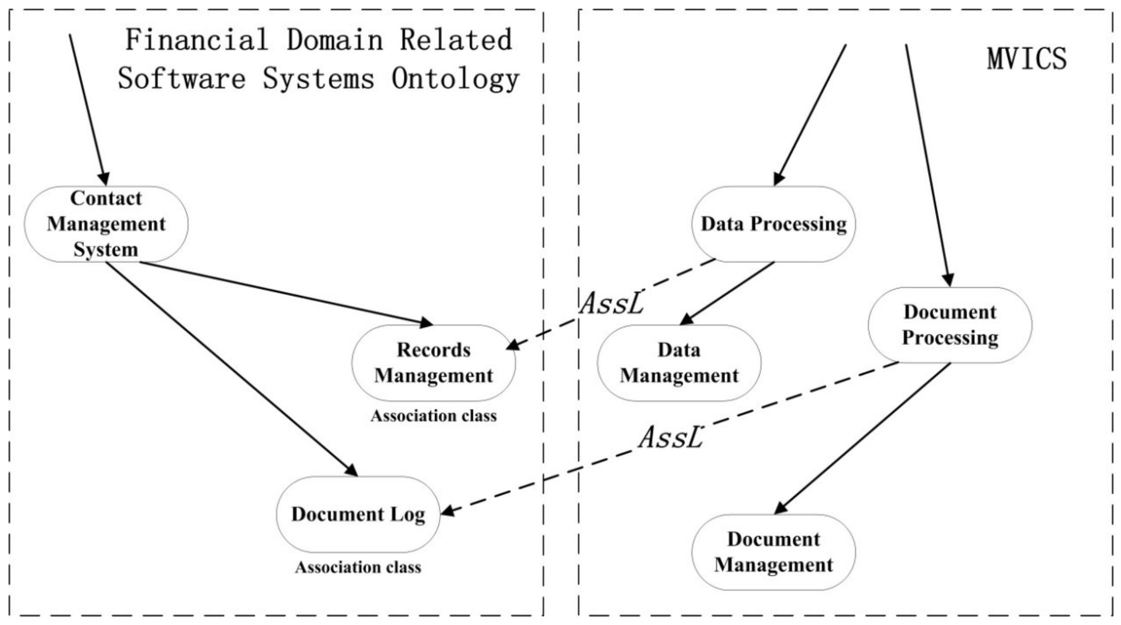


Fig. 5.4. An example of Association Link

5.5.2 Aggregation and Aggregation Link

Aggregations in MVICS are defined as a set of MVICS classes which work together to implement a larger function. Apart from the classes in the specific domain that can be linked in the way of *AssL*, other domain operation modules are more comprehensive and multifunctional. In this case, a set of reusable *Aggreg*ations in terms of the IT function in MVICS are first established, to represent the function of the domain operations. These reusable *Aggregatio*ns are the function units of MVICS with minimum intersection of reusable MVICS functions. Each *Aggregation* is viewed as a reusable unit oriented to different functional operations. To link a domain model with MVICS, an *Aggregation* Link is defined as a link from a domain class to an *Aggregation* in MVICS. Classes in the domain ontology are linked to MVICS through *AssL*, where a counterpart super/subclass relation exists, and/or *AggL*, where a domain class is composed of one or a set of *Aggregations* in MVICS.

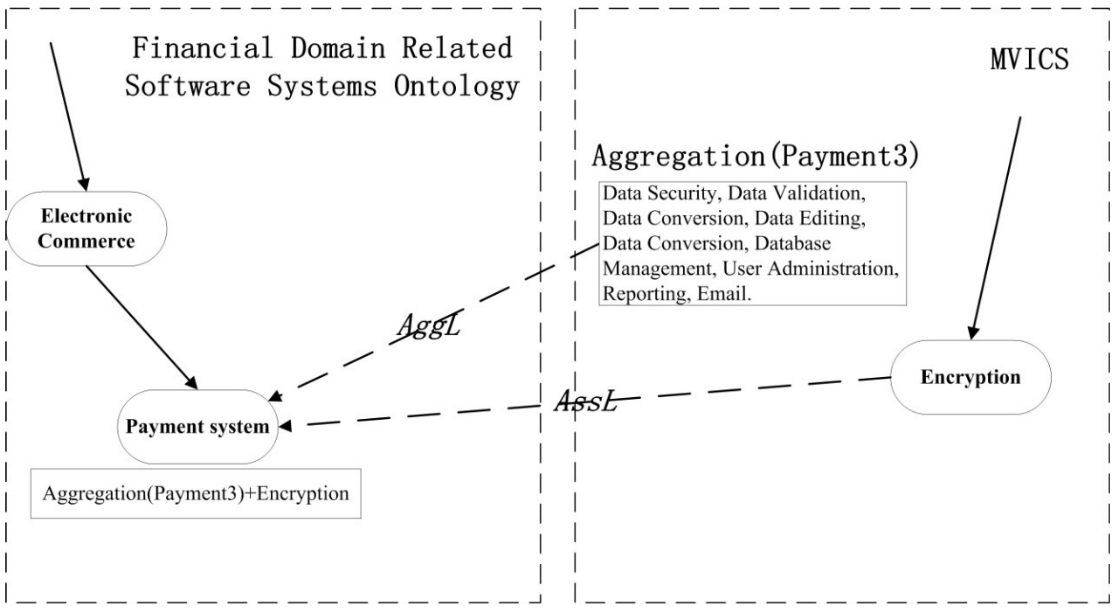

Financial Domain Related Software Systems Ontology
MVICS
Aggregation(Payment3)
Data Security, Data Validation, Data Conversion, Data Editing, Data Conversion, Database Management, User Administration, Reporting, Email.
Electronic Commerce
AggL
Encryption
AssL
Payment system
Aggregation(Payment3)+Encryption

Fig. 5.5. An example of *Aggregation Link*

An example of *AggL* is shown in Figure 5.5. In total, class *Payment System* in financial domain ontology has the following ten functions which are already defined in MVICS, including *Data Security, Data Validation, Data Conversion, Data Editing, Data Conversion, Database Management, User Administration, Reporting, Email* and *system Encryption*. The first nine functions are composed into an *Aggregation (Payment 3)* in MVICS, and linked with *AggL* to class *Payment System.* In addition to the above nine functions, the class *Payment System* has an extra function *Encryption*, and this is expressed with an *AssL* from class *Payment System* to the MVICS class *Encryptio*n. Thus, the whole function of class *Payment System* is expressed by a combination of the *AssL* and *AggL*.

Classes in the domain ontology are therefore linked to MVICS, either through *AssL* where a corresponding super/subclass relation exists, or through *AggL* where a domain class is composed of one or one set of *Aggregations* in MVICS. Via these two mechanisms, different domain ontologies can be integrated with MVICS by building new links of *AssL* and *AggL*. To automate the integration between domain ontology and MVICS, a domain ontology migration method was proposed. This method supports the semi-automation of the integration through the following process: Step 1: *Association class* identification; Step 2: operation module analysis; Step 3: *Aggregation* update. With the aid of a domain expert, the domain ontology can be integrated with MVICS more smoothly and effectively.

5.5.3 Linkage Definition

The linkage between the domain ontology and the MVICS are established by *Association Class, AssL*, *Aggregation* and *AggL*. Because the *Association Class* is a kind of domain ontology class, the definition of the *Association Class* is the same as the domain ontology class.

To define *AssL* in OWL-DL, let C_i^n represent a class in the nth level of the hierarchical architecture of the MVICS model. Let C_d^m represent a class in the mth level of the hierarchical architecture in the domain ontology. Let's assume there is an *AssL* relationship. The relationship is then defined as follows:

we assert that the definition of the relationship is the respective class C_i^n:

$$C_i^n \equiv \forall\, isA\,.\; C_d^m,$$

for example,

$C_{DT}^3 \equiv \forall\, isA\ .C_{MT}^3$, which defines the relationship *AssL* links the class *Money Transfer* in the domain ontology to the class *Data Transfer* in the MVICS.

To define *Aggregation* in OWL-DL, let C_i^n represent a class in the nth level of the hierarchical architecture of the MVICS model. Let C_A represent an *Aggregation* in the MVICS model. The *Aggregation* is then defined as follows:

$$C_A \equiv \bigcup_{i=1}^{m}$$

To define *AggL* in OWL-DL, let C_d^m represent a class in the mth level of the hierarchical architecture of the domain ontology. Let C_A represent an *Aggregation* in the MVICS model, we define the *AggL* relationship as that the linked class in domain ontology (C_d^m) must have an *IsA* relationship with at least one linked *Aggregation* (C_A) in MVICS.

$$C_d^m \equiv \forall\, isA\,.\; C_A$$

6. Holistic and Precise Component Retrieval Method

“Holistic” here refers to the fact that the MVICS is a comprehensive component specification model, and the approach considers a spectrum of respects in component specification and retrieval. As a core part, the MVICS-based component retrieval is achieved through original component search, domain component search, and adaptive component search. In addition, result component precision calculation and result component profiles, make the approach more complete to help the user find the more suitable components.

6.1 MVICS-based Component Retrieval

The MVICS component retrieval is based on the MVICS model and the linkage with the domain ontology model. It focuses on retrieving the relevant components from the repository according to the refined user keywords. For all components and their related adaptive assets located in the repository, their specifications are represented by the MVICS model and the related domain ontology in a semantic and logical way. The component name links to the relevant classes, which belongs to the MVICS and the linked domain ontology. The MVICS component retrieval will identify the matched classes with the refined user keywords. And it will retrieve the result components via the matched classes. As a unique feature of the MVICS-based approach, the adaptive components can be matched by identifying the adaptive search path. And the results provide not only the matched components with the relevant adaptation assets/methods, but also their suggested effort. The adaptive search results give users more options during the system development. Overall, four types of search paths are identified according to the location of the matched classes in the ontologies.

6.1.1 MVICS Component Search Path

The first type is original MVICS search path, which is generated by analysing the matched class in the function model, intrinsic model and context model of the MVICS. It starts from the matched class located in the sub model of MVICS, and ends with its top level class in the corresponding sub model. For example, the keywords "Windows XP" is matched with the class *Windows XP* in context model. The search path is from the beginning class *Windows XP* to the top class *OS* as shown in Figure 6.1.

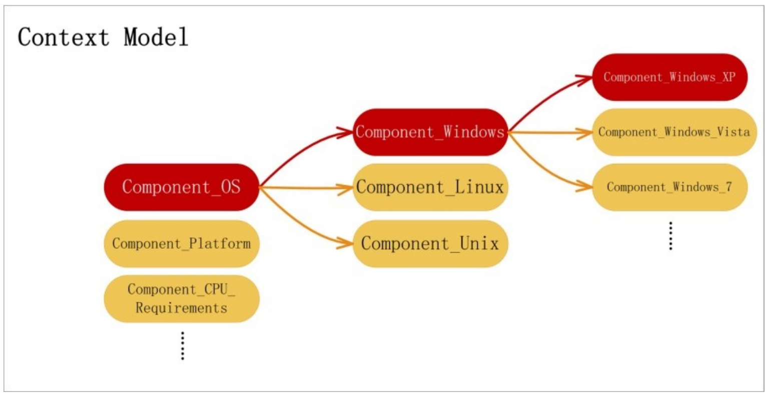


Fig. 6.1. Example of the original search path

6.1.2 AssL and AggL Search Path

The second and third types of search path are called domain *Association Link* (*AssL*) search path and domain *Aggregation Link* (*AggL*) search path, which are connected with domain specific keywords. In the MVICS-based component retrieval, the *AssL* and *AggL* are used to generate the domain related keywords search paths.

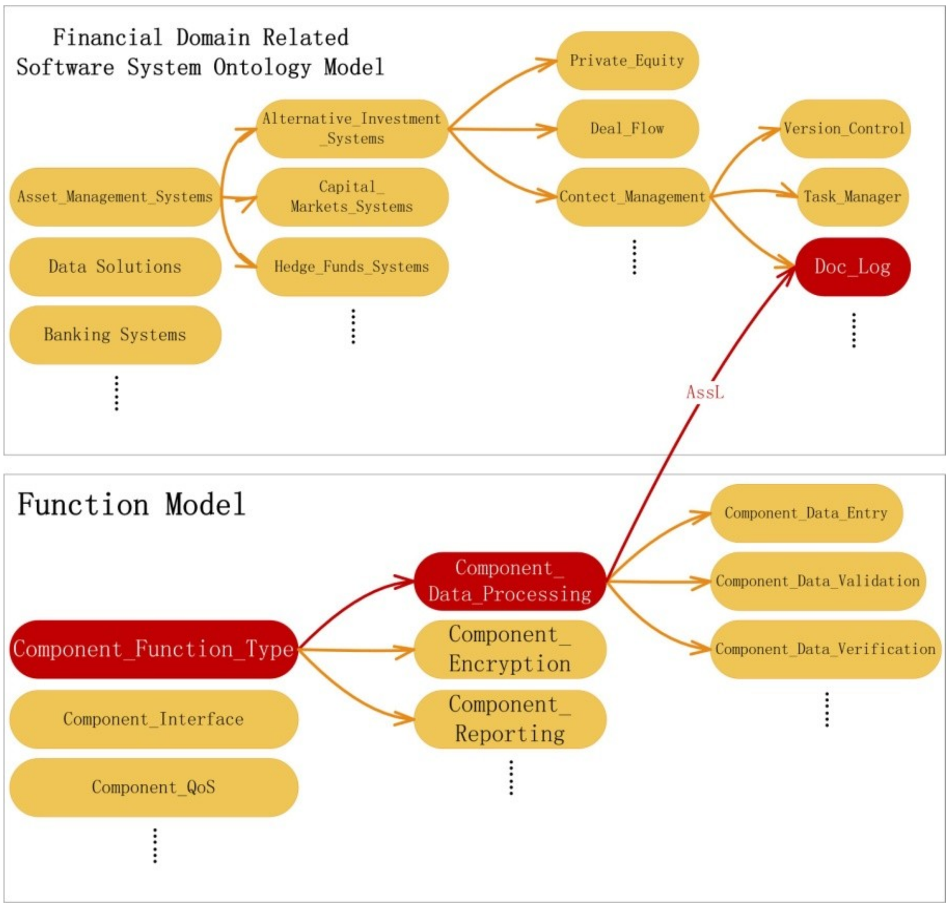


Fig. 6.2. Example of domain *AssL* search path

According to the definition, *AssL* is used to link the domain class (*Association class*) with their super class counterparts in MVICS. When the user keyword is matched with an *Association class*, a domain *AssL* search path is generated starting from this

Association class, linking with its super class in the MVICS by *AssL*, and ending with the top class in the corresponding sub model. Here we cite keywords “Document Log” as an example. As shown in Figure 6.2, the search path starts from the matched class *Document Log* in the financial domain ontology, and then connects to the class *Document Processing* by the *AssL*, finally it ends with the top level class *Function Type* in the function of MVICS.

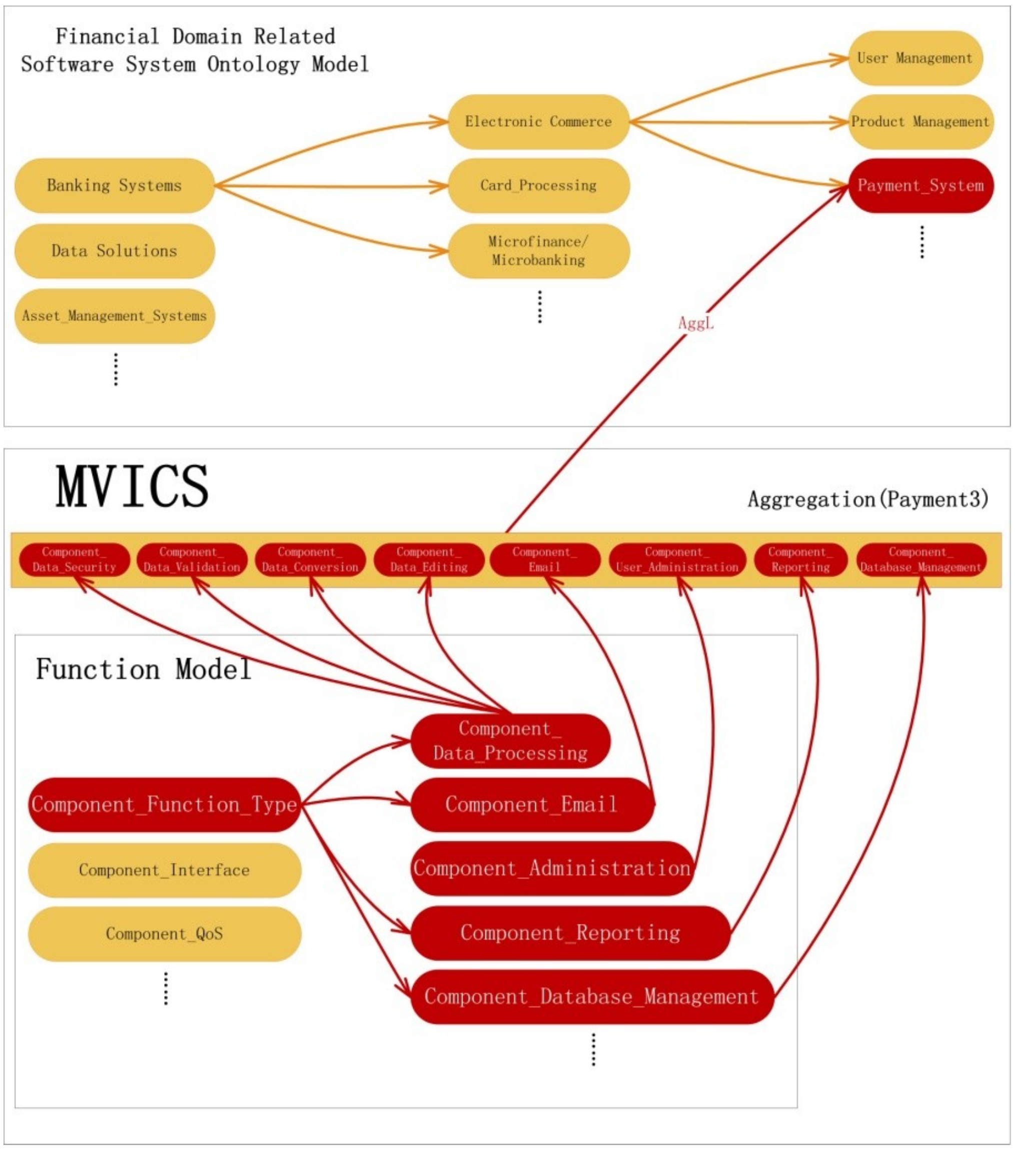


Fig. 6.3. Example of domain *AggL* search path

For the domain *AggL* search path, it is built in accordance with the definition of *AggL*, in the event that the user keyword matched with a domain ontology class, which links up with an *Aggregation* in MVICS model. The domain *AggL* search path can be counted as the set of original MVICS search paths, which are obtained by the original MVICS classes located in the *Aggregation*. For example, the class *Payment System* in the financial domain ontology is matched with the user keyword, and it links with the MVICS model through an *Aggregation (Payment 3)* as shown in 6.3. According to the definition of *Aggregation*, each class in the *Aggregation (Payment 3)* has an original MVICS search path in the MVICS model. Therefore, the search path of class *Payment System* is the set of search paths of these MVICS classes.

6.1.3 Adaptive Search Path

The last type is the adaptive search path, which is achieved during the adaptive component retrieval. In MVICS-based specification, we call those components whose function and QAs may vary via the application of adaptation assets “adaptive components”. The adaptive components are linked to a class via adaptation methods/assets if the component becomes relevant to that class after adaptation with that method or asset.

The retrieval path is then recorded as an adaptive path, in contrast to the direct path, i.e. without adaptation. In addition, an adaptation suggestion will give every matched methods/assets an effort suggestion. The effort suggestions are classified into three levels, which indicate the effort to complete the adaptation as Strong, Medium and Weak. The conclusion of method efforts comes from the analysis of the popular component adaptation methods in the literature review chapter. The degree of effort is given on basis of the two criteria: 1) whether software engineers need to have much related knowledge, and 2) whether they need to do many preparations for the task.

For each available adaptation methods/asset, the effort suggestion information is defined as attributes of the relevant classes, which are stored in the MVICS model as well.

It may not be possible for an adaptation mechanism to satisfy each requirement, since these criteria are drawn from disparate sources. Six popular methods are considered including Active Interface, Binary Component Adaptation, Inheritance, SAGA, Superimposition and Wrapping. By evaluating component adaptation mechanisms against these requirements, we can determine which existing adaptation methods need most effort. The following list presents the conclusions of method efforts:

Strong Effort:	*Active Interface, Inheritance, SAGA*
Medium Effort:	*BCA, Superimposition*
Weak Effort:	*Wrapping*

In the component retrieval stage, the search path that connects the user keyword to the matched class via the adaptation method/assets is recorded as an adaptive component search path. An example is taken as shown in Figure 6.4, the Components 1, 2, 3 that are located in the repository have no relationship with the class Data Security. However the components 1, 2, 3 whose function varies to meet the user query "data security" via the application of adaptation assets "Wrapper 1". After the adaptive search, the components 1, 2, 3 are identified as the result components for the query "data security" and their search paths are recorded as the adaptive search path which connected through the adaptive asset "Wrapper 1".

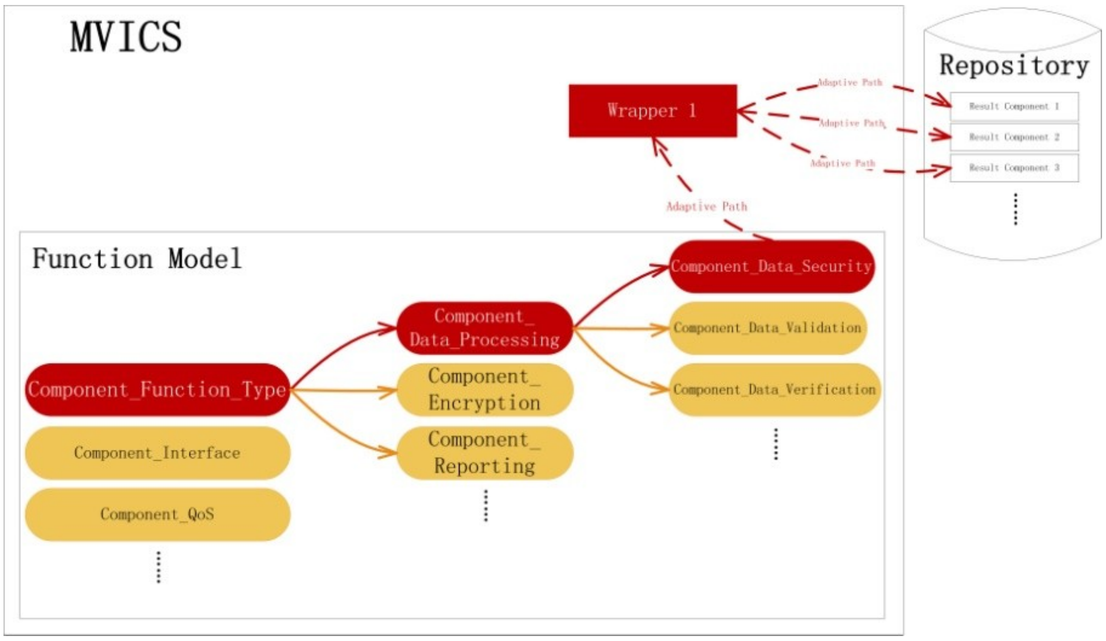


Fig. 6.4. Example of adaptive component search path

6.2 Result Component Precision Calculation Method

In order to obtain the high component matching precision, a precision calculation method is developed by analyzing the search paths of the result component. The Weight of class (*W*) is defined first as the foundation of the method.

6.2.1 Weight of Class

In each sub-model of MVICS, every class is given a weight to calculate the relevance of each search result. The rules of weight assignment are: i) In one facet, the lower a layer is the heavier weight its classes have; ii) In different facets, classes at the same depth in the function model are heavier than those in the intrinsic and context models. The weight assignment method is defined as follows:

$$W = (1+X)^{n} \quad (1)$$

where n is the level of the layer in which the class locates, $X = 0.5$ for class in the function model, $X = 0.3$ for class in the intrinsic model, $X = 0.2$ for class in the context model

The initial weights (X) of the classes in each model are given based on our experience and subject to continuing adjustment. According to the test data collected from the user, the initial weights will be updated dynamically after every 100 groups of user keywords are obtained. Each group of the keywords will be recorded and classified by the facets of the MVICS. The rules of dynamic original class weight assignment are: the more frequently the keywords are used in a facet, the heavier is the original weight of this facet. Let N represent the occurring times of the keywords in a facet, the subscripts *f, i, c* indicate the related facet (function, intrinsic or context). The weight assignment rules are defined as follows:

$$X_f = 0.5 \times \frac{N_f}{N_f + N_i + N_c}; \quad X_i = 0.3 \times \frac{N_i}{N_f + N_i + N_c}; \quad X_c = 0.2 \times \frac{N_c}{N_f + N_i + N_c} \qquad (2)$$

Moreover, in order to further improve the accuracy of the original weights, the MVICS approach takes into account the background of users. The users are classified into three groups, including software engineering researchers, software engineers and software engineering amateurs. Let the superscript of *X* and *N* represent the group of the users. The superscripts *R, E, A* indicate which user group is related, namely software engineering researchers, software engineers and software engineering amateurs. Taking into account the impact of the user's background, the original weight assignment rules are refined as follows:

$$X_f = 0.5 \times \frac{N_f^R \Big/ \frac{N^R}{N^R + N^E + N^A} + N_f^E \Big/ \frac{N^E}{N^R + N^E + N^A} + N_f^A \Big/ \frac{N^A}{N^R + N^E + N^A}}{Nf + Ni + Nc}$$

$$X_i = 0.3 \times \frac{N_i^R \Big/ \frac{N^R}{N^R + N^E + N^A} + N_i^E \Big/ \frac{N^E}{N^R + N^E + N^A} + N_i^A \Big/ \frac{N^A}{N^R + N^E + N^A}}{Nf + Ni + Nc} \qquad (3)$$

$$X_c = 0.2 \times \frac{N_c^R \Big/ \frac{N^R}{N^R + N^E + N^A} + N_c^E \Big/ \frac{N^E}{N^R + N^E + N^A} + N_c^A \Big/ \frac{N^A}{N^R + N^E + N^A}}{Nf + Ni + Nc}$$

Some users may wish only to be influenced by the views of the users in their own user group. Hence, the following rules are proposed to assign original class weight for this purpose, i.e., in support of user group oriented component search, which can further improve the precision of search results.

$$X_f^R = 0.5 \times \frac{N_f^R}{N_f};\ X_i^R = 0.3 \times \frac{N_i^R}{N_i};\ X_c^R = 0.2 \times \frac{N_c^R}{N_c}$$

$$X_f^E = 0.5 \times \frac{N_f^E}{N_f};\ X_i^E = 0.3 \times \frac{N_i^E}{N_i};\ X_c^E = 0.2 \times \frac{N_c^E}{N_c} \qquad (4)$$

$$X_f^A = 0.5 \times \frac{N_f^A}{N_f};\ X_i^A = 0.3 \times \frac{N_i^A}{N_i};\ X_c^A = 0.2 \times \frac{N_c^A}{N_c}$$

6.2.2 Weight of Search Path

The weight of a search path (*Wp*) in the MVICS (original component search path) is the sum of the weights of the classes included in it. The weight of a domain ontology class is given on the basis of the MVICS class weights. As mentioned in the chapter 5, the *AssL* and *AggL* were developed to link the domain ontology to the MVICS. In the domain ontology, for the class linked with a MVICS class via *AssL*, its weight is the same as the sub-class of the MVICS class it linked to. For a domain class linked with MVICS through *AggL*, its weight is the sum of weights of classes contained in the *Aggregation*. The *Wp* of the *AggL* is the sum of the weight of the classes and *aggregations* included in it.

6.3 Precision Calculation Method

To correspond with the MVICS model in which component specifications are classified into three facts, the keywords of a user query are also divided into three groups: Function Keywords (*FK*, the domain keywords belong to *FK*), Intrinsic Keywords (*IK*), and Context Keywords (*CK*).

The MVICS-based component search will then match the three groups of keywords one by one in the OWL files of MVICS and the domain ontology. Meanwhile, it will record the search path of each keyword and calculate the weight of the path. After retrieval, a set of records is obtained for each keyword, which includes the result component name, the search path and its weight. The match precision of a result component (*Pc*) is calculated with the following unified formula:

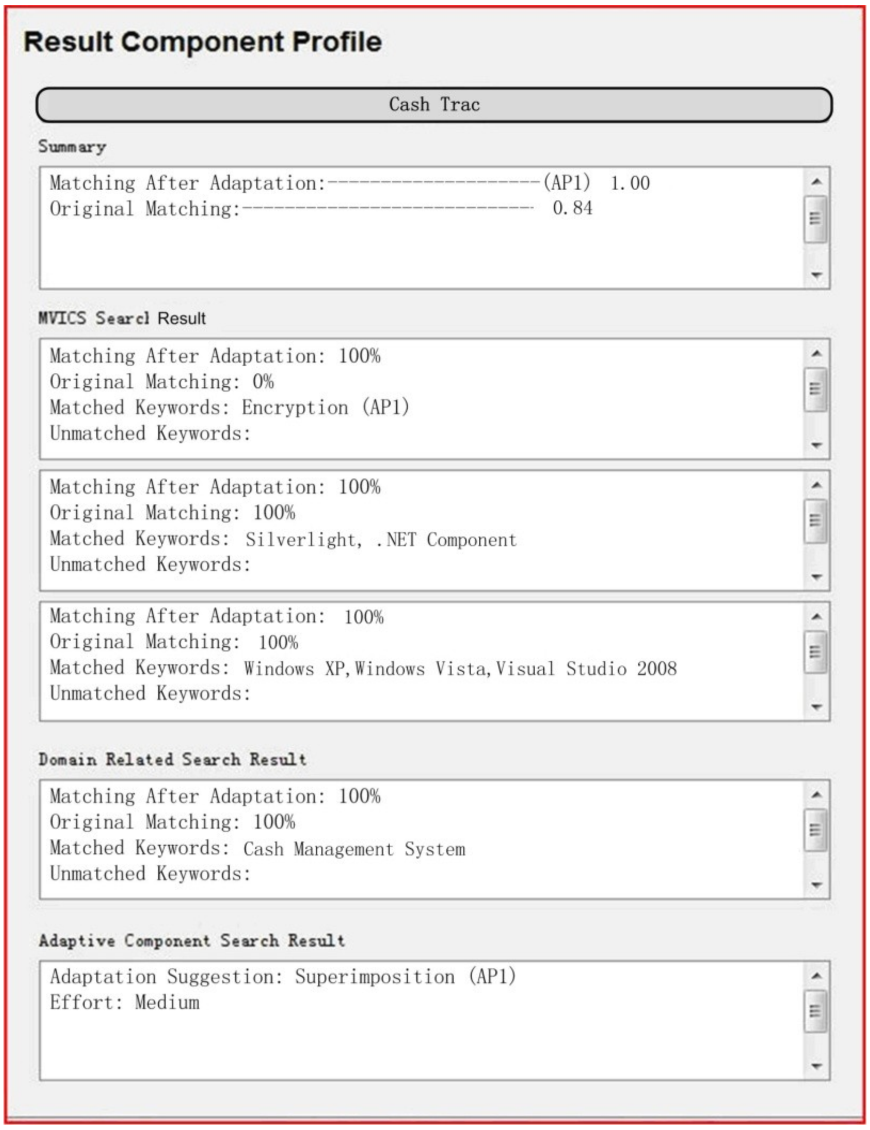
$$Pc = \frac{\sum_{r=1}^{a} WpFK_r}{\sum_{t=1}^{i} WpFK_t} \times X_f + \frac{\sum_{r=1}^{b} WpIK_r}{\sum_{t=1}^{j} WpIK_t} \times X_i + \frac{\sum_{r=1}^{d} WpCK_r}{\sum_{t=1}^{n} WpCK_t} \times X_c \qquad (5)$$

The numerators in the formula represent the path weight of the result components that partially match with the keywords in each facet, and the denominator represents the path weight of those perfectly matched. X is the original weight, $X = 0.5$ for a class in the function model, $X = 0.3$ for one in the intrinsic model, $X = 0.2$ for class in the context model. The yield value of the X for each sub model is given based on our experience, and it will be updated dynamically by the dynamic original weight assignment.

6.4 Result Component Profile

In contrast to most existing approaches, which present only the name and precision of the result component, a holistic Result Component Profile is designed (as shown in Figure 6.5) of the result component to help the user make the best decision in component selection. The profile consists of: i) the result component name is shown in the top middle; ii) search summary next, including the precision with component adaptation, and the precision without adaptation; iii) the match results in sub models: function model, intrinsic model, and context model; iv) the match results in the domain ontology model; v) the associated adaptation method or asset and its incurred effort.

The new structural result component profile shows the result from the whole to the part. First of all, the summary helps the user to realize how close of the match between the result component and the user query right away. It includes the result component precision with adaptation and without adaptation by which the best several components are selected for the further comparison. The next is the match results in sub facets of the user query that are classified according to the sub-models of the MVICS and the domain ontology. The search details in each facet help the user further pick the most suitable component in the previous set of similar components by considering the requirements. Finally, the details of the adaptive search result will be considered as additional factors to help the user to make the decision.


Result Component Profile

Cash Trac

Summary

```
Matching After Adaptation:--------------------(AP1)  1.00
Original Matching:---------------------------   0.84
```

MVICS Search Result

```
Matching After Adaptation: 100%
Original Matching: 0%
Matched Keywords: Encryption (AP1)
Unmatched Keywords:
```

```
Matching After Adaptation: 100%
Original Matching: 100%
Matched Keywords: Silverlight, .NET Component
Unmatched Keywords:
```

```
Matching After Adaptation: 100%
Original Matching: 100%
Matched Keywords: Windows XP,Windows Vista,Visual Studio 2008
Unmatched Keywords:
```

Domain Related Search Result

```
Matching After Adaptation: 100%
Original Matching: 100%
Matched Keywords: Cash Management System
Unmatched Keywords:
```

Adaptive Component Search Result

```
Adaptation Suggestion: Superimposition (AP1)
Effort: Medium
```

Fig. 6.5. The Result Component Profile

7. Realisation of the MVICS-Based Component Specification and Retrieval

To exert the power of the MVICS model in expressing semantics and process automation in component specification and retrieval, a MVICS-based component search tool and related component repository have been developed. The tool implements the complete process of component search, starting from filling the initial query and ending up with receiving the result component profile; the whole process is accomplished automatically. The tool consists of a set of modules, which carry out component search, including Dynamic Class Weight Assignment, Search Precision Calculator, Search Time Recorder and QAs Suggestion Processer. A component repository is built to store the components and their relevant adaptation assets/methods under the structure of the MVICS model and linked domain ontology model. Corresponding to the MVICS approach framework, the system architecture contains four functional parts: Users Query Refinement, Ontologies and Component Repository, Component Search and Result Display. The Ontologies and Component Repository is identified as the core of the component search process and it controls or supports the key subsystems in other parts of the tool. As the foundation of the search tool, the MVICS OWL files and the corresponding component specification in the

Ontologies have been introduced in previous sections. Other functional parts of the search tool will be described in the following subsections in the order of the workflow of component search.

7.1 Realisation of MVICS-Based Component Repository

Different from the current ontology-based approaches, the MVICS model is more versatile in terms of refining the user query, supporting component specification and managing the repository. Being registered into a repository, a component will be specified in a MVICS format form. The specification of this component will be linked, with the help of the form, to the relevant classes of each sub-model in the MVICS. Such a linkage reveals the semantic information of the originally syntax-based component specification, through either the relationship between classes within one facet, or the interrelationship between different facets. Furthermore, the domain semantics are represented with the linkage of MVICS to domain models.

All the contents of the MVICS model, including classes (MVICS classes and domain ontology classes), relationships, interrelationships, *AssL*s, *AggL*s, *Aggregation*, and relevant adaptive component information, are saved into four types of OWL files. When a user searches for components, the user's keywords will be searched in the OWL file to locate the matched classes in the MVICS. All the components relevant to the matched classes are identified as result components for this particular search.

Moreover, the components in the repository are managed with the support of the MVICS as well. When a component is changed, added to, or deleted from the repository, the repository manager can complete the operation by modify the OWL file of the component specification related classes. The impact of this operation to other classes will be realised through the automatic process by the reasoner.

7.1.1 OWL Files

During the component search process, the MVICS model plays a primary role, which specifies components from a range of perspectives. The contents of the MVICS model and domain ontology, including classes, individuals, relationships and constraints are edited by the ontology-editing tool Protégé [84] and are saved in OWL files. According to the facets of the MVICS and the domain ontology model, the OWL files can be categorised into four types, including the Original MVICS OWL file, Linkage OWL file, Adaptive OWL file and Domain Ontology OWL file.

The Original MVICS OWL file is generated by editing the contents of the function, intrinsic, context and meta-relationship models of the MVICS model. The *Association Class*, *Aggregation*, *AssL* and *AggL* are applied to integrate the domain ontology with the MVICS. Such contents are saved in the linkage ontology OWL file. The adaptive OWL file consists of the OWL format specification of the adaptive methods/assets, the relationship between the adaptive methods/assets and the MVICS class, and the adaptive suggestions (represented as the attributes). The connected domain ontology, its classes and superior-subordinate relationships are saved in the domain ontology OWL file.

7.1.2 Component Repository

Under the management of the MVICS and the domain ontology, 600 components (actually component specifications) are stored in a Microsoft Access database. The specifications of the components were selected from component sale websites, open source websites and the MVICS project website. The component names are registered with the related classes in the ontologies (MVICS and domain). The repository is maintained via the MVICS project website (http://ceres.napier.ac.uk/staff/chengpu/index.asp).

7.2 Financial Domain Related Ontology

To implement the MVICS-based approach to a specific domain, a financial domain related software system ontology was built as an example, by migrating existing financial operation ontologies. Following the proposed domain ontology migration method, financial operation ontology was retrieved from the protégé ontology library [85] with help of Google filetype search [35]. The required financial operation ontologies were then developed by migrating the financial operations from the selected ontologies. Each class in the new ontology represents one software system or module that carries out a financial operation. Superior-subordinate relationships have been used to describe the affiliations of the functions of these systems or modules. The top level classes include *Asset Management Systems*, *Payments & Transfers Systems* and *Risk Management Systems*. Their subordinates and the subsequent sub-subordinates constitute their sub classes and sub-sub-classes, till the most specific function units at the bottom level. Finally, the relevant *AssL*s, *Aggregation*s and *AggL*s are developed by the domain expert. The full details of the financial domain ontology are shown in Appendix B.

The method to define the domain ontology in OWL-DL is the same as the MVICS ontology, except the classes locate at the same level are not disjoint in domain ontology.

$C_i^1 \sqsubseteq \top$

$C_i^n \sqsubseteq C_k^{n-1}$, which defines C_i^n is a subclass of C_k^{n-1}, for example,

$C_{FI}^2 \sqsubseteq C_{PE}^1$, which states class *Fund Investment,* is a subclass of *Private Equity*.

Among these classes, *hasA* relationship is used to describe super- and sub-class links between classes in the adjacent levels. The relationship is defined as follows:

Relationship definitions:

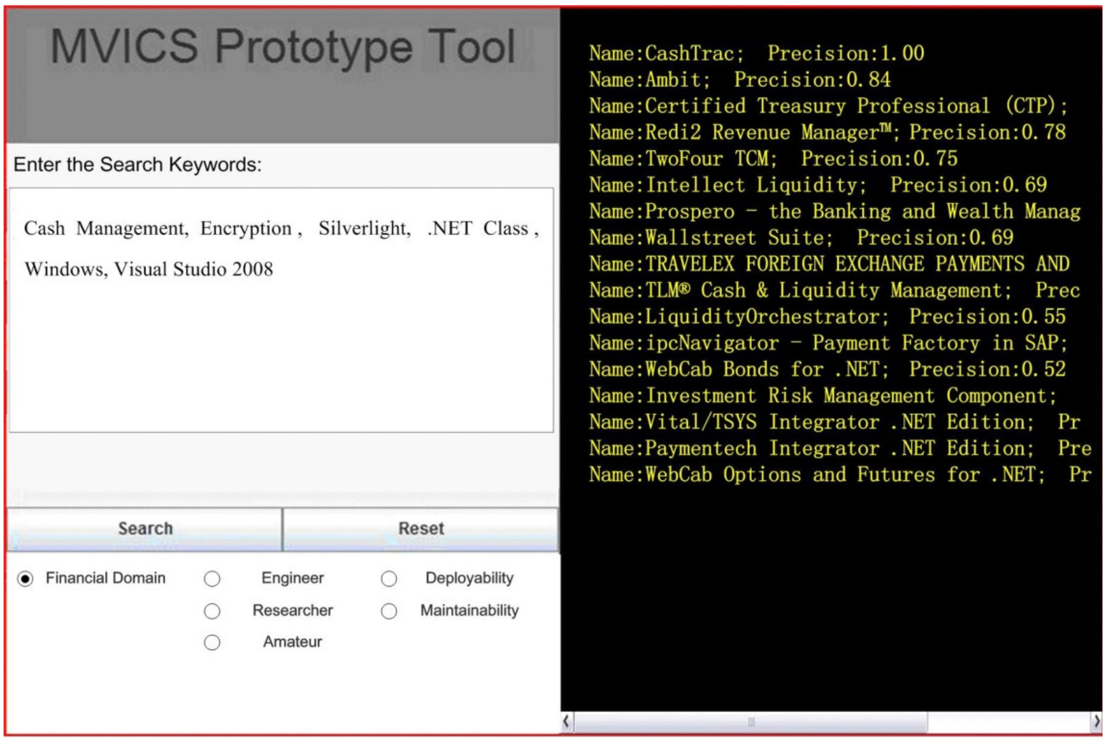
$C_i^n \equiv \forall hasA \,.\, C_i^{n+1}$, for example,

$C_{PE}^1 \equiv \forall hasA \,.\, C_{FI}^2$, which defines the relationship *hasA* links the class *Private Equity* to the class *Fund Investment*.

To transfer the component semantics between the MVICS and the financial domain ontology, *AssL*s and *AggL*s were established to link the financial domain ontology with the MVICS. All the *Association classes*, *Aggregations* and their related *AssL*s and *AggL*s are stored as the subclasses of the *financial domain* class in the function model of MVICS.

7.3 User Interface

Prior to the functional parts, a user-friendly interface is developed for users to fill in the search keywords and to operate the search options. A sample user interface of the tool is shown in Figure 7.1.



Enter the Search Keywords:

Cash Management, Encryption , Silverlight, .NET Class , Windows, Visual Studio 2008

[✓ Financial Domain] Engineer Researcher Amateur Deployability Maintainability

```
Name:CashTrac;  Precision:1.00
Name:Ambit;  Precision:0.84
Name:Certified Treasury Professional (CTP);
Name:Redi2 Revenue Manager™; Precision:0.78
Name:TwoFour TCM;  Precision:0.75
Name:Intellect Liquidity;  Precision:0.69
Name:Prospero - the Banking and Wealth Manag
Name:Wallstreet Suite;  Precision:0.69
Name:TRAVELEX FOREIGN EXCHANGE PAYMENTS AND
Name:TLM® Cash & Liquidity Management;  Prec
Name:LiquidityOrchestrator;  Precision:0.55
Name:ipcNavigator - Payment Factory in SAP;
Name:WebCab Bonds for .NET;  Precision:0.52
Name:Investment Risk Management Component;
Name:Vital/TSYS Integrator .NET Edition;  Pr
Name:Paymentech Integrator .NET Edition;  Pre
Name:WebCab Options and Futures for .NET;  Pr
```

Fig. 7.1. The main interface of the prototype tool

On the top left, there is a text area for the user to fill in search keywords. On the bottom left, these are three columns of option buttons: the first column lists the available domain ontologies, which has Financial Domain as an example. The second column spreads out user-oriented options which include Software Engineering Researcher, Software Engineer and Software Engineering Amateur. The user-oriented options can further improve the search precision by using different parameters, which are obtained by analyzing the user queries and collecting the users feedback. The last column lists the three available quality attributes. The result components are presented based on selected QA from high to low, when they have a similar search precision. The use of the option buttons will be introduced in the case study. On the right hand side of the UI is a black panel showing the summary of search results.

7.4 User Query Refinement

User query refinement has been developed in depth [30][170]; it focuses on determining the nouns and verbs from natural language, and processing the keywords based on the related domain model. According to the existing user query refinement methods, the MVICS component search tool combines three sub systems including Synonym Operator, Symbol Operator and Keywords Discriminator (as shown in fugire 7.2). The Synonym Operator formats the synonyms into a uniform keyword based on the classes of the MVICS model. The Symbol Operator identifies the function of number restriction constructors (≤, ≥, <, >), and logic symbols (¬). With the support from the MVICS model, the Keywords Discriminator recognizes keywords which are suitable for database search, such as the keywords of exact component name, component version and component vendor which are implemented through database keyword search directly. This is because a database search on these terms will obtain the same search result but with a faster speed than the MVICS based ontological search.

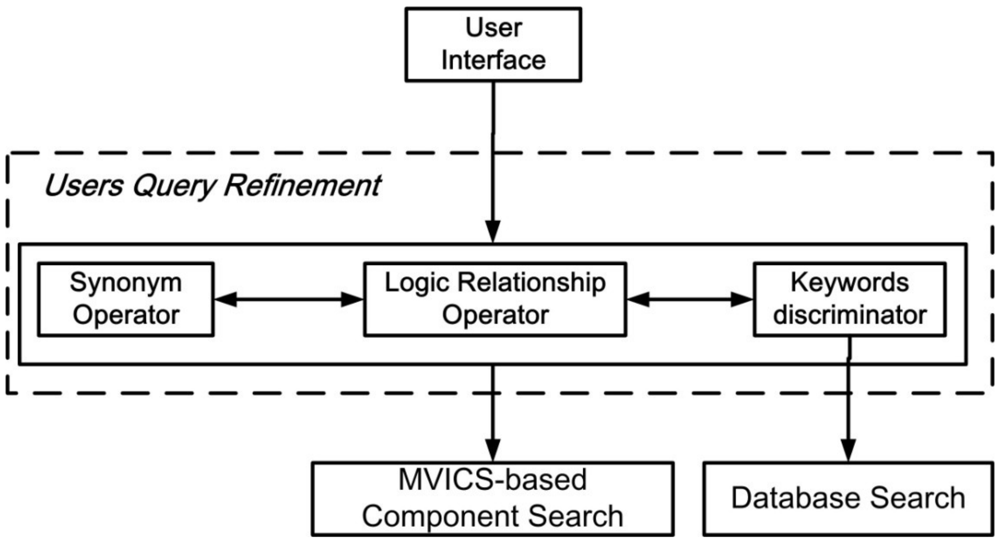
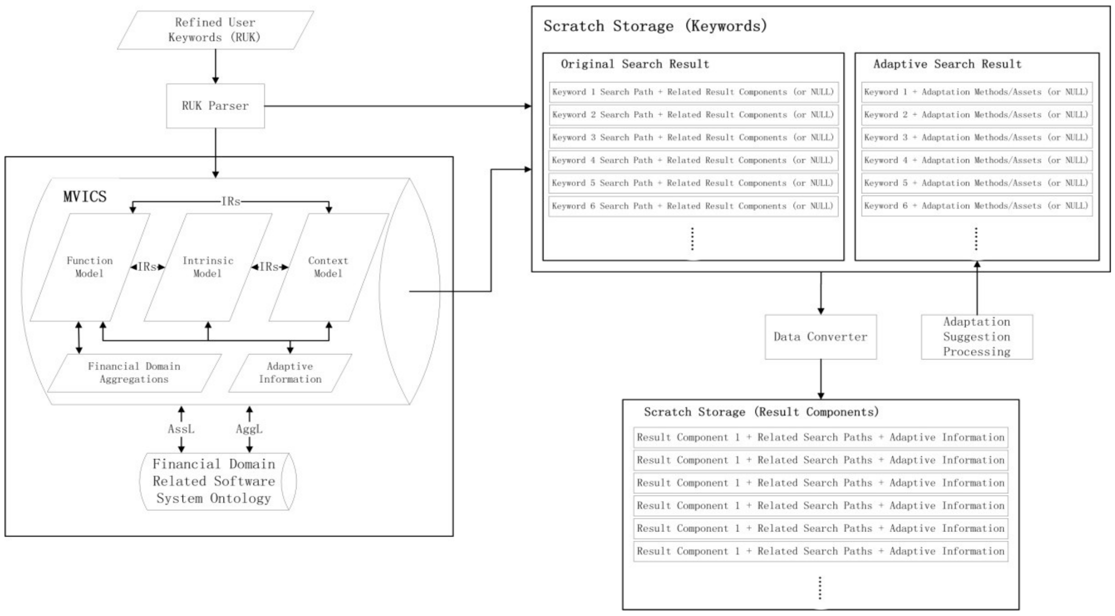


Fig. 7.2. The structure of the user query refinement

7.5 MVICS Component Search

The MVICS component search is based on the MVICS model and the linkage with the domain ontology model. It focuses on retrieving the relevant components from the repository according to the refined user keywords by identifying the search paths. The

MVICS component search identifies the matched classes in the MVICS, *AssL*, *AggL* and adaptive OWL files with the refined user keywords and it retrieves the result components via the matched classes. The workflow of the MVICS component search is shown in Figure 7.3, the details will be introduced in the following sub-sections.



Refined User Keywords (RUK)
RUK Parser
MVICS
IRs
Function Model
Intrinsic Model
Context Model
Financial Domain Aggregations
Adaptive Information
AssL
AggL
Financial Domain Related Software System Ontology
Scratch Storage (Keywords)
Original Search Result
Keyword 1 Search Path + Related Result Components (or NULL)
Keyword 2 Search Path + Related Result Components (or NULL)
Keyword 3 Search Path + Related Result Components (or NULL)
Keyword 4 Search Path + Related Result Components (or NULL)
Keyword 5 Search Path + Related Result Components (or NULL)
Keyword 6 Search Path + Related Result Components (or NULL)
Adaptive Search Result
Keyword 1 + Adaptation Methods/Assets (or NULL)
Keyword 2 + Adaptation Methods/Assets (or NULL)
Keyword 3 + Adaptation Methods/Assets (or NULL)
Keyword 4 + Adaptation Methods/Assets (or NULL)
Keyword 5 + Adaptation Methods/Assets (or NULL)
Keyword 6 + Adaptation Methods/Assets (or NULL)
Data Converter
Adaptation Suggestion Processing
Scratch Storage (Result Components)
Result Component 1 + Related Search Paths + Adaptive Information

Fig.7.3. The workflow of the MVICS component search

7.5.1 Refined User Keywords Parser

Prior to the component search, the refined user keywords will be further processed by the Refined User Keywords Parser (RUK Parser). The parser first parses the keywords based on the sub models of MVICS. It classifies the keywords into three groups, including function keywords (domain related keywords belong to function keywords), intrinsic keywords and context keywords. In the second step, the parser generates several scratch storages for the component search according to the numbers of the keywords. The scratch storages deposit the temporary search data generated during the searching process.

7.5.2 MVICS Component Search Path

The MVICS component search tool searches the classified keywords in the function model, intrinsic model and context model of the model and the domain ontology model respectively. The components related to the matched classes in each model are identified as the result components. The search also assigns a precision to illustrate the relevance rating for each result component, which is calculated on the basis of search paths obtained during the search. Four types of search paths are identified according to the location of the matched classes in the ontologies as mentioned in the previous chapter, including the original MVICS search path, the *Association Link* (*AssL*) search path, the *Aggregation Link* (*AggL*) search path and the adaptive search path.

7.5.3 Adaptive Component Search Scratch Storage and Adaptive Suggestion Processing

The adaptive component search is actually part of the MVICS component search. The search results are the components that match with the user query after the relevant adaptation. For the MVICS component search tool, the same number of scratch storages for the adaptive search paths will be created after the adaptive component search. The searched keywords with their matched adaptation methods/assets are saved in the corresponding storages as show in Figure 7.3. In addition, an adaptation suggestion processing will give every matched method/asset an effort suggestion. The effort suggestions are classified into three levels, which indicate as Strong, Medium and Weak. For each available adaptation method/asset, the effort suggestion information is defined as attributes of the relevant classes, which are stored in the MVICS model. An adaptation suggestion process invokes these attributes and saves them in the adaptive search result storage for displaying to the user.

7.5.4 Result Component Oriented Data Conversion

After the component search, the search results will be processed by the Result Component Oriented Data Converter. The data converter will implement three tasks. The first task is to map the four types of component search paths (the original MVICS search path, the *AssL* search path, the *AggL* search path and the adaptive search path) into Function Keyword search path (pFK), Intrinsic Keyword search path (pIK) and Context Keyword search path (pCK), based on whether the matched classes in the component search paths belong to the function model, the intrinsic model or the context model of the MVICS. The second task is data conversion. The original search results and the adaptive search results in the scratch storage are saved according to users keywords. In order to calculate the precision of each result component, the keyword oriented search results (original and adaptive) should be converted into result component oriented by the data converter. The converted data includes the result component names and related search paths (pFK, pIK and pCK), are stored in a new scratch storage. The third task is that the data converter records the information of matched keywords, unmatched keywords and adaptive information for each result component.

7.5.5 Precision Calculation

Based on the converted data, the match precision of a result component (*Pc*) is calculated with the unified formula (Formula (5)) as defined in Chapter 6.

7.5.6 Dynamic Weight Assignment

The MVICS component search tool is based on the tree structure of the MVICS model, and calculates the precision of the result component by supplying the values of the search path weights and the original weights (*X*) into the Formula (5). As mentioned in section 6.2.The *X* of the classes in each model are updated dynamically by the weight assignment Rules (2) after every 100 groups of user keywords are obtained. Moreover, in order to further improve the accuracy of the original weights, the dynamic *X* are further updated by taking into account the impact of the user backgrounds with the weight assignment Rule (3) and Rule (4).

7.5.7 QAs-based Result Component Ordering

After the calculation of result components precision, the result components are listed in the descending order of precision. If the user selects the option button of "QAs suggestion", the result components with the same search precision will be further ranked in the descending order of the values of the selected QAs. A QAs-based result component order processor will put the result components that have more QA related factor (as mentioned in section 5.1) in front.

7.5.8 Result Component Profile

In the MVICS-based component search approach, a Result Component Profile is proposed to present a comprehensive view of the search result to the user. This task is fulfilled by the result component profile creator in the MVICS-based component search tool. The creator collates and arranges the search result and the result component precision, in accordance with the result component profile format. It carries out three functions: firstly, the result component precision is formatted with the accuracy of 2 decimal digits; secondly, for each result component, its relevant search paths are arranged by the MVICS-based component search result, domain related component search and adaptive component search; and thirdly, having gone through the MVICS component search and the adaptive component search, the result component profile creator will calculate the percentages of the matched keywords amongst all searched keywords in every facet.

7.5.9 Search Time Recorder

To measure the speed, the tool has a Search Time Recorder device, which is used to record the time consumed during component search. The device has two recording points: the first point is used to measure the model search time, and it is set to the completion of the MVICS-based search (includes adaptive component search). The second point is used to measure the post-search processing time, and it is set to the formation of the result component profile. The post-search processing time includes the search precision calculation, the user query record, the QAs-based result component order and the result component profile generation.

7.6 Result Display

In the MVICS component search tool, three panels are developed to show the results. In the user-friendly UI (Figure 7.2), the black panel at the right hand side shows the summary of search results, which comprises the result component names and their precisions. The result components are arranged in the descending order of their precision. If the QAs suggestion options (deployability and maintainability) are selected, the result components that match the most QAs related factors will be listed at the front of the list if the result components happen to have the same value of precision. If the user clicks the result component name, the result component profile will appear. In contrast to most existing component search tools, which only present to the user the name and precision of the result component, the MVICS tool provides a holistic profile of the result component to help the user make the best decision in component selection. The profile shows the matching result in each sub-model of MVICS, the result in domain ontology and the corresponding adaptation information. By clicking the component name in the profile, the complete specification of the component will be presented.

7.7 The MVICS Project Website

In order to facilitate the users to test the approach through the prototype tool, a project website (http://ceres.napier.ac.uk/staff/chengpu/index.asp) was built as an intermediary platform. The project website helps users find the relevant information of the project (Appendix C.1) test the prototype tool online (Appendix C.2), and leave comments and feedback. The website contains the following main information for the users:

- The introduction of project. It details the project background, main contributions, critical methods and the architecture of accessible ontologies (MVICS and domain ontologies).

- The instruction of the MVICS component prototype search tool. It specifies the applicable domain keywords, layout of the UI, the component search steps, the functions of the tool and the specification of the result component profile.

- The specification of the sample components. The specifications of six hundred sample components are represented and managed by the facets classification methods on this website.

- The online prototype tool. This tool is transformed into a web application (Java applet) and published on the website.

- Database keyword search. On the website, a traditional component search tool was plugged in as well, which was supported by the basic keyword search and the faceted classification approaches. This search tool is placed here for the comparison test with MVICS component search tool. The comparative criteria include search recall, precision and speed, etc.

- User registering and member interface. Prior to the test, users are requested to register as a member first. Part of the registration information is the user background, which will be recorded as the data for analysis in the user-oriented search. A user-friendly member interface panel will pop up after log in.

- Own component register. Users can register their own components by filling out a MVICS format registration form. With the consent from the administrator, the component specifications are link to the related class with the ontologies. The semantics will be generated and extended with the help of defined relationships. The registered components are exhibited on the component list and can be used for the approach evaluation.

- Feedback questionnaire. A questionnaire of the test result will be prompted to the user for completion. It refers to search recall, search precision, result display, and other related questions.

Moreover, the project website helps the administrator to manage the component repository and summarise the feedback and comments.

- Component specification management. In the backend management of the website, the registered components (including the component specification which are added by the repository administer and the users) can be managed by the administrator, e.g. add, amend, approve, reject and delete components.

- Data statistics of the questionnaires. The outcome of questionnaires will be recorded and analysed at the backstage by generating an XML-based result summary, which will be updated after every ten questionnaires.

8. Case Studies

8.1 Introduction

Three case studies have been undertaken to illustrate and validate the MVICS-based approach, in terms of its capabilities of 1) supporting the accurate component specification and retrieval, 2) seamless linkage with the domain ontology, 3) adaptive component suggestion, 4) comprehensive result component profile and 5) Quality Attributes suggestion. Each case study focuses on one or more of the above aspects. Case studies are selected for the analyses, only when the relevant participants meet all of the following 4 criteria: 1) They are specialised in one type of software, and focused on component reuse for software development; 2) They possess large quantity of components, with similar functions, and with independent intellectual property rights; 3) They are interested in new semantic-based approach and willing to feedback their comparison of the MVICS-based approach with their existing techniques; 4) Their participation may help test one or more of the 5 capabilities of the MVICS-based approaches. As the start point, the first case study is an essential experiment which aims at checking the correctness and the capability of the MVICS-based approach. The first four capabilities of the method can be validated by this case study. As further extension of the evaluation, the other two case studies place stress on exhibiting the advantages of the approach in real-life applications. The large-scale two-stage Metadata Management Platform System (MMPS) project case study focus on testing the capability 1), 3) and 4). And the Racing Games case study focus on testing the capability 1), 2), 4) and 5). The user feedback has been collected as the results of MVICS validation, and as the basis for further improvement.

8.2 Essential Validation and Evaluation of the MVICS Prototype Tool

8.2.1 Background

The empirical method is used for the MVICS approach validation; user feedback is collected after testing the use of the prototype tool. To exemplify the use of the MVICS component search prototype tool, twelve search scenarios with corresponding search results are offered on the project website. Users can opt to test some of these given scenarios, or construct their own scenarios for the testing. In this case study, a financial domain scenario of developing an encrypted Cash Management Systems with user-friendly interface is taken as an example, to illustrate the function and process of the MVICS repository and its linkage with a domain related software system ontology. As mentioned in section 7.2, a financial domain related software system ontology has been built, by migrating existing financial operation ontologies. The relevant *AssLs*, *Aggregations* and *AggLs* have been developed with the support of domain experts and saved under the sub-class *Financial* of class *Component Domain* in the Function model.

8.2.2 The Process

The functional requirements of the required encrypted Cash Management Systems, as its name implies, basically include encryption and cash management. Generally speaking, the software system will be composed of a number of components and it will manage liquidity, account balances, payments and other functions related to cash management under an encryption environment. The required components are .NET type, more specifically Silverlight or .NET Class. Correspondingly, they will be deployed in a Windows environment with Visual Studio 2008 or above.

After the user query refinement, the scenario is specified as the following search requirements:

Function:	*Cash Management, Encryption*
Component Type:	*Silverlight, .NET Class*
Component Platform:	*Windows*
Component Container:	*Visual Studio 2008*

The user may fill in the relevant keywords into the text area of the search UI and click the financial domain option button to extend the search into financial specific components. Next the deployability suggestion option button is selected to ensure the result components which can be deployed in more platforms to be listed in the front of the result list. Lastly, the user-oriented option buttons are left blank for original precision calculation. Once the search starts, the search tool first identifies which sub-model the keywords belong to in MVICS. Meanwhile, the information of the keywords and their related models are recorded by the requirement recorder for future refinement. The search engine searches the keywords one by one in the original MVICS OWL file, the financial domain OWL file and the adaptive OWL file respectively.

During the search process, the relationships between the sub-models provide more semantics: for instance, the matched class *silverlight* in the intrinsic model has a Matching Propagation Relationship with class *multimedia, graphics* and *animations* in function model. With the semantics of the interrelationship, the *silverlight* type components should have the function *multimedia, graphics* and *animations.* Therefore, these functions are taken into account in the search as supplement. At the same time, in the context model, the matched class *visual studio 2008* specifies that the OS requirements should be Windows XP or above by the Conditional Matching Propagation Relationship. With the support of the Supersedure Relationship, the search tool will identify the components as the result if the edition of their container is beyond the 2008, e.g. *visual studio 2010.*

The keywords "*Encryption, Silverlight, .NET Class, Windows* and *Visual Studio 2008*" are matched with the classes of MVICS, and their relevant components are identified as the result components first. Afterwards, the search tool continues to search the unmatched keywords "*Cash Management*" in the financial domain OWL files in the same way.

After searching the MVICS and the financial domain ontology OWL files, the tool searches available adaptive methods/assets in the adaptive OWL files. All the components relevant to the matched classes are identified as the result components of the search, and the precision for each result component is calculated on the basis of the retrieved search paths by the Precision Calculator. The findings offer the user more options to develop the cash management software systems.

The names of the result components and their precision are displayed in the descending order of precision; the more portable components are listed in front of other components with the same precision on the right panel of the UI. When a result component is highlighted, its result component profile will pop up, which provides comprehensive search information for each result component, including the result component name, overall precision of the component search, the match results in function, intrinsic and context model, the match result in the financial domain ontology and the available adaptation methods/assets with their efforts to apply.

To compare with the MVICS component search tool, an SQL database search tool and a domain ontology-based component search tool are presented on the project website for the comparison test. The SQL database search tool implements the traditional keyword search approach with the support of the facet classification and behavior matching approach. The domain ontology-based component search tool implements the existing domain ontology-based approach. It uses the same financial domain ontology for refining the user requirements and specifying the component without the support of MVICS. Software engineers, researchers and amateurs are able to use the applications with the same testing scenarios and to comment on the tool and the search result via a questionnaire.

8.2.3 Analysis

As mentioned in chapter 7, to facilitate users understand the MVICS component search prototype tool and to perform the comparison testing and leave comments, a project website is built. To date, 125 users have tested the tool in practice, among whom 43% from Europe, 33% from Asia, 17% from North America and 7% from the rest of the world. In the self-appraisal to scale their own software engineering experiences from level 1 to 5, 15% opt for scale 1, 7% for 2, 44% for 3, 23% for 4 and 11% for 5. The explanation of test participant's experiences level is shown in Table 8.1.

Users are classified into three groups:

- Software Researchers are those who pursue research careers in institute or software companies;

- Software Engineers are those who engage themselves in software development in the software companies;
- Amateurs cover the rest of the users.

Given such a classification, the different impacts of user's background and occupations are tested. The differentiation of academic and industrial, professional and amateur is to verify the impacts of different users' behaviour on the results of MVICS-based approach in application. This classification can be used as the basis of further improvement. It will also make it convenient to provide users of different background with Precision calculations fitful for their own application scenarios.

Table 8.1. Level of Experience of Testers

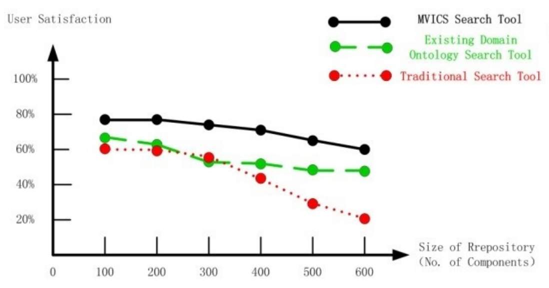
Level of Experience	**Explanation**	**Proportion**
1	user has more than 10 years software engineering experience	7%
2	user has more than 5 years	15%
3	user has more than 3 years	44%
4	user has more than 1 year	23%
5	user has no experience	11%

Table 8.2. Professional Background of Testers

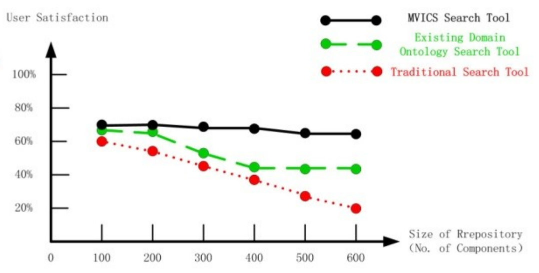
Occupation	Software Researcher	Software Engineer	Amateur
No. of respondents	59	39	27
Years of experience (Average)	7	5	3

The information of test participant's professional background is shown in Table 8.2. And the results of these component retrieval experiments are analyzed and given in Figure 8.1.

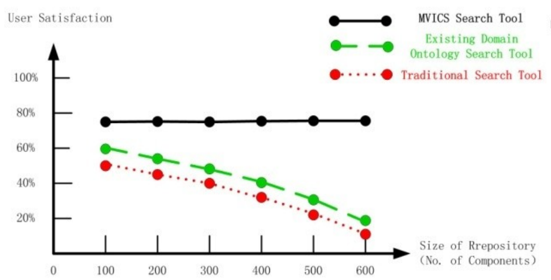
Precision and *recall* [153][190] are two critical dimensions to evaluate the effectiveness of the component search. The MVICS component search tool improves the search precision and recall, particularly for the large repository as shown in Figure 8.1 a. and 8.1 b. Such improvements come from its semantic foundation, i.e., the formalized MVICS ontology model and its linkage with the domain ontology, as well as the successful automation in the repository and search tool. MVICS represents component specification in a multi-faceted and hierarchical structure. In addition, the interrelationships in the MVICS model and the linkages with the domain offer more semantic meaning among the classes of the ontologies (MVICS and domain). In this case, more search paths are retrieved during the search process. This should lead to further improvement on recall and precision. Comparatively, with lower description capability and less relationship in their ontology, other existing domain ontology search tools offered lower satisfaction in *precision* and *recall* than the MVICS-based tool. Without the support of ontology for the component specification and retrieval, the traditional search tool offers even lower satisfaction when searching in a large repository.



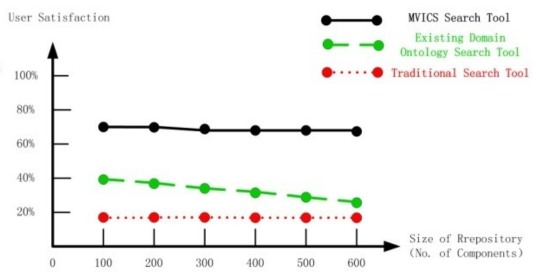
(a) Satisfaction of Precision



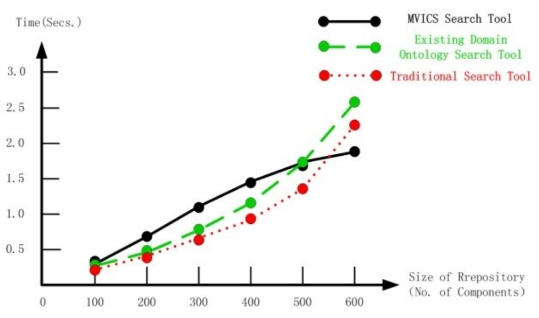
(b) Satisfaction of Recall



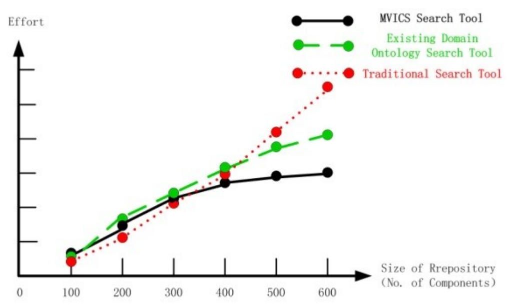
(c) Satisfaction of Result Display



(d) Satisfaction of Adaptation Suggestion



(e) Search Speed



(f) Maintenance Effort

Fig. 8.1. The level of satisfaction of the MVICS component search tool, existing ontological domain specific search tools and traditional search tools

The *result display* is to indicate the degree of user satisfaction with the display of the result components in terms of the completeness, clearness and usefulness. Via the proposed result component profile, the search results are shown more effectively in the MVICS component search tool (Figure 8.1c).

The criteria *adaptation suggestion* is used to estimate the degree of usefulness and user acceptance of the found adaptation suggestion. With the novel and unique feature of the proposed adaptation component search among all component search tools, the MVICS offers remarkably improved satisfaction in the aspect of adaptation suggestion, as shown in Figure 8.1d.

Further to the improvement of search accuracy, the MVICS component search tool also takes into account other related properties, such as search speed. As mentioned in the section 7.5, the tool has a Search Time Recorder device to record the time consumed of model search time and post-search processing time. Figure 8.1e shows the comparison of the search speeds of the different search tools in repositories of varying sizes. Ignoring the time consumed in the pre-preparation work, the existing domain ontology search tools and the traditional search tools are faster than the MVICS approach when the size of a repository is less than 500 components; however, the MVICS component search tool becomes faster when the repository is large. This is because the MVICS component search tool searches classes in the ontology along its multi-faceted and hierarchical structure. The reason for the speed loss in the MVICS component search tool is the additional semantic processing, i.e., semantic-based precision calculation, adaptive component search and data collection for a whole profile of result components.

Regarding the effort of maintenance, it is observed that the ontology-based search tools (MVICS and existing domain ontology-based approach) are easier to manage in medium-sized repositories, as shown in Figure 8.1f. Again this advantage comes from the fact that the MVICS component search tool uses an ontology formally defined in OWL-DL, which makes it possible for automatic validation through ontology reasoners. More effort in developing the *AssL*s, *Aggregation* and *AggL*s to integrate domain ontology model into MVICS will ensure the domain viability of the MVICS approach.

8.3 Large-scale Two-stage Metadata Management Software System Case Study

This case study places more emphasis on demonstrating that the MVICS-based approach is capable for industrial scale component-based development in real life. The use of MVICS enhances component specification and the highly automated capability that improves the efficiency of large-scale multi-functional software development. In particular, the approach utilizes its unique adaptive component suggestion to provide more appropriate choices for the developers.

8.3.1 Background

In this case study, we cooperated with Beijing Mysoft Technology Co., Ltd. [97] in the development of metadata management software system for China State Grid Beijing Branch. Beijing Mysoft Technology Co., Ltd. is a developer and service provider that specializes in Enterprises Management Systems and Database Management Systems. Its independently developed product "Smart Collaborative Management Platform" passed the system evaluation by China National Software Testing Centre (CSTC) [90]. Mysoft has successfully constructed a number of Enterprise Integrated Service Management Platforms and Enterprise Portal Management Platforms, which are applied to enterprise database management, enterprise integrated office management, human resources management, asset management, customer relationship management, procurement management, and portal management.

The project of the large-scale two-stage metadata management software system was done by the development group led by Mr. Jia Ding, which consisted of 10 members. The MVICS-based approach is then applied to resolve the problems encountered in the previous process. The problems are similar to those often occurring in CBD, including inappropriate component specification, inaccurate component retrieval and missing of adaptive components (a kind of low level of component search recall). The

MVICS approach is implemented with support of the Mr. Ding led four-person test group. Their feedbacks are collected as the results of MVICS validation.

8.3.2 Mysoft's Existing Component Specification and Retrieval

8.3.2.1 Introduction

Mysoft has about 500 components, including 58 metadata management components, in their in-house repository. The components are obtained from foregoing projects related to metadata management system development. The component developers produce the components, which can implement one or more functions of metadata management. Because the components come from different projects, the diversity of their characteristics is large, such as size, functions type, container and platform. In order to specify the components accurately and clearly, a popular business component specification method was selected [41]. The method divides the specification of a component into seven levels, including interface, behaviour, interaction, quality, terminology, task and marketing. The specification aspects of each level are shown in Figure 8.2.

After the specification, the components are saved in Mysoft's in-house repository with the specification documents (an example is given in Table 8.3) searchable through keywords. An SQL database search tool is proposed to facilitate the component retrieval. In order to further support narrowing the search space and still retrieve relevant components, three facets of the search parameters are classified for the developers. The details of function type facet, component type facet and platform facet are shown in Figure 8.3.

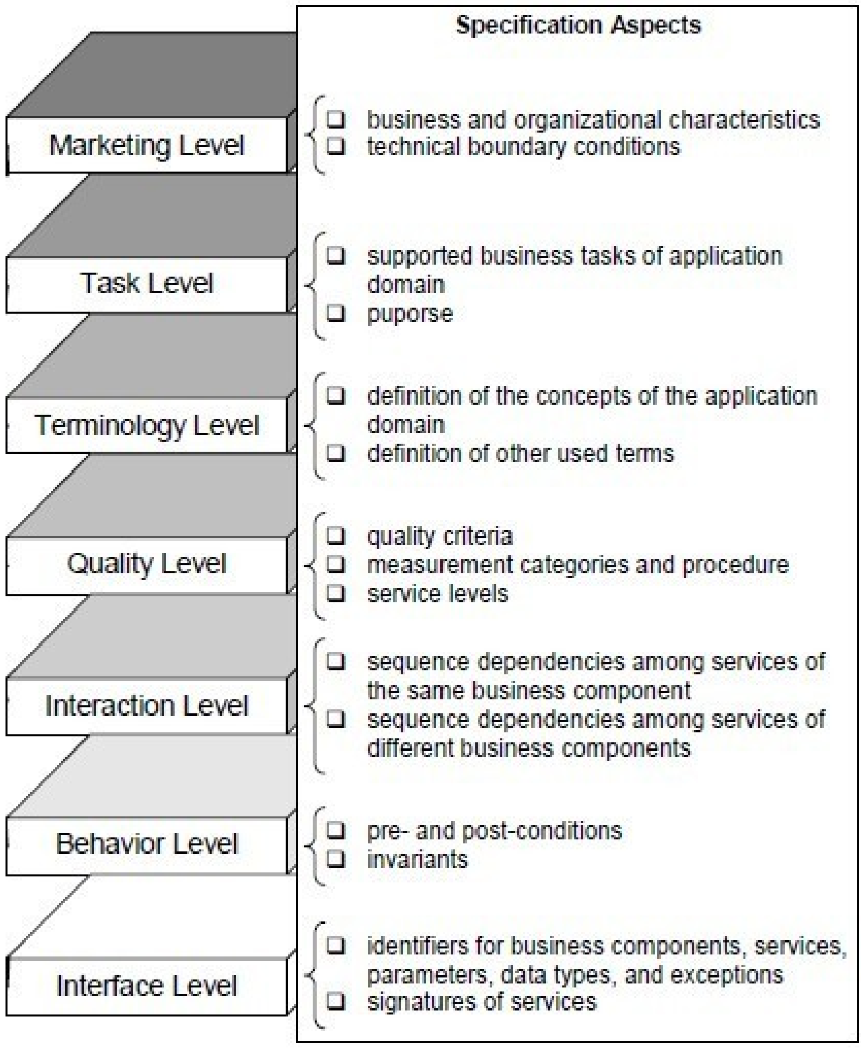
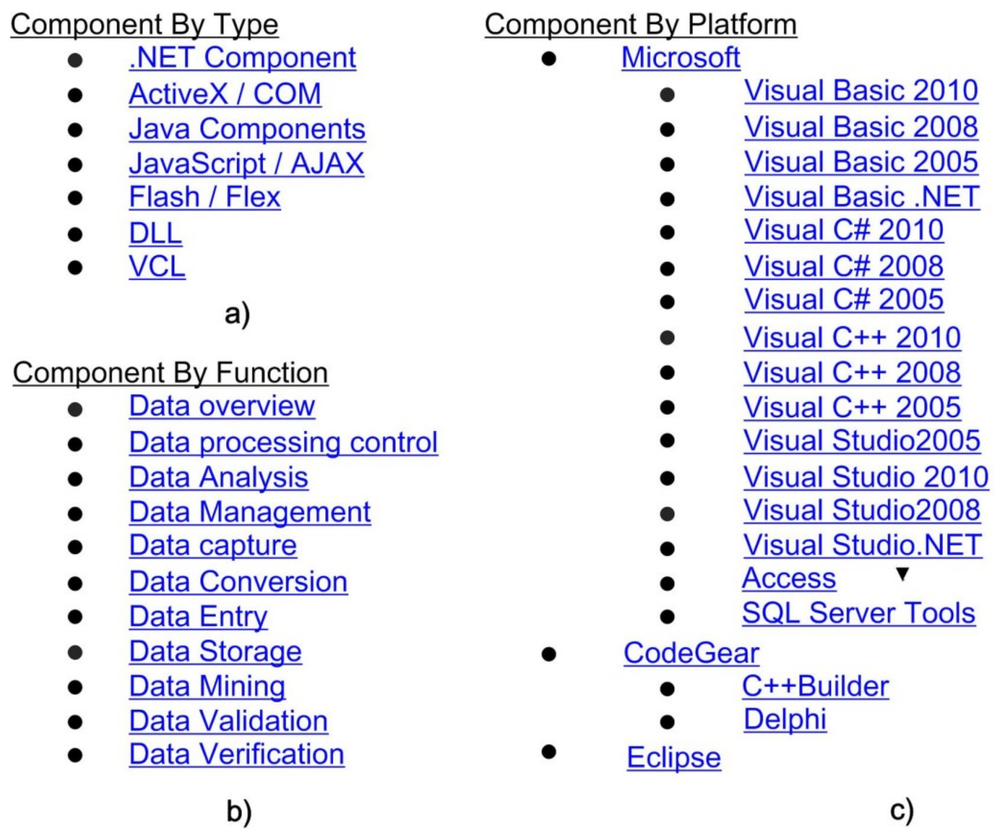

Specification Aspects
Marketing Level
business and organizational characteristics
technical boundary conditions
Task Level
supported business tasks of application domain
puporse
Terminology Level
definition of the concepts of the application domain
definition of other used terms
Quality Level
quality criteria
measurement categories and procedure
service levels
Interaction Level
sequence dependencies among services of the same business component
sequence dependencies among services of different business components
Behavior Level
pre- and post-conditions
invariants
Interface Level
identifiers for business components, services, parameters, data types, and exceptions
signatures of services

Fig. 8.2. Specification levels and specification aspects

Table 8.3 Example of component specification document

Mysoft Data Analysis ASP 025	
Function Summary:	Add data mining and multi-dimensional analysis to ASP.NET applications. Mysoft Data Analysis ASP 025 is a comprehensive data analysis, data mining, and visual reporting solution for ASP.NET 2. With Mysoft Data Analysis ASP 025, your users can break down raw data in any manner they require using easy-to-understand Windows commands. By incorporating the Mysoft Data Analysis ASP 025 in your application, you can deliver an almost endless array of reports. Mysoft Data Analysis ASP 025 is able to slice and dice information efficiently and provide customers with an intuitive end-user experience. Mysoft Data Analysis ASP 025 delivers numerous layout customization options with total end-user control over each individual on-screen report. Mysoft Data Analysis ASP 025 Subscription with Source code Licenses available. Data Source Support, Display Table Data into Fully Customizable Visual Reports, Control the Level of Detail, Arrange Values Hierarchically, Automatic and Manually Specified Totals, Sort Data and Display Top Rows, Filter Data, Hide unnecessary values in the axes and Filter the data against which calculations are based.
Component No.:	DA-03-025
Version:	1.0
Component Type:	ASP.NET WebForms, ASP.NET AJAX, .NET Class, AJAX
Built Using:	Visual C# .NET
Product Class:	User Interface Components
OS	Windows 7, Windows Vista, Windows XP, Windows 2000
Compatible Containers:	Microsoft Visual Studio 2010, Microsoft Visual Studio 2008, Microsoft Visual Studio 2005, Microsoft Visual Basic 2010, Microsoft Visual Basic 2008, Microsoft Visual Basic 2005, Microsoft Visual C++ 2010, Microsoft Visual C++ 2008, Microsoft Visual C++ 2005, Microsoft Visual C# 2010, Microsoft Visual C# 2008, Microsoft Visual C# 2005
Prerequisites:	Disk Space Required: 10MB Memory Required: 64MB
Behaviors:	TypeCodes Self. ListOfTypes->forall (b:Types) | (b. TypesCode > 00) TypeCodes::convertTypeCode(name : Typename): TypeCode pre : self.ListOfTypes->exists(b: Type | b. TypeName = Typename)
Throughput:	2.3s
Response time:	23ms
Time distribution	0.0425ms



Component By Type
- .NET Component
- ActiveX / COM
- Java Components
- JavaScript / AJAX
- Flash / Flex
- DLL
- VCL

a)

Component By Function
- Data overview
- Data processing control
- Data Analysis
- Data Management
- Data capture
- Data Conversion
- Data Entry
- Data Storage
- Data Mining
- Data Validation
- Data Verification

b)

Component By Platform
- Microsoft
  - Visual Basic 2010
  - Visual Basic 2008
  - Visual Basic 2005
  - Visual Basic .NET
  - Visual C# 2010
  - Visual C# 2008
  - Visual C# 2005
  - Visual C++ 2010
  - Visual C++ 2008
  - Visual C++ 2005
  - Visual Studio2005
  - Visual Studio 2010
  - Visual Studio2008
  - Visual Studio.NET
  - Access ▼
  - SQL Server Tools
- CodeGear
  - C++Builder
  - Delphi
- Eclipse

c)

Fig. 8.3. Facets of search parameters

8.3.2.2 Search Scenario

To test the Mysoft component specification and retrieval method, a scenario of developing a two-stage Metadata Management Software system (MMS) was adopted. The function requirements of the MMS are divided into two stages. In the first stage, the five basic metadata management functional modules are built, including metadata survey, metadata analysis, metadata management and metadata mining. The last three modules can be further divided into several sub-modules. In the second stage, an operation metadata browser and analysis module is developed by holding, improving or optimizing the basic modules in first stage and creating new function modules; two new second stage modules, namely metadata management and metadata mining are created; the modules of metadata management and metadata are kept. Figure 8.4 shows the details of the function requirements of the two stages. The modules with light grey are created in the first stage; the modules with dark light are added in the second stage; the black modules in the second stage are improved or optimized on the basis of the first stage modules. The solid arrow indicates the process of improvement and the dotted arrow shows the optimization.

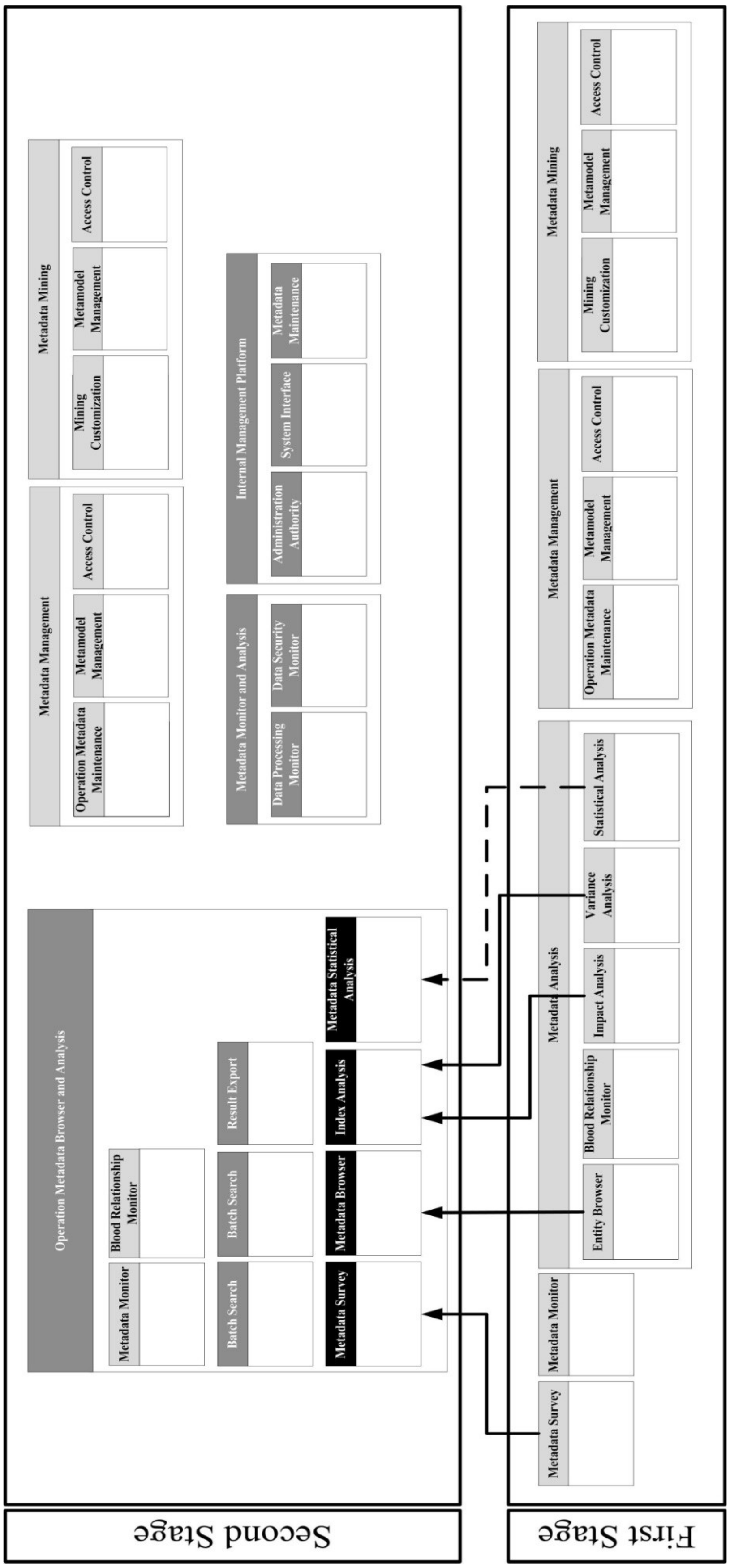


Fig. 8.4. Two-stage functional requirements

8.3.2.3 Process

For the first stage requirements, system developers search for components by adding the function keywords into the text area of the tool. The function keywords comprise from the top level system functions to the bottom level specific functions. More accurate keywords are added, more precise the result components are obtained. SQL-based search tool retrieve the components by matching with the contents of the component specification documents. The result components are listed in the order of the matched keywords from the most to the least.

Before the developers use the "most accurate" components for the further development, they still need considerable effort to read the component documents for the following reasons: 1) the keyword-based search is not efficient. It often results in too many or too few hits, or it may retrieve components that are completely unrelated. The search precision depends on the regulation and quality of component description. However, the component descriptions are written by different component developers textually in natural language. It is difficult to improve the search efficiency. For example, when you search for the keyword "**order** processing", the text "in **order** to", "re**order**", and "can not **order**" are also matched with the keyword; 2) the sizes of the result components vary from single specific function module to large multi-functions module. Predictably, the three sizes of result components will be retrieved in comparison with the function scope of the requirement, as shown in Figure 8.5. The first size of the result component (component 1) contains one or more Required Functions (RF), which is in the boundary of the required system functions scope. The second size, like component 2, part of its functions belongs to the required functions. Last, all the required functions are included in the component 3. The decisions of which components will be used need further analysis by considering other factors, such as effort for the composition, capability of the developer, further development for the second stage. After the component search, the developers still need much effort to select the components. Furthermore, the final decision depends largely on the background of the developers.

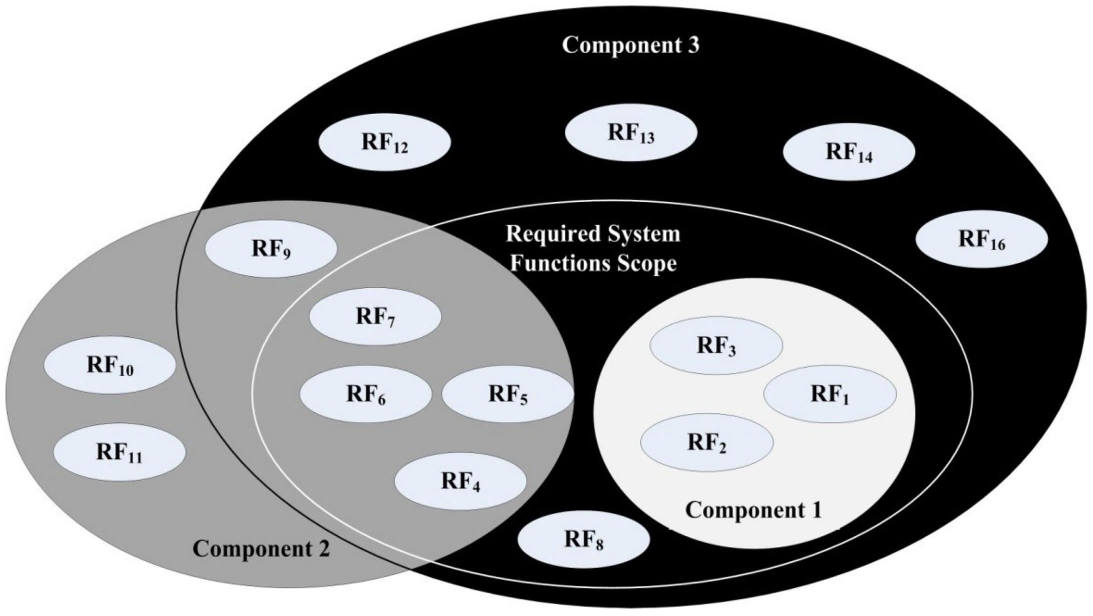


Fig. 8.5. Three sizes of result components compares with required system functions scope

For the second stage, the component search process and the problem encountered will be the same as the first stage. Furthermore, a new problem emerges in the process of function improvement or optimization: although the component adaptation is an easy way to reach the goal, the Mysoft method does not support the adaptation for the reason of lacking adaptive assets and poor support of the adaptation methods.

Although a large number of adaptation assets in similar projects are preserved, most of them are neither specified in the component repository nor linked to their related components. Therefore, adaptation assets/methods cannot be effectively used let along reused; this drawback reduces the efficiency of the CBD. The problems will be solved by the MVICS-based approach.

8.3.3 MVICS Component Specification and Retrieval

8.3.3.1 Introduction

The MVICS ontology model contains a large amount classes about the data management, which are organized into tree type architecture. Each class is formal defined to present one data management function and is not disjoint with others. The relationship between the super- and sub-classes and the inter-relationship among the different facets of the MVICS are investigated with the help of domain experts. Although the MVICS covers most data management functions classes, new functions may emerge which were missed during the rapid development of applications. In the MVICS-based component specification, the missing functions will be generated as new classes in the MVICS structure. The component information, related adaptive assets description and QAs-oriented factors are specified by completing a component specification registration form in the MVICS prototype tool (as shown in the Appendix C.3). After the registration, the data management components, their component adaptation assets and suggestion information are connected to the related classes in the MVICS.

8.3.3.2 Process

In the first stage, the same scenario is used to examine the MVICS approach in the component specification and retrieval. The keywords are first processed by the Synonym Operator to ensure that correct terms are used in the query. And then the refined keywords map against the MVICS component OWL file to find the matched classes automatically. At the same time, the relationships between the classes provide more semantics information to retrieve more related classes. In addition to the original component search, the adaptation component search occurs by matching the keywords with the adaptation OWL file. The search paths (original and adaptive) of each function are recorded for calculating the search precision. Finally, all the components related to the matched classes are identified as the result components for development. After the retrieval, the result components are listed in descending order of precision. For each result component, a comprehensive result profile of each component shows information from the summary to the details including: the result component name, overall precision of the component search, the match results in function, intrinsic and context model and the available adaptation method(s) or asset(s) with their efforts to apply.

8.3.3.3 Analysis

The use of the MIVCS-based approach eliminates or relieved the problems which occurred in the Mysoft method in the first stage. The problem of the wrong or unrelated results disappears due to the semantic matching between the refined query and the OWL format specification. The result component precision and the profile will help the developers to make a better decision with less effort and limited lower background. For the second stage, the component search process will be same as the first stage. The MVICS-based approach surpasses the Mysoft method in function improvement, and the implementation of optimization is easier with the help of adaptation matching result. The related adaptation assets/methods will be recommended, which offer more development options to the developer.

After the comparative test, the developers gave their feedback according to four criteria through the questionnaires, including user satisfaction of search precision and recall, required time, users satisfaction of adaptive assets/methods and human intervention.

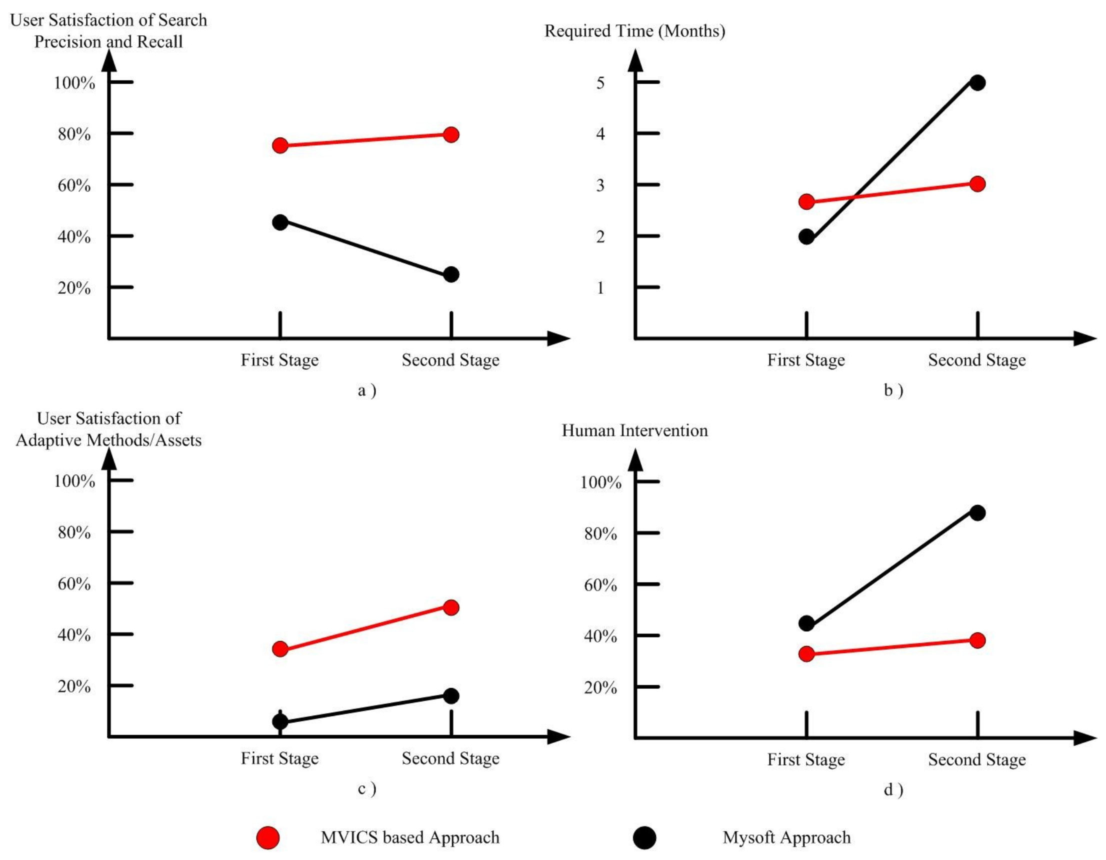


Fig. 8.6. Feedback analysis of the MVIVS-based approach and the Mysoft approach

Firstly, the MVICS-based approach improves search recall and precision to some extent. The results (Figure 8.6 a) are similar to the prototype experiment, which demonstrates the MVICS-based approach is suitable for industrial applications. Secondly, for large CBD development, the MVICS approach can shorten the development cycle (as shown in Figure 8.6 b) by its powerful description capability and logical definition, which can bring highly effective and fast component specification. In addition, the use of MVICS-based component registration is also an influential factor. Thirdly, the ignored adaptive methods/assets in the Mysoft approach are now suggested to the developers, and the resulting improvement is shown in Figure 8.6 c). In particular, from the first to the second stage, part of the components used in the first stage are adapted with the help of the adaptive suggestion, rather than finding new components; this is another reason for the shortened development time. Finally, because the MVICS method improves the search recall and precision, provides the adaptive assets/methods and corresponding adaptive suggestion, the development experience requirements for the developers and the efforts for the following system development are reduced. Finally, the effects are expressed in the reduction of the human intervention, as shown in Figure 8.6 d).

8.4 Racing Game Case Study

Unlike the second case study, this case study focuses on illustrating the ability of the MVICS-based approach to develop a domain specific system. Firstly, the aim of the case study is to examine the strength of the linkage with domain ontology mechanism in real component-based development. In addition to of the improvement of the search efficiency and the automated capability in development of component-based applications, the benefits can be delivered into specific domain software development by using the linkages (*AssL* and *AggL*) of the MVICS. Furthermore, the efficiency of Quality Attribute suggestion will be reflected in the further game platform deployment.

8.4.1 Background

This case study is in collaboration with the Mobile Game Development Department of Gameloft Co., Ltd in Beijing [93][103]. Gameloft (Euronext: GFT) ,a French publisher, primarily creates games for mobile phone handsets equipped with platforms including iOS [76], Android [74], Window Phone [78], BlackBerry OS [75], Symbian OS [77] etc. There are various types of classic mobile games which have been published in recent years, such as Asphalt series, Block Breaker series, Platinum Solitaire series, Guitar Rock Tour series and Uno.

The Mobile Game Development Department Beijing branch mainly focuses on developing Sports Games (SPG), for example, more than 8 types, 12 patterns and 45 editions of Racing Games (RAG) have been developed. There are large numbers of racing background components, racing car model components, racing control components, database management components and sound processing components that can be reused in the future development of similar games. However, the reuse oriented game development is still at white-box level, which restricts their efficiency. The current situation of component specification and retrieval is the same as the MysSoft in the last case study. The code, software packages, or modules obtained in the previous projects, which implement one or more functions, are encapsulated as components and are described by their specified interface. A local department repository was built to store the components and help the developers reuse by traditional component retrieval methods. The existing Gameloft component specification of retrieval provides a way to improve the development efficiency, however, it is still not nearly as satisfying as developers expected, due to the low precision and recall of the component search. In particular, the existing methods do not consider the influence of component QAs, which will affect the process of platform transplantation and even cause redevelopment. As a starting point, a case of City Racing Game is selected to test the improvement of the MVICS-based approach in the component-based development.

8.4.2 MVICS-Based Gameloft Component Specification and Retrieval

The application of the MVICS-based approach to the City RAG consists of the following steps: i) Development of RAG domain ontology; ii) linkage between MVICS and RAG domain ontology; iii) MVICS-based component specification and retrieval; iv) MVICS-based QAs suggestion. The last step is actually realized in the process of the MVICS-based component specification and retrieval.

8.4.2.1 RAG Domain Ontology

The use of an accurate RAG Domain Ontology is a key part of the application of the MVICS-based approach in this case study. However, it is difficult to retrieve an exiting domain ontology that fully meets the requirements in practice. Usually the structure and contents of the selected ontology need to be modified, added and even rebuilt.

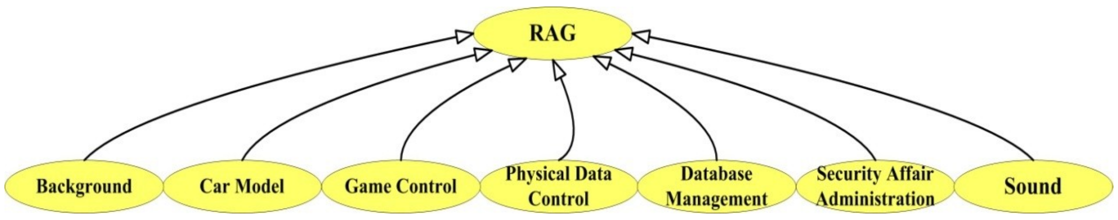


Fig. 8.7. The top level classes of RAG domain ontology

The basic RAG domain ontology used in the case study was retrieved from the game ontology library [92]. According to the requirements of RAG, the framework of the RAG domain ontology was updated, which consists of seven parts, including the Background, Car Model, Game Control, Physical Data Control, Database Management, Security Affair Administration and Sound, as shown in Figure 8.7. The seven parts are recorded as the top level classes, and their sub-classes, sub sub-classes are built by migrating other existing RAG related ontologies, such as Game Database Management ontologies and Game Security Affair Administration ontologies.

The RAG related ontologies were retrieved from the protégé ontology library [85], with help of Google filetype search [35]. The relevance specific classes in the selected RAG related ontologies were rearranged under the class *Background*, class *Car Model*, class *Game Control*, class *Physical Data Control* and class *Sound* of the new RAG domain ontology.

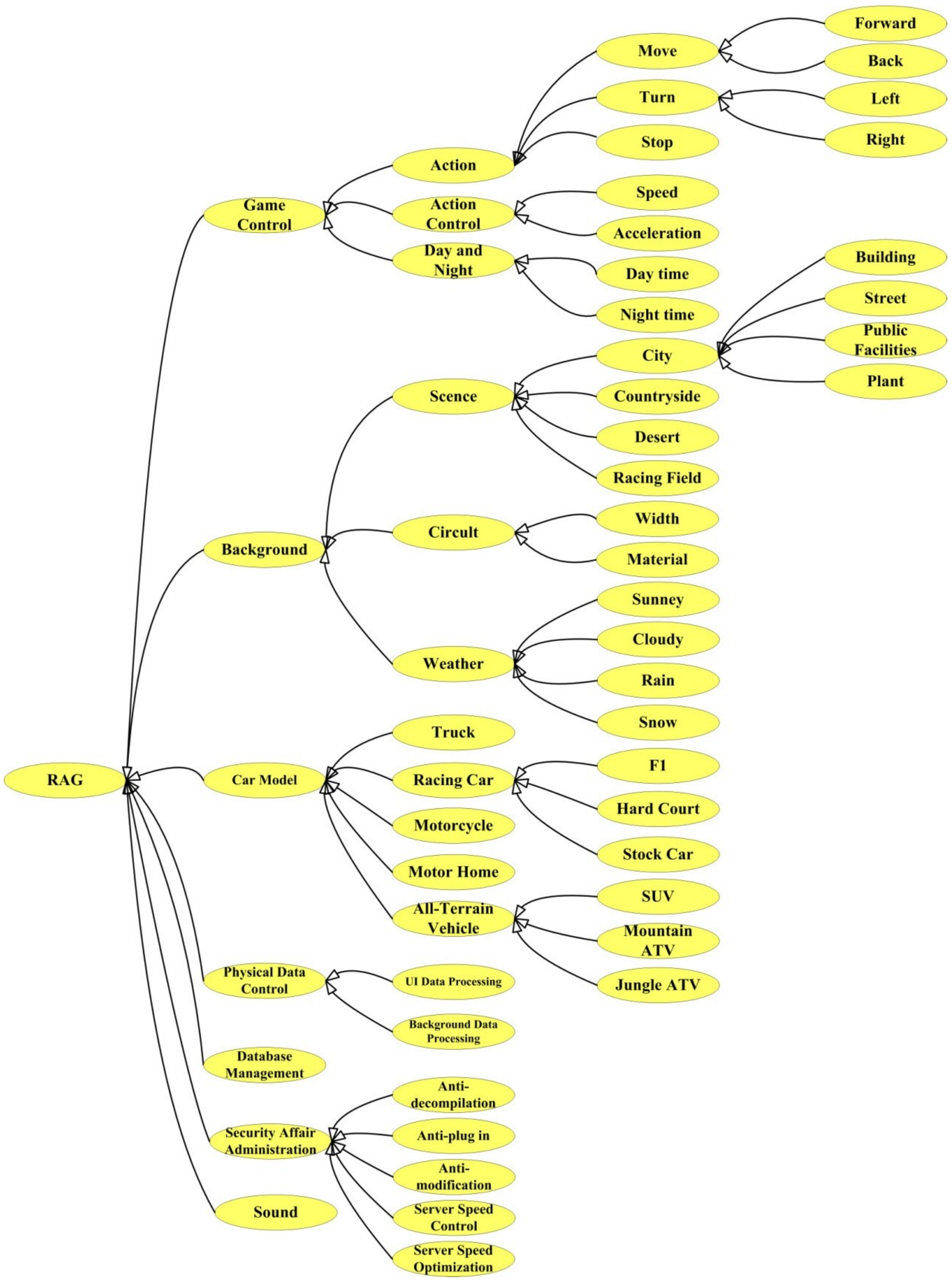


Fig. 8.8. RAG domain ontology

The method to construct the class *Game Database Management* and the class Game *Security Affair Administration* is the same: their sub-classes migrate from the relevance existing ontologies. Finally, the new RAG related ontology is updated by recording and adding the new features of Gameloft City Racing Game, which cannot be obtained from the existing ontologies. Each class in this ontology represents one module that carries out a functional requirement in the development of the City RAG. Superior-subordinate relationships have been used to describe the affiliations of the related modules. The classes of RAG domain ontology is shown in Figure 8.8 and its detail list in Appendix D.

8.4.2.2 Linkage Construction

To extend the semantic-based component search into the RAG domain, classes of the RAG domain ontology are linked to MVICS, either through *AssL* where a corresponding super/sub-class relation exists, or through *AggL* where a domain class is composed of one or one set of *Aggregations* in MVICS. To automate the integration between domain ontology and MVICS, the following steps are taken to support the semi-automation of the integration: i) *Association class* identification; ii) Operation module analysis; iii) *Aggregation* update.

- *Association Class Identification*

  Except for the existing class *Database Management* and class *Sound* in the MVICS, the class *Background*, class *Car Model* and class *Game Control* represent specific operations, as a specialization of relevant MVICS classes in the RAC domain. These classes and their sub-classes, sub sub-classes can be viewed as sub-classes of their relevant MVICS classes. Therefore, the above four classes are identified as the *Association Classes*, and *AssL* is used to link these classes with their super class counterparts in MVICS. The class *Background* and class *Car Model* are viewed as the sub-classes of Class *Rendering Model* of the MVICS; the sub-classes of class *Game Control*, including *Move*, *Turn*, *Stop*, *Acceleration* can be seen as the sub-class of class *Physical Engine* of the MVICS; and these RAG ontology classes are linked with MVICS by *AssL*.

- *Aggregation Analysis*

Differing from the classes in RAG domain ontology that can be linked via *AssL*, the sub-classes of class *Physical Data Control* and class *Security Affair Administration* are more comprehensive and multifunctional. In this case, we first analyze the *Aggregations* in terms of the IT function in MVICS, which represent the group functions of the Data Control and Security Administration operations. These *Aggregations* are viewed as reusable units oriented to different Physical Data Control and Game Security Administration operations in the RAG domain. The *AggL* is used to link RAG domain classes to the relevant *Aggregations* in MVICS. For example, the *Aggregation* (*Data management 2*) in the MVICS model comprises the sub-function classes *Data Acquisition*, *Data Processing*, *Data Validation*, *Data Conversion* and *Data Output*, which can cooperate to implement the Physical Data Control for the RAG. Thus, the class *Physical Data Control* is linked to the *Aggregation (Data management 2)* through the *AggL*.

- *Aggregation Update*

In the context of linkage with class *Security Affair Administration*, a problem encountered is that Security Affair Administration for the RAG involves issues of interaction with the server. It means that the Class *Security Affair Administration* not only links to *Aggregation (Security 1)* but also the Server Speed related *Aggregation.* However, there is no such kind of *Aggregations* in MVICS. In this case, the *Aggregation* of Server Speed, namely *Aggregation (Server Speed Management)*, should be added into the MVICS. After the new *Aggregation* update, the class *Security Affair Administration* in RAG domain ontology is linked to the MVICS by *Aggregation (Security 1)* and *Aggregation (Server Speed Management)*.

8.4.2.3 MVICS-Based RAG Component Specification and Retrieval

As the linkages were defined formally, the MVICS-based RAG component specification and retrieval work is the same as the financial domain example. The already built components and related adaptation assets are specified with help of the

MVICS and the RAG domain ontology. The QAs related factors of each component are identified at the same time as the preparation work for the QAs suggestion. During the search, the selected deployability and maintainability of the result components are considered. The advantages of search precision and recall, adaptation suggestion and result display are also presented. In addition, the use of MVICS-based method relieves the problem that occurred in the process of racing game transplantation.

After its initial development, the racing game is adapted to many editions in accordance with mobile devices by the Game Transplantation Group. The implementation of the racing Game is restricted by CPU, RAM, OS and size of display screen of mobile device. Usually the transplantation Group needs to modify or delete some non-requisite components to meet the hardware requirements, and sometimes redevelop some functions of the game, such as change some requisite components.

Because the MVICS-based approach provides the QAs suggestions, it can solve the above problems in transplantation at a certain level. The features of a component which affect their deployrability and maintainability are stored as forms in the class Quality Attribute of the function model. Deployability [37][60][109] is defined as the ability of a component to run in different computing environments. It does not imply that the application will simply require to be recompiled for the new platform. Neither does it allow for major re-engineering projects. Deployability is the ability to provide the application on another system with known and economically reasonable cost and effort [14]. Maintainability [3][115][142]is defined as a capacity to undergo repairs and modifications. Maintenance is of three different types, corrective, adaptive and perfective maintenance [4]. Corrective maintenance is the removal of faults, thereby improving the reliability of components under the condition that no new faults are introduced. Adaptive maintenance is performed when a component is modified to meet modifications in the target environment. Perfective or preventive maintenance addresses improvements of the components in response to users or designers' input [123]. The result components that support multiple platforms, convenient modification and easy deployment, are recommended in priority to the user.

8.4.2.4 Analysis

The implementation of the MVICS approach in this case study is led by Mr. Xikun Yang. Other two developers give assistance to developing the ontology, specifying and retrieving the components, and developing the RAG game. Their feedbacks proved again that the MVICS-based approach improves the search recall and precision in practical applications. At the same time, it reduced the problems cause by the human intervention to some extent. Because the MVICS-based approach avoids the situations that the manual component specification and retrieval, inexperienced component adaptation and software transplantation without component QAs suggestion. Users satisfaction of the MVICS-based approach and the Gameloft approach as shown in figure 8.9.

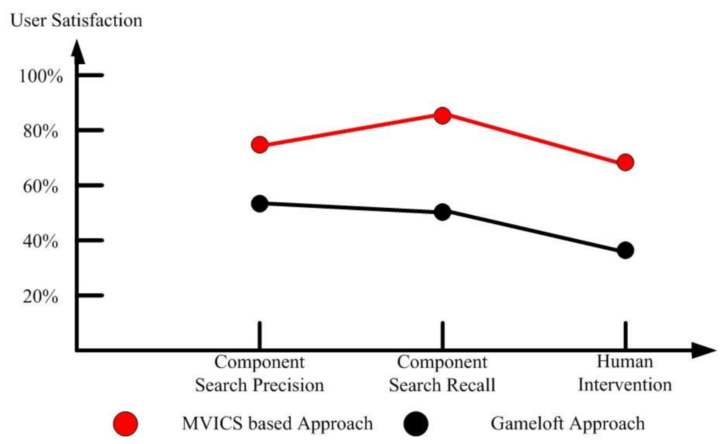


Fig. 8.9. User satisfaction of the MVICS-based approach and the Gameloft approach

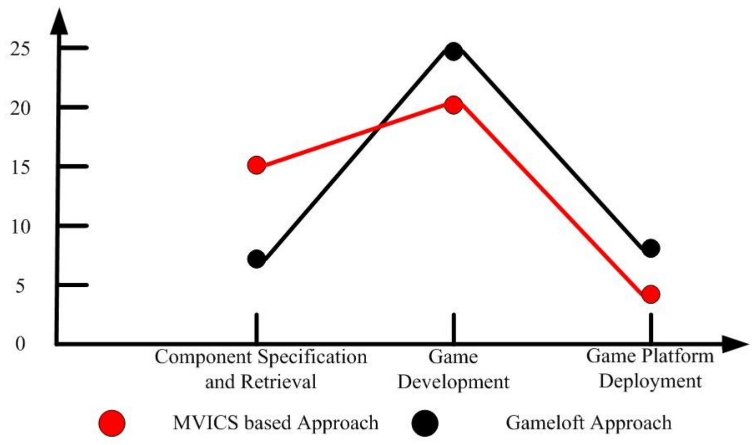


Fig. 8.10. The statistics of working days for each phase

However, a noteworthy circumstance is both approaches took about seven weeks on the condition that the MVICS-based approach saves time for the game development obviously. The MVICS approach saved eight working days in the phases of development and transplantation, whereas there time was almost wasted in the component specification and retrieval phase. The statistics of working days for each phase is shown in Figure 8.10.

The MVICS-based approach took more than eight working days to build the RAG ontology and the linkage with MVICS ontology. For the Gameloft Beijing branch, the MVICS-based approach can only save the time for the next Racing Game development. If the next project is another type of game development, such as Role Playing Game (RPG) or Action Game (ACT) Puzzle Game (PUZ), part of the class *Physical Data Control*, class *Database Management*, class *Security Affair Administration* and class *Sound* of RAG ontology can be reused after minor changes. Mostly the MVICS-based approach needs to re-establish the corresponding game domain ontology and its linkage with MVICS, which undoubtedly reduces the attraction of MVICS in actual applications.

8.4.3 Extended Game Domain Ontology Model

To make the MVICS-based approach fully deployed in the Gameloft game development, and to avoid extra time and resource consumption of establishment of the appropriate ontology and its linkage, the RAG ontology needs to be updated to a generic game development ontology for the Gameloft.

According to the common features of game development for various game types, the top level classes of RAG domain ontology needs to be redefined in order to improve its description capability, which can cover all types of games in the Gameloft. The top level classes of the updated Gameloft game ontology comprise class *Graphics Rendering*, class *Game Control*, class *Physical Data Control*, class *Database Management*, class *Security Affair Administration* and class *Sound*, as shown in Figure 8.11 (the details of the Gameloft ontology is given in Appendix E). The changes are mainly manifested in the class *Graphics Rendering* and class *Game Control* compared with the RAG ontology.

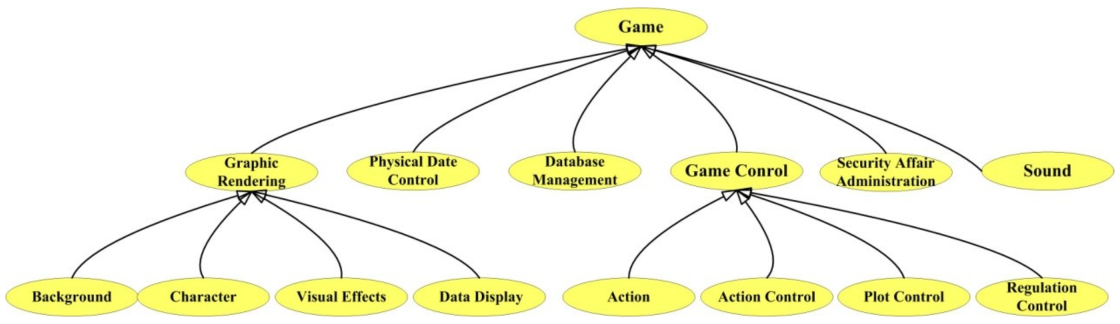


Fig. 8.11. The top two level classes of the Gameloft ontology

The class *Graphics Rendering* refers to coverings all types of the graphics in the game, which includes the *Background*, *Character*, *Visual Effects* and *Data Display*. The sub-class *Background* here is the same as the homonymous class in the RAG ontology. The contents of the new Background are more comprehensive. It includes not only the Race game, but also the background of the game RPG, ACT and PUZ. The class *Character* extends the class *Ccar Model* of the RAG ontology. It describes the all the types of the *Characters* occurring in the developed game. The class *Visual Effects* and class Data Display are new in the Gameloft ontology. The *Visual Effects* mainly affect the large 3D RPG and ACT games which are not used in the existing RAG games. The Data Display is listed separately from the class *Physical Data Control* in the RAG ontology, because the Data Display is more important in the other types of games.

The content of the class *Game Control* is improved that mainly embedded in the new sub-classes *Plot Control* and *Regulation control*. The Game Control here is not only the simple movement of a Character and the change of the background. It refers to the change and progress of plot for the games, such as the RAG mission and ACT. The *Regulation Control* focuses on the PUZ type game that should follow the PUZ regulations. The rest of the top-level classes are similar to the RAG ontology, and needed an update to some new classes according to the new requirements.

Through the establishment of the Gameloft ontology, the MVICS approach can be used in all types of the game development in the Gameloft with less time consumed in re-building the new ontology. The pre-preparation work only refers to update some classes according to the new specification of the components which greatly saves time of MVICS approach deployment.

8.5 Summary

The feedbacks from the three case study shows that the MVICS-based approach has the five capabilities in the prototype text and practical use.

The test results from the first case study shows that the MVICS-based approach improves *recall*, *precision*, *result display*, and *adaptation suggestion* effectively, in particular, on the criteria of result display and adaptation suggestion. The result profile and adaptation suggestion are new mechanisms that developed in MVICS in contrast to other existing component search approaches. For the criteria *search speed* and *maintenance effort*, the MVICS and traditional approaches cut both ways under different conditions, but all within a reasonable range. By testing the prototype tool for a large number of scenarios, the initial validation results confirm the improvement of the component specification and retrieval in these five aspects has been achieved by the MVICS approach.

On the basis the prototype test, the second case study validates the MVICS approach in real applications. The implementation of the two-stage large Metadata Management system demonstrates that the MVICS-based approach can be deployed in real life industrial development. The benefits of MVICS include not only the improvement of the component search precision, reducing the development time, but also the decrease of the human intervention on CBD.

The last case study validates the advantages of the MVICS-based approach, including the improvement of search precision and recall, availability in specific domain, suggestion of adaptive assets/methods and reduction of human intervention. Especially, the QAs suggestion in the MVICS-based approach can play a supplementary role in the later game transplantation phase. However, due to the dependency of the MVICS approach on the domain ontology and the RAG domain ontology established in this case study only applicable to racing game, the MVICS approach may be limited to wider practical applications. The updated Gameloft ontology makes the MVICS approach applicable for all types of game development in the Gameloft, which solves the above-mentioned application problem.

9. Conclusions and Future Work

The objectives of the research have been achieved by developing the MVICS ontology-based approach to solve the component mismatch problem via holistic, semantic-based, adaptation-aware and QAs oriented component specification and retrieval, and extending the search scope to domain ontology via the developed semantic links. The literature investigation has shown that the proposed approach has novel contributions to the research area. This chapter first concludes the contributions of the MVICS-based approach according to the proposed objectives in the Chapter 1 and then discusses the possible future directions of the research.

9.1 Conclusions and Novel Contributions

9.1.1 Conclusions

With the continuous expansion of software applications and the increase in component varieties and size, it is becoming more imperative in component-based development to solve the problem of component mismatch. A major hurdle for wider and smoother component reuse is the lack of automated and effective approaches to component specification and retrieval. Although domain model and ontologies have been attempted in component specification and retrieval, producing improved results, it is clear that the ontologies in existing approaches have too simple and/or monolithic a structure and too few relationships to deal with the specification and retrieval of modern components. To correct the limitations of existing work, an approach based on a multiple-viewed interrelated component specification ontology model has been built and evaluated. The following work has been undertaken during the development of the approach:

1) **A holistic ontology-based component specification and retrieval approach**

   A MVICS-based approach is developed to achieve holistic and semantic-based component specification and automatic and precise component retrieval (Chapter 5 and 6). The approach has two characteristics, namely “holistic” and “semantic-based”. “semantic-based” here refers to that the component specification ontology and the domain specific ontology cooperates to provide the semantics foundation of the component specification. And “Holistic” here refers to that the MVICS is a comprehensive component specification model, and the approach considers a spectrum of perspectives in component specification and retrieval. The approach effectively supports: the original component search, the domain component search, the adaptive component search, the result component precision calculation and the result component profiles, all of which together enable the user to find more suitable components effectively and automatically.

2) A prototype tool

As the implementation of the MVICS-based approach, a prototype tool (chapter 7) has been developed to exert the power of the MVICS model in expressing semantics and process automation in component specification and retrieval. The tool implements the complete process of component search, starting from filling the initial query and ending up with receiving the result component profile. The tool consists of a set of modules, which impact on component search, including Dynamic Class Weight Assignment, Search Precision Calculator, Search Time Recorder, and QAs Suggestion Processer.

3) Case studies and evaluation

Three case studies (chapter 8) have been undertaken to illustrate and evaluate the usability and correctness of the approach, in terms of supporting the precise component specification and retrieval, seamless linkage with domain ontology, adaptive component suggestion and comprehensive result component profile.

9.1.2 Contributions

As a solution for the identified problems, the MVICS-based approach originally contributes to the current state of art by producing the following key technologies, which are integrally linked in a holistic and semantic-based specification and retrieval framework:

1) **Multiple-viewed and interrelated component specification ontology model** (Section 5.1- Section 5.4)

   The MVICS ontology model has a novel architecture, which contains four facets: function model, intrinsic model, context model and meta-relationship model. Each of the four models specifies one perspective of a component and as a whole they construct a complete spectrum of semantic-based component specification. It removes the over-complication problem in traditional monolithic ontologies because it has a set of highly coherent and relatively loosely coupled sub-models. The inter-relationships among the classes in different sub-models ensure a holistic view in component specification and retrieval. A formal definition of the classes, relationships and constraints of MVICS is given in OWL-DL. Via these formal definitions, the automatic semantic-based component search and validation are achieved with the support of an ontology reasoner. Overall, the role of the MVICS model includes supporting semantic query refinement, conducting the component specification, connecting the domain ontology and supporting the sub-functions in the component retrieval, such as search precision calculation, adaptive component search and component QAs suggestion.

2) **Domain ontology linkage technique** (Section 5.5)

To extend the MVICS-based component search into a specific domain, two mechanisms, namely *Association Link* (*AssL*) and *Aggregation Link* (*AggL*), are developed to integrate the domain related software system ontology into MVICS. The *AssL* and *AggL* are used to link two different kinds of classes of domain ontology to the MVICS. Furthermore, the *AssL* and *AggL* are defined formally and to automate the component search and repository building. With such integration, the domain ontology is linked to MVICS effectively and thus extends the application scope of MVICS without changing the architecture of the model.

3) **MVICS-based component retrieval method** (Section 6.1- section 6.3)

The MVICS-based component retrieval is based on the MVICS model and the linkage with the domain ontology model. It focuses on retrieving the relevant components from the repository according to the refined user keywords and providing an accurate search precision. As a unique feature of the MVICS-based component retrieval, the adaptive component can be matched by identifying the adaptive search path. And the results provide not only the matched components with the relevant adaptation assets/methods, but also their suggested effort. Finally, as the focus of the MVICS-based component retrieval method, a result component precision calculation method is developed to indicate the matching degree between a result component and the user requirement. The precision values of the outcome are accurate, because it is obtained through a search precision calculation algorithm, which is established on the basis of the MVICS.

4) **Comprehensive result component profile** (Section 6.4)

A holistic Result Component Profile is designed for the result component to help the user make the best decision in component selection. The profile shows the result from the whole to the part, which consists of: i) the result component name; ii) search summary, including the precision with component adaptation, and the precision without adaptation; iii) the match results in sub-models: function model, intrinsic model, and context model; iv) the match results in a domain ontology model; v) the associated adaptation method or asset and its incurred effort.

9.2 Limitations and Future Work

Based on the discussions in former sections, it is concluded that the approach has novel contributions and is successful in component specification and retrieval. The resulting tool automates the approach and is consistent with the approach. However, the MVICS-based approach still has some limitations:

To face a torrent of new emerging characteristics of components, the structure, classes, attributes and relationships, the MVICS model need to keep expanding and/or fine-tuned accordingly. The new relationships among the classes in the different facets need to keep expanding and being refined to accommodate the new component characteristics.

The MVICS model connects to the domain specific ontology by the *AssL*s and the *AggL*s. However the linkages are established through the cooperation of domain experts and software developer, which involves much human intervention.

The formulas proposed in the chapter 6 lack of the support from the mathematics theory. The dynamic class weight calculation, formulae of search path weight calculation and the formulae of precision calculation need justification and further refinement.

The level of the adaptation suggestions at this stage is proposed on the basis of our background, It is still weak and lack of widely acceptance.

In the future, the following sections have explored some possible extensions of the present work.

1) **To increase the automation of the establishment of domain ontology linkage**

New techniques or mechanisms will be explored to automate the linkage establishment by using more rigorous formal definition. The human intervention of the process will thus be reduced with the help of relevant reasoners.

2) **To refine of the precision calculation**

With wider and more user feedbacks and industrial test results, it is possible to achieve further refinement of the formulae of dynamic class weight calculation, formulae of search path weight calculation and the formulae of precision calculation.

3) **To establish adaptation ontology model**

In the MVICS-based approach, the adaptive component search is taken as an integral part of the MVICS-based approach. The component can be identified as a result component, if matched with the requirement after adaptation with the relevant methods or assets. Therefore a new facet namely “adaptation facet” may be considered in the MVICS. to extend the functionality of the existing adaptive component search.

4) To improve of the quality attributes suggestion

In the MVICS case study, the impacts of the quality attributes are introduced in the component specification and retrieval. The functionality of the QAs suggestion can be subjected to further testing and expansion, with the help of the feedback from the industrial software development. Along with these practices, more QAs will be analyzed which may help build a QAs facet into the MVICS model.

5) To validate the feedbacks from the case studies

The outcome of the present experimental case study obtained via questionnaires can be validated through future practical case studies. Drawbacks of MVICS-based approach will then be identified and overcome along with more scenarios of MVICS-based approach applied in practice.

Reference

[1] Aggarwal, K. K., Singh, Y. and Chhabra, J. K. (2002). An Integrated Measure of Software Maintainability. *Proceedings of Annual Reliability and Maintainability Symposium*, IEEE.

[2] Åkerholm, M. (2008). Reusability of Software Components in Vehicular Applications. *PhD Thesis*, Mälardalen University.

[3] Ash, D., Alderete, J., Yao, L., Oman, P. W. and Lowther, B. (1994). Using Software Maintainability Models to Track Code Health, *Proceedings of International Conference on Software Maintenance*, IEEE.

[4] Avizienis, A., Laprie, J. C., Randell, B. and Landwehr, C. (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Transactions on Dependable and Secure Computing*, Volume 1, No. 1, Pages 11-33.

[5] Baader, F., Horrocks, I. and Sattler, U. (2004). Description Logics. Handbook on Ontologies. *International Handbooks on Information Systems*, Pages 3-28. Springer.

[6] Baker, T. (2002). Lessons Learned Integrating COTS into Systems. *Proceeding of International Conference on COTS-based Software Systems*, Orlando, FL: Springer-Verlag.

[7] Balk, L. D., and Kedia, A. (2000). A COTS Integration Case Study. *Proceedings of International Conference on Software Engineering*, Limerick, Ireland: ACM Press.

[8] Barbacci, M., Klein M., Longstaff, T. and Weinstock, B. (1995). Quality Attributes. *Technical Report*, CMU/SEI-95-TR-021, ESC-TR-95-021.

[9] Bass, L., Buhman, C., Comella-Dorda, S., Long, F., Robert, J., Seacord, R. and Wallnau, K. (2000). Market Assessment of Component-based Software Engineering. *Technical Report*, Volume I, 2000, CMU/SEI - Carnegie Mellon University/Software Engineering Institute.

[10] Bass, L., Clements, P. and Kazman, R. (1998). Software Architecture in Practice, *Addison-Wesley*.

[11] Batory, D. (1998). Product-Line Architectures. *Smalltalk & Java in Industry and practical Training*, Erfurt, Germany.

[12] Bechhofer, S., Goble, C. and Horrocks, I. (2001). Daml+oil Is Not Enough. *Proceedings of the First Semantic Web Working Symposium (SWWS'01)*, Pages 151-159.

[13] Beck, K., Crocker, R., Meszaros, G., Vlissides, J., Coplien, J. O., Dominick, L. and Paulisch, F. (1996). Industrial Experience with Design Patterns. *Proceedings of the 18th International*

Conference on Software Engineering, Berlin, Germany, IEEE Computer Society Washington, DC, USA.

[14] Bell, A. (2003). Portable GUI Development. *Tessella Support Services plc.* Issue V1.R4.M0.

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

[16] Birngruber, D. (2001). A Software Composition Language and Its Implementation, Perspectives of system informatics. *Proceedings of the 4th International Andrei Ershov Memorial Conference*, Akademgorodok, Novosibirsk, Russia.

[17] Bondavalli, A., Chiaradonna, S., Cotroneo, D. and Romano, L. (2004). Effective Fault Treatment for Improving the Dependability of COTS and Legacy-based Applications. *IEEE Transactions on Dependable and Secure Computing*, Volume 1(4), Pages223-237, 2004.

[18] Bosch, J. (1999). Superimposition: A Component Adaptation Technique. *Information and Software Technology*, Volume 41(5), March 1999.

[19] Braga, R., Mattoso, M. and Werner, C. (2001). The Use of Mediation and Ontology Technologies for Software Component Information Retrieval. *Proceedings of the* 2001 *symposium on Software Reusability*, Page. 19-28.

[20] Braga, R., Werner, C. and Mattoso, M. (2006). Odyssey-Search: A Multi-agent System for Component Information Search and Retrieval. *Journal of Systems and Software*, Volume 79(2), Pages. 204-215.

[21] Bray, T. J., Paoli, C., Sperberg-McQueen, M. and Maler, E. (2000). Extensible Markup Language (xml) 1.0 (second edition).*w3c recommendation*, 6 October 2000.

[22] Carroll, J., Bizer, C., Hayes, P. and Stickler, P. (2004). Named Graphs, Provenance and Trust. *Proceedings of the 14th International World Wide Web Conference.*

[23] Ceria, S., Nobili, P. and Sassano, A. (2005). A Lagrangian-based Heuristic for Large-scale Set Covering Problems. *Mathematical Programming*, Volume 81, No 2.

[24] Cheesman, J. and Daniels, J. (2001). UML Components, A Simple Process for Specifying Component-Based Software, *Addison-Wesley*, 2001.

[25] Crnkovic, I. (2001). Component-based Software Engineering – New Challenges in Software Development. *Software Focus,* Volume 02, No. 04, 2001, Pages. 27-33.

[26] 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, Pages 257-278. 2005.

[27] Coleman, D., Ash, D., Lowther, B. and Oman, P. (1994).Using Metrics to Evaluate Software System Maintainability. *IEEE Computer,* Volume 27, Issue 8, 1994.

[28] Coplien, J. O. and Schmidt, D. C. (1995). Pattern Language of Program Design. *Addison-Wesley Professional*.

[29] Councill, B. (2001). Third-Party Certification and Its Required Elements, *Proceedings of the 4th Workshop on Component-Based Software Engineering (CBSE)*, Canada, May, 2001.

[30] Dale, R., Somers, L. and Moisl, H. (2000). Handbook of Natural Language Processing. *Marcel Dekker*, 2000.

[31] Davis, L., Gamble, R. and Payton, J. (2002). The Impact of Component Architectures on Interoperability. *Journal of Systems and Software*, 61:31–45, 2002.

[32] Decker, S., Erdmann, M., Fensel, D., Horrocks, I., Klein, M. and Van Harmelen, F. (2000). Oil in A Nutshell. *Proceedings of the 12th International Conference in Knowledge Engineering and Knowledge Managemen (EKAW'00)*, France, 2000.

[33] Denny, M. (2004). Ontology Tools Survey, Revisited. *http://www.xml.com/ pub/a/2004/07/14/onto.html, 2004*.

[34] D'Souza, D. and A. Wills. (1999). Objects Components, and Frameworks: The Catalysis Approach. *Addison-Wesley*.

[35] DuCharme, B. (2004). Googling for XML. http://www.xml.com/pub/a/2004/02/11/ googlexml.html.

[36] Due, R. (2000). The Economics of Component-Based Development. *Information Systems Management*, Volume 17 (1), 2000.

[37] Egyed, A. and Gacek, C. (1999). Automatically Detecting Mismatches During Component-based and Model-based Development, *Proceeding of the 14th IEEE International Conference on Automated Software Engineering*, Florida, USA, Pages 191–198.

[38] Feldmann, R., Geppert, B., Mahnke, W., Ritter, N. and Roessler, F. (2000). An ORDBMS-based Reuse Repository Supporting the Quality Improvement Paradigm -- Exemplified by the SDL-Pattern Approach. *Proceedings of the 34th International Conference on Technology of Object-Oriented Languages and Systems - TOOLS*. Pages 125-136.

[39] Fayad, M. E. and Schmidt, D. C. (1997). Object-oriented Application Frameworks. *Communications of ACM*. Volume 40(10), Pages. 32-38.

[40] Fensel, D., Crubezy, M., Van Harmelen, F. and Horrocks, I. (2000). Oil & Upml: A Unifying Framework for the Knowledge Web, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, Berlin, Germany, 2000.

[41] Fettke, P. and Loos, P. (2002). Specification of Business Components International Conference NetObjectDays on Objects. *Components, Architectures, Services, and Applications for a Networked World*, Springer-Verlag.

[42] Fikes, R. and Farquhar, A. (1999). Large-Scale Repositories of Highly Expressive Reusable Knowledge, *IEEE Intelligent Systems*, Volume 14(2), 1999.

[43] Francalanci, C. and Fuggetta, A. (1997). Integrating Conflicting Requirements in Process Modelling: A Survey and Research Directions. *Information and Software Technology*, Volume 39 (March 1997), Pages. 205-216.

[44] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-oriented Software. *1st edition, Addison Wesley Professional.*

[45] Gao, T., MA, H., Yen, I-Ling., Khan, L. and Bastani, F. (2006). A Repository for Component-Based Embedded Software Development. *International Journal of Software Engineering and Knowledge Engineering*, Volume. 16 (4), Pages. 523-552.

[46] Garcia, V., Lucrédio, D., Durão, F, et al. (2006). From Specification to Experimentation: A Software Component Search Engine Architecture. *Proceedings of the 9th International Symposium on ComponentBased Software Engineering (CBSE 2006)*, Volume 4063, Springer-Verlag, Pages 82-97, 2006.

[47] Genesereth, M and Fikes, R. E. (1992). Knowledge Interchange Format (Version 3.0). *Reference Manual*. Computer Science Dept., Stanford University, Stanford, CA.

[48] Gennari, H., Musen, A., Fergerson, W., Grosso, E., Crubezy, M., Eriksson, H., et al. (2003). The Evolution of Protege: An Environment for Knowledge-based Systems Development. *Human-Computer Study*, 2003, Volume 58(1), Page 89–123.

[49] Girardi, R., Faria, D. (2004). An Ontology-based Technique for the Specification of Domain and User Models in Multi-agent Domain. *CLEI Electronic*, 2004.

[50] Girardi, R. and Serra, I. (2004). Using Ontologies for the Specification of Domain-Specific Languages in Multi-Agent Domain Engineering. *Proceedings of the 2004 Agent_Oriented Information Systems Workshop (AOIS 2004)*, *2004 Conference on Computer Aided Software Engineering*, Riga, 2004.

[51] Golbeck, J., Parsia, B. and Hendler, J. (2003). Trust Networks on the Semantic Web. *Proceedings of Cooperative Intelligent Agents (CIA03).*

[52] Goldberg, A. and Rubin, K. S. (1995). Succeeding with Objects: Decision Frameworks for Project Management, *1st edition (May 1, 1995), Addison-Wesley Professional.*

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

[54] Grinter, R. E. (2001). From Local to Global Coordination: Lessons from Software Reuse. *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, Boulder, Colorado, USA, 2001, Pages 144-153.

[55] Gruber, T. (1993). A Translation Approach to Portable Ontology Specications. *Knowledge*

Acquisition, Volume 5, Pages 199 220, 1993.

[56] Gschwind, T., Oberleitner, J. and Pinzger, M. (2003). Using Run-Time Data for Program Comprehension. *Proceedings of the 11th IEEE International Workshop on Program Comprehension (IWPC'03)*, Oregon, USA on May 10-11 2003, IEEE Computer Society.

[57] Guo, J. and Lu, Q. (2000). A Survey of Software Reuse Repositories. *Proceedings of the IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, 2000, Edinburgh, Scotland.

[58] Haarslev, V. and Moller, R. (2001). Description of the RACER System and Its Applications. *Proceedings of the International Workshop in Description Logics 2001 (DL2001).*

[59] Hall, J. (1993). Generalized Behavior-Based Retrieval, *Proceedings of the 15th International Conference on Software.*

[60] Hakuta, M. and Ohminami, M. (1997). A Study of Software Portability Evaluation. *Journal of SYSTEMS SOFTWAREI,* 1997, 38:145-154.

[61] Han, J. (1998). A Comprehensive Interface Definition Framework for Software Components. *Proceedings of Asia-Pacific Software Engineering Conference* (*APSEC'98*).

[62] Han, J. (1999). An Approach to Software Component Specification. *Proceedings of 1999 International Workshop on Component Based Software Engineering*, Los Angeles, USA, May 1999.

[63] Hancock, J. (2000) Application Frameworks before system frameworks. *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 2000.

[64] Harrison, N., Foote, B. and Rohnert, H. (1997). Pattern Language of Program Design 4. *Addison Wesley Publishing Company*, 1st Edition (December 17, 1999).

[65] Heineman, G. T. and Councill, W. T. (2001). Component-Based Software Engineering Putting the Pieces Together., *Addison Wesley Longman*, Inc., California, Menlo Park, USA.

[66] Heineman, G. T., Councill, T., et al. (2000). Component-Based Software Engineering and the Issue of Trust. *Proceedings of the 22nd International Conference on Software Engineering (ICSE)*, Canada, 2000, Pages. 661-664.

[67] Heineman, T. and Councill, T. (2001). Component Technology and QoS Management. *Proceedings of the 7th Component-Based Software Engineering (CBSE'04).*

[68] Hissam, S., Moreno, G., Stafford, J. and Wallnau, K. (2003). Enabling Predictable Assembly. *Journal of Systems and Software*, Volume 65, No. 03, 2003, Pages 185-198.

[69] Hölzle, U. (1993). Integrating Independently-Developed Components in Object-Oriented Languages. *Proceedings of the 7th European Conference on Object-Oriented Programming (ECOOP '93)*, July 26-30 1993, Pages 36-56.

[70] Horrocks, I. (1998). The FaCT System. *Proceedings of the 2nd International Conference on*

Analytic Tableaux and Related Methods (TABLEAUX'98), Volume 1397 of *Lecture Notes in Artificial Intelligence*, Pages 307-312. Springer.

[71] Horrocks, I. and Sattler, U. (2002). Description Logics Basics, Applications, and More. *Tutorial at ECAI-2002*, http://www.cs.man.ac.uk/horrocks/Slides/ecai-handout.pdf; 2002.

[72] http://dl.kr.org/

[73] http://edge.cs.drexel.edu/assemblies/software/owljesskb/

[74] http://en.wikipedia.org/wiki/Android_(operating_system)

[75] http://en.wikipedia.org/wiki/BlackBerry_OS

[76] http://en.wikipedia.org/wiki/IOS_(Apple)

[77] http://en.wikipedia.org/wiki/Symbian_OS

[78] http://en.wikipedia.org/wiki/Windows_Phone_7

[79] http://fowl.sourceforge.net

[80] http://jena.sourceforge.net

[81] http://logic.stanford.edu/kif/Hypertext/kif-manual.html.

[82] http://merobase.com/?cid=2899

[83] http://owl.man.ac.uk/factplusplus/

[84] http://protegewiki.stanford.edu/wiki/Protege

[85] http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library

[86] http://softwarereuse.nasa.gov

[87] http://sweetrules.projects.semwebcentral.org

[88] http://swoogle.umbc.edu

[89] http://triple.semanticweb.org/

[90] http://www.cstc.org.cn/

[91] http://www.daml.org/ontologies/

[92] http://www.gameontology.org/

[93] http://www.gameloft.com

[94] http://www.kingdee.com/en/index.jsp

[95] http://www.ksl.stanford.edu/software/JTP/

[96] http://www.mindswap.org/2003/pellet

[97] http://www.myally.com.cn/index.aspx

[98] http://www.racer.org.com

[99] http://www.schemaweb.info

[100]http://www.sei.cmu.edu/cbs/

[101]http://www.semwebcentral.org/

[102]http://www.sts.tu-harburg.de/r.f.moeller/racer/

[103]http://www.ubi.com/UK/default.aspx/

[104]http://www.w3.org/2003/08/owl-systems/test-results-out/

[105]http://www.w3.org/2004/ontaria/

[106]http://www.w3.org/TR/owl-features/

[107]http://www.w3.org/TR/rdf-schema/

[108]http://www.wiwiss.fu-berlin.de/suhl/bizer/toolkits/

[109]Immonen, A. (2008). Antti Validation of the Reliability Analysis Method and Tool. *Proceedings of the 12th International Software Product Line Conference*, Second Volume, Pages 163 – 168, 2008.

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

[111]Jacobsen, I., Griss, M., et al. (1997), Software Reuse, Reading, MA: Addison-Wesley.

[112]Jensen, K. (1997). Coloured Petri Nets, Basic Concepts, Analysis methods, and Practical. *Second Edition, Springer-Verlag*, Volume 1, 1997.

[113]Jhumka, A., Hiller, M., and Suri, N. (2002). Component-based Synthesis of Dependable Embedded Software, Formal Techniques in Real-Time and Fault-Tolerant Systems. *Proceedings of the 7th International Conference on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT 2002)*, Pages 111–128, 2002.

[114]Kalyanpur, A., Parsia, B. and Hendler, J. (2005). A Tool for Working with Web Ontologies. *International Journal on Semantic Web and Information Systems*, Volume 1, 2005.

[115]Kanai, A., Furuyama, T. and Takahashi, M. (1992). A Cost Model for Software Conversion Based on Program Characteristics and A Converter Effect. *Proceedings of International Computer Sojiwanz and Application Conference*, Pages 63-68.1992.

[116]Keller, R. and Hölzle, U. (1998). Binary Component Adaptation. *Proceedings of the 12th European Conference on Object-Oriented Programming*, July 1998.

[117]Kifer, M., Lausen, G. and Wu, J. (1995). Logical Foundations of Object-oriented and Frame-based Languages. *Journal of the ACM*, 1995.

[118]Kim, D. K., Ghosh, S., France, R. B. and Song, E. (2002). Software Component Specification Using Role-Based Modelling Language. *Proceedings of the 11th OOPSLA Workshop on Behavioral Semantics: Serving the Customer*, November 4, 2002.

[119]Kim, R. and Edward, A. (1998). Software Reuse: Survey and Research Directions. *Journal of Management Information Systems*, Volume 14, No. 4, Springer 1998, Pages 113 – 149.

[120]Kopena, J. and Regli, W. (2003). *DAMLJessKB:* A Tool for Reasoning with the Semantic Web, *IEEE Intelligent Systems*, 2003, 18(3):74–77.

[121]Laddad, R. (2003). AspectJ in Action, Practical Aspect-Oriented Programming. *Manning Publications Co*.

[122]Larsson, M. (2000). Applying Configuration Management Techniques to Component-Based Systems. *Licentiate Thesis*, *Dissertation 2000-2007*, Department of Information Technology Uppsala University. 2000.

[123]Larsson, M. (2004). Predicting Quality Attributes in Component-based Software Systems, *Ph D Thesis*, Mälardalen University.

[124]Lassila, O. and Swick, R. R. (1999). Resource Description Framework (rdf) Model and Syntax Speciation. *w3c recommendation 22, February 19. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/*, 1999.

[125]Lenat, D. and Guha, R. (1990). Building Large Knowledge-based Systems, Representation and Inference in the Cyc Project. 1990, *Addison-Wesley*, Reading, Massachusetts.

[126]Liu, Q., Jin, X. and Long, Y. (2007). Research on Ontology-based Representation and Retrieval of Components, *Proceedings of the 8th ACIS International Conference*, Volume 1, Pages 494 – 499, 2007.

[127]Liu, X., Wang, B. and Kerridge, J. (2005). Achieving Seamless Component Composition through Scenario-Based Deep Adaptation and Generation. *Journal of Science of Computer Programming (Elsevier), Special Issue on New Software Composition Concepts*, 56, 2.

[128]Liu, Y. and Cunningham, H. C. (2002). Software Component Specification Using Design by Contract. *Proceedings of the SouthEast Software Engineering Conference, Tennessee Valley Chapter, National Defense Industry Association*. Huntsville*,* AL, April 2002.

[129]Lorenz, D. H. and Vlissides, J. (2003). Pluggable Reflection: Decoupling Meta-interface and Implementation, *Proceedings of the 25th International Conference on Software Engineering (ICSE 2003)*, Pages 3-13, Portland, Oregon, May 1-10 2003, IEEE Computer Society.

[130]MacGregor, R. (1991). Using a description classier to Enhance Deductive Inference. *Proceedings of the 7th IEEE Conference on AI Application*, Florida, Pages. 93-97, 1991.

[131]Martin, R. C., Riehle, D. and Buschmann, F. (1997). Pattern Language of Program Design 3. *Addison-Wesley Professional*, 1st edition (October 7, 1997).

[132]Matzel, K. U. and Schnorf, P. (1997). Dynamic Component Adaptation. *Technical Report* 97-6-1, Union Bank of Swizerland, June 1997.

[133]McGregor, J., Stafford, J. and Cho, I. (2003). Measuring Component Reliability, *Proceedings of the 6th Workshop on Component-Based Software Engineering (CBSE)*, USA, 2003, Pages 13-24.

[134]Meyer, B. (1997). Object-Oriented Software Construction, *2nd Edition, Prentice Hall*, London, 1997.

[135]Mili, A., Mili, R. and Mittermeir, R. (1997). Storing and Retrieving Software Components: A Refinement-Based System, *Proceedings of the IEEE Transactions on Software Engineering*, 1997, Volume 23, No. 7, Pages 445 – 460.

[136]Mili, A., Mili, R. and Mittermeir, T. (1998). A survey of Software Reuse Libraries, *Annals of Software Engineering*, 1998, Pages 349-414.

[137]Mili, H., Mili, A., Yacoub, S. and Addy, E. (2002). Reuse-Based Software Engineering, Techniques, Organization, and Controls. *Wiley Inter-Science*, ISBN: 0-471-39819-5.

[138]Morel, B. and Alexander, P. (2002). A Slicing Approach for Parallel Component Adaptation. *Technical Report*, the University of Kansas Center for Research.

[139]Morris, J., Lee, G., Parker, K., Bundell, G. and Lam, C. (2001). Software Component Certification. *IEEE Computer*, Volume 34, No 09, 2001.

[140]Motta, E. (1998). An Overview of the Ocml Modeling Language. *Knowledge Engineering Methods and Languages*, 1998.

[141]Mylopoulos, J., Borgida, A., Jarke, M. and Koubarakis, M. (1990). Telos: Representing Knowledge about Information Systems. *Information Systems*, 8(4):325-362, 1990.

[142]Oman, P. and Hagemeister, J. (1992). Metrics for Assessing a Software System's Maintainability. *Proceedings of Conference on Software Maintenance*, *IEEE*, 1992.

[143]Ostertag, E., Hendler, J., Prieto-Diaz, R., and Braum, C. (1992). Computing Similarity in A Reuse Library System: An AI-based Approach. *ACM Transactions on Software Engineering and Methodology*, 1992, Volume 1, No. 3, Pages 205 – 228.

[144]Pahl, C. (2007). An Ontology for Software Component Matching. *International Software Tools Technology Transfer*, 2007, Pages 169–178.

[145]Parsia, B., Sirin, E. and Kalyanpu, A. (2005). Debugging OWL Ontologies. *Proceedings of the 14th International World Wide Web Conference (WWW-05)*, 2005.

[146]Patrizio, A. (2000). The New Developer Portals. I*nformation Week*, No. 799, Aug 2000, Pages 81-86.

[147]Penix, J. (1997). Automated Component Retrieval and Adaptation Using Formal Specifications. *Proceeding in the 17th International Conference on Software Engineering*, Doctoral Consortium, May 1997, Boston, Massachusetts.

[148]Penix, J. and Alexander, P. (1995). Design Representation for Automating Software Component Reuse. *Proceedings of the 1st international workshop on Knowledge-Based systems for (re)Use of Program libraries*, November 23-24, 1995 Sophia Antipolis France.

[149]Pentti, T. (2007). Adaptability Evaluation of Software Architectures; A Case Study. *Proceedings of the 31st Annual International Computer Software and Applications Conference*, Volume 2, Pages 579-586, 2007.

[150]Pfarr, T. and Reis, J. E. (2002). The integration of COTS/GOTS within NASA's HST Command and Control System. *Proceedings of 1st International Conference on COTS-based Software Systems*, Orlando, FL: Springer-Verlag.

[151]Poore, J., Mills, H. and Mutchler, D. (1993). Planning and Certifying Software System Reliability. *IEEE Computer*, Volume 10, No. 01, 1993, Pages 88-99.

[152]Presso, M. J. (2000). Declarative Descriptions of Component Models as a Generic Support for Software Composition, *Proceedings of the 5th International Workshop on Component-Oriented Programming*, Blekinge Institute of Technology.

[153]Prieto-Diaz, R. and Freeman, P. (1987). Classifying Software for Reuse, *IEEE Software*, 1987, Volume 4(1), Pages 6–16.

[154]Rainsford, C. (2001). Towards Automated Software Component Assembly Systems. *Evolve Conference*, 2001.

[155]Riehle, A. and Züllighoven, H. (1996). Understanding and Using Patterns in Software Development. *Theory and Practice of Object Systems,* Volume 2 (1), Pages 3-13, 1996.

[156]Rierson, L. (2000). Software Reuse in Safety-Critical Systems, *Master's Thesis*, Rochester Institute of Technology, May 2000.

[157]Rohde, L., Dyson, K., Geriner, P. and Cerino, D. (1996). Certification of Reusable Software Components: Summary of Work in Progress. *Proceedings of the 2nd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, Canada, 1996, Pages 120-123.

[158]Samentinger, J. (1997). Software Engineering with Reusable Components. *Springer Verlag*.

[159]Schmidt, D.C. (1999). Why Software Reuse has Failed and How to Make It Work for You. *C++ Report*, 11 (1), 1999, 46-52.

[160]Schreibe, B., Wielinga, J., Akkermans, H., Van de Velde, W. and Anjewierden, A. Cml:the Commonkads Conceptual Modeling Language. *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW'94)*, Pages 283-300, 1994.

[161]Seacord, R., Hissam, S. and Wallnau, K. (1998). Agora: A Search Engine for Software Components. *Technical Report*, CMU/SEI - Carnegie Mellon University/Software Engineering Institute, 1998.

[162]Serra, I. (2004). A Technique for the Development of Domain Specific Languages. *Master Degree Dissertation*, Federal University of Maranhão – CPGEE, 2004.

[163]Shadbolt, N., Motta, E and Rouge, A. (1993). Constructing Knowledge Based Systems, *IEEE Software*, 10(6):34-38, 1993.

[164]Shukla, D., Fell, S. and Sells, C. (2002). Aspect-Oriented Programming Enables Better Code Encapsulation and Reuse, *MSDN Magazine*.

[165]Sintek, M. and Decker, S. (2002). TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. *Proceedings of the 1st International Semantic Web Conference (ISWC-02)*, Springer-Verlag, 2002, Pages 364–378.

[166]Sirin, E.and Parsia, B. (2004). Pellet: An OWL DL Reasoner. *Description Logics*, 2004.

[167]Sommerville, I. (2007). Software Engineering, 8th Edition. *Addison-Wesley*, ISBN: 978-0-321-31379-9.

[168]Speck, A., Pulvermuller, F. and Mezini, M. (2000). Reusability of Concerns. *Proceedings of the Aspects and Dimensions of Concerns Workshop (ECOOP2000).* France, June 2000.

[169]Stafford, J. and Wallnau, K. (2001). Is Third Party Certification Necessary?. *Proceedings of the 4th Workshop on Component-Based Software Engineering (CBSE)*, Canada, May, 2001.

[170]Stojanovic, N. (2005). On the Query Refinement in the Ontology-based Searching for Information. *Information Systems*, Volume 30, Issue 7, November 2005, Pages 543-563.

[171]Sugumaran, V.and Storey, V. (2003). A Semantic-Based Approach to Component Retrieval. *The Database for Advances in Information Systems*, Volume 34(3) (2003).

[172]Swe, S. M., Zhang, H. and Jarzabek, S. (2002), XVCL: A Tutorial. *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, Volume 27, Pages 341-349, Ischia, Italy.

[173]Torchiano, M. and Morisio, M. (2004), Overlooked Facts on COTS-based Development. *IEEE Software*, Volume 21, Pages 88-93.

[174]Traas, V. and Hillegersberg, J. (2000). The Software Component Market on the Internet Current Status and Conditions for Growth. *Software Engineering Notes*, Volume 25, No. 1. 2000.

[175]Tracz, W. (1995). Confessions of a Used Program Salesman: Institutionalizing Software Reuse. *Addison-Wesley*, Reading, MA, 1995.

[176]Tracz, W. (2001). COTS Myths and Other Lessons Learned in Component-based Software Development. *Component-Based Software Engineering*, *Addison-Wesley*, Pages 99-112.

[177]Tsarkov, D., Horrocks, Ian. (2003). Implementing New Reasoner with Data types support. *Wonder Web Ontology Infrastructure for the Semantic Web Deliverable*, 2003.

[178]Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 11(2), 1996.

[179]Vlissides, M., Coplien, J. O., Kerth, N.L. and Coplien, J. (1996). Pattern Language of Program Design 2. 1st edition (June 14, 1996), *Addison-Wesley Professional.*

[180]Voas, J. (1998). Certifying Off-the-Shelf Software Components. *IEEE Computer*, Volume 31, No. 06, 1998, Pages 53-59.

[181]Voas, J., Payne, J. (2000). Dependability Certification of Software Components. *Journal of Systems and Software*, Volume 52, No. 2-3, 2000, Pages 165-172.

[182]Walker, R. J. and Clarke, S. (2001). Composition patterns: An Approach to Designing Reusable Aspects. *Proceedings of 23rd International Conference on Software Engineering (Toronto, Ontario, Canada; 12--19 May)*, Pages 5--14, 2001.

[183]Wallnau, C. (2003). Volume III: A Technology for Predictable Assembly from Certifiable Components. *Software Engineering Institute (SEI)*, *Technical Report*, Volume 03, April, 2003.

[184]Wang, B., Liu, X., and Kerridge, J. (November, 2004). Scenario-based Generative Component Adaptation in .NET Framework. *Proceedings of the IEEE International Conference on Information Reuse and Integration*, Las Vegas, USA.

[185]Warmer, J. B. and Kleppe, A. G. (1998). The Object Constraint Language: Precise Modeling with Uml. *Addison Wesley Publishing Company*, ISBN: 0201379406, October 1998.

[186]Welty, C. and Guarino, N. (2001). Supporting Ontological Analysis of Taxonomic Relationships. *Data & Knowledge Engineering*, 39:51-74, 2001.

[187]Wirfs-Brock, R.J. and Johnson, R.E. (1990). Surveying Current Research in Object-oriented Design. *Communications of ACM*, Volume 33(9), Pages 104-124.

[188]Wohlin, C. and Runeson, P. (1994). Certification of Software Components. *IEEE Transactions on Software Engineering*, Volume 20, No. 06, 1994, Pages 494-499.

[189]Wolberg, J. R. (1981). Comparing the Cost of Software Conversion to the Cost of Reprogramming. *SIGPLAN Notices*, 16(4), 104-l 10.

[190]Yao, H. and Letha, E. (2004). Towards A Semantic-based Approach for Software Reusable Component Classification and Retrieval. *ACM Southeast Conference*, 2004.

[191]Yen, I., Goluguri, J. et. al. (2002). A Component-based Approach for Embedded Software Development. *Proceedings of the 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'02)*, Pages 0402 (2002).

[192]Zaremski, M. and Wing, M. (1993). Signature Matching: A Key to Reuse. *Software*

Engineering Notes, 1993, Volume 18, No. 5, Pages 182–190.

[193]Zaremski, M. and Wing, M. (1995). Specification Matching of Software Components. *Software Engineering Notes*, 1995, Volume 20, No. 4, Pages 6–17.

[194]Zhuge, H. (2000). A Problem-oriented and Rule-based Component Repository. *The Journal of Systems and Software*, 50:201– 208, 2000.

[195]Zou, Y., Finin, T., and Chen, H. (2004). F-OWL: an Inference Engine for the Semantic Web. *Formal Approaches to Agent-Based Systems*, Volume 3228 of *Lecture Notes in Computer Science, Springer-verlag*, 2004.

Appendix A: The OWL Doc of the MVICS Model

A.1 Function Model

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
      <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
      <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
      <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
      <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
      <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
      <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
      <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
      <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
      <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
      <!ENTITY Ontology1228882645 "http://www.owl-ontologies.com/Ontology1228882645.owl#" >
      <!ENTITY Time_ "http://www.owl-ontologies.com/Ontology1228882645.owl#Time_/" >
      <!ENTITY Calendar_ "http://www.owl-ontologies.com/Ontology1228882645.owl#Calendar_/" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1228882645.owl#"
       xml:base="http://www.owl-ontologies.com/Ontology1228882645.owl"
       xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
       xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
       xmlns:Calendar_="&Ontology1228882645;Calendar_/"
       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
       xmlns:swrl="http://www.w3.org/2003/11/swrl#"
       xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
       xmlns:owl="http://www.w3.org/2002/07/owl#"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
       xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:Ontology1228882645="http://www.owl-ontologies.com/Ontology1228882645.owl#"
       xmlns:Time_="&Ontology1228882645;Time_/">
      <owl:Ontology rdf:about="http://www.owl-ontologies.com/Ontology1228882645.owl"/>
      <!--
      ///////////////////////////////////////////////////////////////////////////////////////
      //
      // Object Properties
      //
      ///////////////////////////////////////////////////////////////////////////////////////
       -->
      <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#hasApplicationDomain -->
      <owl:ObjectProperty rdf:about="&Ontology1228882645;hasApplicationDomain">
            <rdfs:domain rdf:resource="&Ontology1228882645;Component_Application_Domain"/>
            <owl:inverseOf rdf:resource="&Ontology1228882645;isApplicationDomainOf"/>
      </owl:ObjectProperty>
      <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#hasFunctionType -->
      <owl:ObjectProperty rdf:about="&Ontology1228882645;hasFunctionType">
            <rdfs:domain rdf:resource="&Ontology1228882645;Component_Function_Type"/>
            <rdfs:range>
                  <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
```

```
                                <rdf:Description rdf:about="&Ontology1228882645;Component_Barcode"/>
                                <rdf:Description
rdf:about="&Ontology1228882645;Component_Calendar_and_Schedule"/>
                                <rdf:Description rdf:about="&Ontology1228882645;Component_Data_Processing"/>
                                <rdf:Description rdf:about="&Ontology1228882645;Component_Encryption"/>
                                <rdf:Description rdf:about="&Ontology1228882645;Component_File_Processing"/>
                                <rdf:Description rdf:about="&Ontology1228882645;Component_Image_Processing"/>
                                <rdf:Description rdf:about="&Ontology1228882645;Component_Reporting"/>
                                <rdf:Description
rdf:about="&Ontology1228882645;Component_Text_and_Word_Processing"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#isApplicationDomainOf -->
    <owl:ObjectProperty rdf:about="&Ontology1228882645;isApplicationDomainOf">
        <rdfs:range rdf:resource="&Ontology1228882645;Component_Application_Domain"/>
    </owl:ObjectProperty>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#isFunctionTypeOf -->
    <owl:ObjectProperty rdf:about="&Ontology1228882645;isFunctionTypeOf">
        <rdfs:range rdf:resource="&Ontology1228882645;Component_Function_Type"/>
        <owl:inverseOf rdf:resource="&Ontology1228882645;hasFunctionType"/>
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="&Ontology1228882645;Component_Barcode"/>
                    <rdf:Description
rdf:about="&Ontology1228882645;Component_Calendar_and_Schedule"/>
                    <rdf:Description rdf:about="&Ontology1228882645;Component_Data_Processing"/>
                    <rdf:Description rdf:about="&Ontology1228882645;Component_Encryption"/>
                    <rdf:Description rdf:about="&Ontology1228882645;Component_File_Processing"/>
                    <rdf:Description rdf:about="&Ontology1228882645;Component_Image_Processing"/>
                    <rdf:Description rdf:about="&Ontology1228882645;Component_Reporting"/>
                    <rdf:Description
rdf:about="&Ontology1228882645;Component_Text_and_Word_Processing"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>
    <!--        ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //      ///////////////////////////////////////////////////////////////////////////////////////
      -->
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Account -->
    <owl:Class rdf:about="&Ontology1228882645;Account">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Adaptation_Assets -->
    <owl:Class rdf:about="&Ontology1228882645;Adaptation_Assets">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Maintainability"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Adaptation_Methods -->
    <owl:Class rdf:about="&Ontology1228882645;Adaptation_Methods">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Maintainability"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Aggregation_(Financial) -->
    <owl:Class rdf:about="&Ontology1228882645;Aggregation_(Financial)">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Financial"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Aggregation_(Game) -->
    <owl:Class rdf:about="&Ontology1228882645;Aggregation_(Game)">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Game"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Analytics -->
    <owl:Class rdf:about="&Ontology1228882645;Analytics">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Association_Class_(Financial) -->
```

```
<owl:Class rdf:about="&Ontology1228882645;Association_Class_(Financial)">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Financial"/>
</owl:Class>
        <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Association_Class_(Game) -->
<owl:Class rdf:about="&Ontology1228882645;Association_Class_(Game)">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Game"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Calculation -->
<owl:Class rdf:about="&Ontology1228882645;Calculation">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Calendar_/_Schedule -->
<owl:Class rdf:about="&Ontology1228882645;Calendar_/_Schedule">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Card_Authorisation -->
<owl:Class rdf:about="&Ontology1228882645;Card_Authorisation">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Charting_and_Graphing -->
<owl:Class rdf:about="&Ontology1228882645;Charting_and_Graphing">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Communication -->
<owl:Class rdf:about="&Ontology1228882645;Communication">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Compiler -->
<owl:Class rdf:about="&Ontology1228882645;Compiler">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Portability"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Account -->
<owl:Class rdf:about="&Ontology1228882645;Component_Account">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Administration -->
<owl:Class rdf:about="&Ontology1228882645;Component_Administration">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Analytics -->
<owl:Class rdf:about="&Ontology1228882645;Component_Analytics">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Application_Domain -->
<owl:Class rdf:about="&Ontology1228882645;Component_Application_Domain"/>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Barcode -->
<owl:Class rdf:about="&Ontology1228882645;Component_Barcode">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Calculation -->
<owl:Class rdf:about="&Ontology1228882645;Component_Calculation">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Calendar_and_Schedule -->
<owl:Class rdf:about="&Ontology1228882645;Component_Calendar_and_Schedule">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Category -->
<owl:Class rdf:about="&Ontology1228882645;Component_Category">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Charting_and_Graphing -->
<owl:Class rdf:about="&Ontology1228882645;Component_Charting_and_Graphing">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Data_Cleaning -->
<owl:Class rdf:about="&Ontology1228882645;Component_Data_Cleaning">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Data_Processing"/>
</owl:Class>
```

```
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Data_Conversion -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Data_Conversion">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Data_Processing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Data_Entry -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Data_Entry">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Data_Processing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Data_Processing -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Data_Processing">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Data_Security -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Data_Security">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Data_Processing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Data_Transfer -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Data_Transfer">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Data_Processing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Data_Validation -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Data_Validation">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Data_Processing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Data_Verification -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Data_Verification">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Data_Processing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Database_Management -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Database_Management">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Education -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Education">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Application_Domain"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Email -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Email">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Encryption -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Encryption">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_File_Handling -->
    <owl:Class rdf:about="&Ontology1228882645;Component_File_Handling">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_File_Processing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_File_Processing -->
    <owl:Class rdf:about="&Ontology1228882645;Component_File_Processing">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_File_Transfer -->
    <owl:Class rdf:about="&Ontology1228882645;Component_File_Transfer">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_File_Processing"/>
    </owl:Class>
    <!--     http://www.owl-ontologies.com/Ontology1228882645.owl#Component_File_Upload_and_Download
-->
    <owl:Class rdf:about="&Ontology1228882645;Component_File_Upload_and_Download">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_File_Processing"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Financial -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Financial">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Application_Domain"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Function_Type -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Function_Type"/>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Game -->
    <owl:Class rdf:about="&Ontology1228882645;Component_Game">
```

```
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Application_Domain"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Image_Compression -->
<owl:Class rdf:about="&Ontology1228882645;Component_Image_Compression">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Image_Processing"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Image_Conversion -->
<owl:Class rdf:about="&Ontology1228882645;Component_Image_Conversion">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Image_Processing"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Image_Processing -->
<owl:Class rdf:about="&Ontology1228882645;Component_Image_Processing">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Imaging -->
<owl:Class rdf:about="&Ontology1228882645;Component_Imaging">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Image_Processing"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_ Integratability -->
<owl:Class rdf:about="&Ontology1228882645;Component_ Integratability ">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Quality_Attributes"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Interface -->
<owl:Class rdf:about="&Ontology1228882645;Component_Interface"/>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Maintainability -->
<owl:Class rdf:about="&Ontology1228882645;Component_Maintainability">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Quality_Attributes"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Model -->
<owl:Class rdf:about="&Ontology1228882645;Component_Model">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_ Integratability "/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Modeling -->
<owl:Class rdf:about="&Ontology1228882645;Component_Modeling">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Monitoring -->
<owl:Class rdf:about="&Ontology1228882645;Component_Monitoring">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Navigation -->
<owl:Class rdf:about="&Ontology1228882645;Component_Navigation">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_PDF -->
<owl:Class rdf:about="&Ontology1228882645;Component_PDF">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Planning -->
<owl:Class rdf:about="&Ontology1228882645;Component_Planning">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Portability -->
<owl:Class rdf:about="&Ontology1228882645;Component_Portability">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Quality_Attributes"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Portfolio_Management -->
<owl:Class rdf:about="&Ontology1228882645;Component_Portfolio_Management">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Post-conditions -->
<owl:Class rdf:about="&Ontology1228882645;Component_Post-conditions">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Interface"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Pre-conditions -->
<owl:Class rdf:about="&Ontology1228882645;Component_Pre-conditions">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Interface"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Presentation -->
```

```xml
<owl:Class rdf:about="&Ontology1228882645;Component_Presentation">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Quality_Attributes -->
<owl:Class rdf:about="&Ontology1228882645;Component_Quality_Attributes"/>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Reporting -->
<owl:Class rdf:about="&Ontology1228882645;Component_Reporting">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Search -->
<owl:Class rdf:about="&Ontology1228882645;Component_Search">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_System_Administration -->
<owl:Class rdf:about="&Ontology1228882645;Component_System_Administration">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Administration"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Text_and_Word_Processing -->
<owl:Class rdf:about="&Ontology1228882645;Component_Text_and_Word_Processing">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Time_and_Date -->
<owl:Class rdf:about="&Ontology1228882645;Component_Time_and_Date">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Toolbar_and_Menu -->
<owl:Class rdf:about="&Ontology1228882645;Component_Toolbar_and_Menu">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Tracking -->
<owl:Class rdf:about="&Ontology1228882645;Component_Tracking">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Type -->
<owl:Class rdf:about="&Ontology1228882645;Component_Type">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_ Integratability "/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_User_Administration -->
<owl:Class rdf:about="&Ontology1228882645;Component_User_Administration">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Administration"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Version_Control -->
<owl:Class rdf:about="&Ontology1228882645;Component_Version_Control">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Component_Workflow -->
<owl:Class rdf:about="&Ontology1228882645;Component_Workflow">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Function_Type"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Constraint -->
<owl:Class rdf:about="&Ontology1228882645;Constraint">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Pre-conditions_of_method_n"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Container -->
<owl:Class rdf:about="&Ontology1228882645;Container">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Portability"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Dashboard -->
<owl:Class rdf:about="&Ontology1228882645;Dashboard">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Data_Management -->
<owl:Class rdf:about="&Ontology1228882645;Data_Management">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Data_Transfer -->
<owl:Class rdf:about="&Ontology1228882645;Data_Transfer">
	<rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
```

```
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Database -->
<owl:Class rdf:about="&Ontology1228882645;Database">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Portability"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Database_Management_1 -->
<owl:Class rdf:about="&Ontology1228882645;Database_Management_1">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Database_Management_2 -->
<owl:Class rdf:about="&Ontology1228882645;Database_Management_2">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Database_Management_3 -->
<owl:Class rdf:about="&Ontology1228882645;Database_Management_3">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Database_Security_1 -->
<owl:Class rdf:about="&Ontology1228882645;Database_Security_1">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Database_Security_2 -->
<owl:Class rdf:about="&Ontology1228882645;Database_Security_2">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Distributions_Processing_1 -->
<owl:Class rdf:about="&Ontology1228882645;Distributions_Processing_1">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Distributions_Processing_2 -->
<owl:Class rdf:about="&Ontology1228882645;Distributions_Processing_2">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Document_Management -->
<owl:Class rdf:about="&Ontology1228882645;Document_Management">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Email -->
<owl:Class rdf:about="&Ontology1228882645;Email">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Management_1 -->
<owl:Class rdf:about="&Ontology1228882645;Management_1">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Management_2 -->
<owl:Class rdf:about="&Ontology1228882645;Management_2">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Marketing -->
<owl:Class rdf:about="&Ontology1228882645;Marketing">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Modelling -->
<owl:Class rdf:about="&Ontology1228882645;Modelling">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Monitoring -->
<owl:Class rdf:about="&Ontology1228882645;Monitoring">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Navigation -->
<owl:Class rdf:about="&Ontology1228882645;Navigation">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Navigation_1 -->
<owl:Class rdf:about="&Ontology1228882645;Navigation_1">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Notifications_and_Approvals -->
```

```
<owl:Class rdf:about="&Ontology1228882645;Notifications_and_Approvals">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Operating_System -->
<owl:Class rdf:about="&Ontology1228882645;Operating_System">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Portability"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#PDF_Creation -->
<owl:Class rdf:about="&Ontology1228882645;PDF_Creation">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Parameter_Range -->
<owl:Class rdf:about="&Ontology1228882645;Parameter_Range">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Pre-conditions_of_method_n"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Parameter_Type -->
<owl:Class rdf:about="&Ontology1228882645;Parameter_Type">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Pre-conditions_of_method_n"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Payment_System_1 -->
<owl:Class rdf:about="&Ontology1228882645;Payment_System_1">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Payment_System_2 -->
<owl:Class rdf:about="&Ontology1228882645;Payment_System_2">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Payment_System_3 -->
<owl:Class rdf:about="&Ontology1228882645;Payment_System_3">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Planning -->
<owl:Class rdf:about="&Ontology1228882645;Planning">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Portfolio_Management_1 -->
<owl:Class rdf:about="&Ontology1228882645;Portfolio_Management_1">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Pre-conditions_of_method_n -->
<owl:Class rdf:about="&Ontology1228882645;Pre-conditions_of_method_n">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Component_Pre-conditions"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Profile -->
<owl:Class rdf:about="&Ontology1228882645;Profile">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Reporting -->
<owl:Class rdf:about="&Ontology1228882645;Reporting">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Search -->
<owl:Class rdf:about="&Ontology1228882645;Search">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Settlement -->
<owl:Class rdf:about="&Ontology1228882645;Settlement">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Statement -->
<owl:Class rdf:about="&Ontology1228882645;Statement">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Switching -->
<owl:Class rdf:about="&Ontology1228882645;Switching">
    <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Text_and_Word_Processing -->
<owl:Class rdf:about="&Ontology1228882645;Text_and_Word_Processing">
```

```
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Time_/_Date -->
    <owl:Class rdf:about="&Ontology1228882645;Time_/_Date">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Tracking -->
    <owl:Class rdf:about="&Ontology1228882645;Tracking">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Transfer -->
    <owl:Class rdf:about="&Ontology1228882645;Transfer">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#User_Administration -->
    <owl:Class rdf:about="&Ontology1228882645;User_Administration">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#User_Administration_1 -->
    <owl:Class rdf:about="&Ontology1228882645;User_Administration_1">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Aggregation_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Version_Control -->
    <owl:Class rdf:about="&Ontology1228882645;Version_Control">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Workflow -->
    <owl:Class rdf:about="&Ontology1228882645;Workflow">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
    </owl:Class>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Workflow_Management -->
    <owl:Class rdf:about="&Ontology1228882645;Workflow_Management">
        <rdfs:subClassOf rdf:resource="&Ontology1228882645;Association_Class_(Financial)"/>
    </owl:Class>
    <!-- http://www.w3.org/2002/07/owl#Thing -->
    <owl:Class rdf:about="&owl;Thing"/>
    <!--    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //    ///////////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#ImagXpress_View -->
    <owl:Thing rdf:about="&Ontology1228882645;ImagXpress_View">
        <rdf:type rdf:resource="&Ontology1228882645;Component_Imaging"/>
        <rdf:type rdf:resource="&owl;NamedIndividual"/>
    </owl:Thing>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Netrix_DOM_Handing -->
    <owl:Thing rdf:about="&Ontology1228882645;Netrix_DOM_Handing">
        <rdf:type rdf:resource="&Ontology1228882645;Component_File_Handling"/>
        <rdf:type rdf:resource="&owl;NamedIndividual"/>
    </owl:Thing>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#PowerTCP_FTP_for_.NET -->
    <owl:Thing rdf:about="&Ontology1228882645;PowerTCP_FTP_for_.NET">
        <rdf:type rdf:resource="&Ontology1228882645;Component_File_Transfer"/>
        <rdf:type rdf:resource="&owl;NamedIndividual"/>
    </owl:Thing>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Rainbow_PDF_Server_Based_Converter -->
    <owl:Thing rdf:about="&Ontology1228882645;Rainbow_PDF_Server_Based_Converter">
        <rdf:type rdf:resource="&Ontology1228882645;Component_File_Handling"/>
        <rdf:type rdf:resource="&owl;NamedIndividual"/>
    </owl:Thing>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#Xceed_Ultimate_Suite_2008 -->
    <owl:Thing rdf:about="&Ontology1228882645;Xceed_Ultimate_Suite_2008">
        <rdf:type rdf:resource="&Ontology1228882645;Component_File_Handling"/>
        <rdf:type rdf:resource="&owl;NamedIndividual"/>
    </owl:Thing>
    <!-- http://www.owl-ontologies.com/Ontology1228882645.owl#ezCrypto_.NET -->
    <owl:Thing rdf:about="&Ontology1228882645;ezCrypto_.NET">
        <rdf:type rdf:resource="&Ontology1228882645;Component_Encryption"/>
        <rdf:type rdf:resource="&owl;NamedIndividual"/>
    </owl:Thing>
</rdf:RDF>
<!-- Generated by the OWL API (version 3.2.3.22702) http://owlapi.sourceforge.net -->
```

A.2 Context Model

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    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/Ontology1226373901.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1226373901.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Component_Oracle_JDeveloper">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Component_Oracle"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_Tuxedo"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_WebLogic_Workshop"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_WebLogic_Express"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_WebLogic_Portal"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_WebLogic_Server"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_Oracle_Database"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_OSGI_Model">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Component_Model"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_.NET_Model"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_CCM_Model"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_COM_Model"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_JavaBean_Model"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_Visual_Cplusplus_2005">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Component_Microsoft"/>
    </rdfs:subClassOf>
  </owl:Class>
```

```
<owl:Class rdf:ID="Component_Visual_Cplusplus_2008">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_JavaBean_Model">
  <owl:disjointWith rdf:resource="#Component_OSGI_Model"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_.NET_Model"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_CCM_Model"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_COM_Model"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Model"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Component_Visual_FoxPro">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Component_SharePoint">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Component_MySQL">
  <owl:disjointWith>
    <owl:Class rdf:ID="Component_Sybase"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Component_Sun"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Oracle"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Component_IBM"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Component_Eclipse"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Component_CodeGear"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Component_Platform"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Component_Windows">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Component_OS"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_CodeGear">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Platform"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Sybase"/>
  </owl:disjointWith>
  <owl:disjointWith>
```

```
      <owl:Class rdf:about="#Component_Sun"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Oracle"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_MySQL"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_IBM"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Eclipse"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Microsoft"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Component_Tuxedo">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_Oracle"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WebLogic_Workshop"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WebLogic_Express"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WebLogic_Portal"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WebLogic_Server"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Oracle_JDeveloper"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Oracle_Database"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_Unix">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_OS"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Component_Windows_9X">
    <rdfs:subClassOf rdf:resource="#Component_Windows"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_Eclipse">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_Platform"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Sybase"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Sun"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Oracle"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_MySQL"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_IBM"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_CodeGear"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Microsoft"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_VisualAge_for_Java">
    <rdfs:subClassOf>
```

```
      <owl:Class rdf:about="#Component_IBM"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_VisualAge_Cplusplus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_WebSphere_Studio"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_WebSphere"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_SUSE">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Component_Linux"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_RedHat"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_Kernel"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_Disk_Requirements">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Component"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Component_RedHat">
    <owl:disjointWith rdf:resource="#Component_SUSE"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Kernel"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_Linux"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Component_PowerJ">
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_PowerBuilder"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_Sybase"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Component_Windows_XP">
    <rdfs:subClassOf rdf:resource="#Component_Windows"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Visual_Studio_2008">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_Microsoft"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Component_Kernel">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_Linux"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Component_SUSE"/>
    <owl:disjointWith rdf:resource="#Component_RedHat"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_WebSphere">
    <owl:disjointWith rdf:resource="#Component_VisualAge_for_Java"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_VisualAge_Cplusplus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WebSphere_Studio"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
```

```
        <owl:Class rdf:about="#Component_IBM"/>
      </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Component_Visual_Csharp_.NET">
      <rdfs:subClassOf>
        <owl:Class rdf:about="#Component_Microsoft"/>
      </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Component_Visual_Cplusplus">
      <rdfs:subClassOf>
        <owl:Class rdf:about="#Component_Microsoft"/>
      </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Component_JBuilder">
      <owl:disjointWith>
        <owl:Class rdf:ID="Component_Visual_Caf茅"/>
      </owl:disjointWith>
      <owl:disjointWith>
        <owl:Class rdf:ID="Component_Kylix"/>
      </owl:disjointWith>
      <owl:disjointWith>
        <owl:Class rdf:ID="Component_Delphi"/>
      </owl:disjointWith>
      <owl:disjointWith>
        <owl:Class rdf:ID="Component_CplusplusBuilder"/>
      </owl:disjointWith>
      <rdfs:subClassOf rdf:resource="#Component_CodeGear"/>
    </owl:Class>
    <owl:Class rdf:about="#Component_WebSphere_Studio">
      <owl:disjointWith rdf:resource="#Component_VisualAge_for_Java"/>
      <owl:disjointWith>
        <owl:Class rdf:about="#Component_VisualAge_Cplusplus"/>
      </owl:disjointWith>
      <owl:disjointWith rdf:resource="#Component_WebSphere"/>
      <rdfs:subClassOf>
        <owl:Class rdf:about="#Component_IBM"/>
      </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Component_Windows_Vista">
      <rdfs:subClassOf rdf:resource="#Component_Windows"/>
    </owl:Class>
    <owl:Class rdf:about="#Component_Linux">
      <rdfs:subClassOf>
        <owl:Class rdf:about="#Component_OS"/>
      </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Component_Memory_Requirements">
      <rdfs:subClassOf rdf:resource="#Component"/>
    </owl:Class>
    <owl:Class rdf:ID="Component_Visual_Studio_.NET">
      <rdfs:subClassOf>
        <owl:Class rdf:about="#Component_Microsoft"/>
      </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Component_Windows_ME">
      <rdfs:subClassOf rdf:resource="#Component_Windows"/>
    </owl:Class>
    <owl:Class rdf:ID="Component_Visual_Csharp_2005">
      <rdfs:subClassOf>
        <owl:Class rdf:about="#Component_Microsoft"/>
      </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Component_Access">
      <rdfs:subClassOf>
        <owl:Class rdf:about="#Component_Microsoft"/>
      </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Component_CPU_Requirements">
      <rdfs:subClassOf rdf:resource="#Component"/>
```

```
</owl:Class>
<owl:Class rdf:about="#Component_Delphi">
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Visual_Caf茅"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Kylix"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_JBuilder"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_CplusplusBuilder"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#Component_CodeGear"/>
</owl:Class>
<owl:Class rdf:about="#Component_WebLogic_Workshop">
  <owl:disjointWith rdf:resource="#Component_Tuxedo"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_WebLogic_Express"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_WebLogic_Portal"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_WebLogic_Server"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_Oracle_JDeveloper"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Oracle_Database"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Oracle"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_Visual_Caf茅">
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Kylix"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_JBuilder"/>
  <owl:disjointWith rdf:resource="#Component_Delphi"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_CplusplusBuilder"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#Component_CodeGear"/>
</owl:Class>
<owl:Class rdf:ID="Component_Sun_ONE_Studio">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Sun"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="Component_Sun_ONE"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="Component_Visual_Studio_2005">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_CCM_Model">
  <owl:disjointWith rdf:resource="#Component_OSGI_Model"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_.NET_Model"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_COM_Model"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_JavaBean_Model"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Model"/>
  </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:about="#Component_Oracle">
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Sybase"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Sun"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_MySQL"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_IBM"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_Eclipse"/>
  <owl:disjointWith rdf:resource="#Component_CodeGear"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Platform"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_Model">
  <rdfs:subClassOf rdf:resource="#Component"/>
</owl:Class>
<owl:Class rdf:ID="Component_Visual_Basic">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_Platform">
  <rdfs:subClassOf rdf:resource="#Component"/>
</owl:Class>
<owl:Class rdf:ID="Component_Internet_Explorer">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_Sun_ONE">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Sun"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Component_Sun_ONE_Studio"/>
</owl:Class>
<owl:Class rdf:ID="Component_Visual_Basic_.NET">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_Oracle_Database">
  <owl:disjointWith rdf:resource="#Component_Tuxedo"/>
  <owl:disjointWith rdf:resource="#Component_WebLogic_Workshop"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_WebLogic_Express"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_WebLogic_Portal"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_WebLogic_Server"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_Oracle_JDeveloper"/>
  <rdfs:subClassOf rdf:resource="#Component_Oracle"/>
</owl:Class>
<owl:Class rdf:ID="Component_Visual_Basic_2005">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Component_Microsoft"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Component_Microsoft">
```

```
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Sybase"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Sun"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Oracle"/>
    <owl:disjointWith rdf:resource="#Component_MySQL"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_IBM"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Eclipse"/>
    <owl:disjointWith rdf:resource="#Component_CodeGear"/>
    <rdfs:subClassOf rdf:resource="#Component_Platform"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_Sun">
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Sybase"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Oracle"/>
    <owl:disjointWith rdf:resource="#Component_MySQL"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_IBM"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Eclipse"/>
    <owl:disjointWith rdf:resource="#Component_CodeGear"/>
    <owl:disjointWith rdf:resource="#Component_Microsoft"/>
    <rdfs:subClassOf rdf:resource="#Component_Platform"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_IBM_AIX">
    <rdfs:subClassOf rdf:resource="#Component_Unix"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Visual_Studio">
    <rdfs:subClassOf rdf:resource="#Component_Microsoft"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_FrontPage">
    <rdfs:subClassOf rdf:resource="#Component_Microsoft"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_Sybase">
    <rdfs:subClassOf rdf:resource="#Component_Platform"/>
    <owl:disjointWith rdf:resource="#Component_Sun"/>
    <owl:disjointWith rdf:resource="#Component_Oracle"/>
    <owl:disjointWith rdf:resource="#Component_MySQL"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_IBM"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Eclipse"/>
    <owl:disjointWith rdf:resource="#Component_CodeGear"/>
    <owl:disjointWith rdf:resource="#Component_Microsoft"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Windows_NT">
    <rdfs:subClassOf rdf:resource="#Component_Windows"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_CplusplusBuilder">
    <owl:disjointWith rdf:resource="#Component_Visual_Caf茅"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Kylix"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_JBuilder"/>
    <owl:disjointWith rdf:resource="#Component_Delphi"/>
    <rdfs:subClassOf rdf:resource="#Component_CodeGear"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_OS">
    <rdfs:subClassOf rdf:resource="#Component"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_IBM">
    <rdfs:subClassOf rdf:resource="#Component_Platform"/>
    <owl:disjointWith rdf:resource="#Component_Sybase"/>
    <owl:disjointWith rdf:resource="#Component_Sun"/>
    <owl:disjointWith rdf:resource="#Component_Oracle"/>
    <owl:disjointWith rdf:resource="#Component_MySQL"/>
    <owl:disjointWith rdf:resource="#Component_Eclipse"/>
    <owl:disjointWith rdf:resource="#Component_CodeGear"/>
    <owl:disjointWith rdf:resource="#Component_Microsoft"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_WebLogic_Server">
    <rdfs:subClassOf rdf:resource="#Component_Oracle"/>
    <owl:disjointWith rdf:resource="#Component_Tuxedo"/>
    <owl:disjointWith rdf:resource="#Component_WebLogic_Workshop"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WebLogic_Express"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WebLogic_Portal"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Oracle_JDeveloper"/>
    <owl:disjointWith rdf:resource="#Component_Oracle_Database"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_PowerBuilder">
    <owl:disjointWith rdf:resource="#Component_PowerJ"/>
    <rdfs:subClassOf rdf:resource="#Component_Sybase"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_WebLogic_Portal">
    <owl:disjointWith rdf:resource="#Component_Tuxedo"/>
    <owl:disjointWith rdf:resource="#Component_WebLogic_Workshop"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WebLogic_Express"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_WebLogic_Server"/>
    <owl:disjointWith rdf:resource="#Component_Oracle_JDeveloper"/>
    <owl:disjointWith rdf:resource="#Component_Oracle_Database"/>
    <rdfs:subClassOf rdf:resource="#Component_Oracle"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_VisualAge_Cplusplus">
    <rdfs:subClassOf rdf:resource="#Component_IBM"/>
    <owl:disjointWith rdf:resource="#Component_VisualAge_for_Java"/>
    <owl:disjointWith rdf:resource="#Component_WebSphere_Studio"/>
    <owl:disjointWith rdf:resource="#Component_WebSphere"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Visual_Csharp_2008">
    <rdfs:subClassOf rdf:resource="#Component_Microsoft"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Office">
    <rdfs:subClassOf rdf:resource="#Component_Microsoft"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_WebLogic_Express">
    <rdfs:subClassOf rdf:resource="#Component_Oracle"/>
    <owl:disjointWith rdf:resource="#Component_Tuxedo"/>
    <owl:disjointWith rdf:resource="#Component_WebLogic_Workshop"/>
    <owl:disjointWith rdf:resource="#Component_WebLogic_Portal"/>
    <owl:disjointWith rdf:resource="#Component_WebLogic_Server"/>
    <owl:disjointWith rdf:resource="#Component_Oracle_JDeveloper"/>
    <owl:disjointWith rdf:resource="#Component_Oracle_Database"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_HP-UX">
    <rdfs:subClassOf rdf:resource="#Component_Unix"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_SQL_Server">
    <rdfs:subClassOf rdf:resource="#Component_Microsoft"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_COM_Model">
    <rdfs:subClassOf rdf:resource="#Component_Model"/>
    <owl:disjointWith rdf:resource="#Component_OSGI_Model"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_.NET_Model"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_CCM_Model"/>
    <owl:disjointWith rdf:resource="#Component_JavaBean_Model"/>
```

```
</owl:Class>
<owl:Class rdf:about="#Component_.NET_Model">
  <owl:disjointWith rdf:resource="#Component_OSGI_Model"/>
  <owl:disjointWith rdf:resource="#Component_CCM_Model"/>
  <owl:disjointWith rdf:resource="#Component_COM_Model"/>
  <owl:disjointWith rdf:resource="#Component_JavaBean_Model"/>
  <rdfs:subClassOf rdf:resource="#Component_Model"/>
</owl:Class>
<owl:Class rdf:ID="Component_Visual_Basic_2008">
  <rdfs:subClassOf rdf:resource="#Component_Microsoft"/>
</owl:Class>
<owl:Class rdf:ID="Component_Sun_Solaris">
  <rdfs:subClassOf rdf:resource="#Component_Unix"/>
</owl:Class>
<owl:Class rdf:ID="Component_Windows_2000">
  <rdfs:subClassOf rdf:resource="#Component_Windows"/>
</owl:Class>
<owl:Class rdf:ID="Component_Visual_Cplusplus_.NET">
  <rdfs:subClassOf rdf:resource="#Component_Visual_Cplusplus_2008"/>
</owl:Class>
<owl:Class rdf:ID="Component_Windows_3.X">
  <rdfs:subClassOf rdf:resource="#Component_Windows"/>
</owl:Class>
<owl:Class rdf:ID="Component_FreeBSD">
  <rdfs:subClassOf rdf:resource="#Component_Unix"/>
</owl:Class>
<owl:Class rdf:about="#Component_Kylix">
  <owl:disjointWith rdf:resource="#Component_Visual_Caf茅"/>
  <owl:disjointWith rdf:resource="#Component_JBuilder"/>
  <owl:disjointWith rdf:resource="#Component_Delphi"/>
  <owl:disjointWith rdf:resource="#Component_CplusplusBuilder"/>
  <rdfs:subClassOf rdf:resource="#Component_CodeGear"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasPlatform">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="isPlatformOf"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Component"/>
  <rdfs:range rdf:resource="#Component_Platform"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOS">
  <rdfs:domain rdf:resource="#Component"/>
  <rdfs:range rdf:resource="#Component_OS"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="isOsOf"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isComponentModelOf">
  <rdfs:domain rdf:resource="#Component_Model"/>
  <rdfs:range rdf:resource="#Component"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasComponentModel"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isCPURequirementOf">
  <rdfs:domain rdf:resource="#Component_CPU_Requirements"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasCPURequirement"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Component"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isPlatformOf">
  <rdfs:range rdf:resource="#Component"/>
  <rdfs:domain rdf:resource="#Component_Platform"/>
  <owl:inverseOf rdf:resource="#hasPlatform"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasComponentModel">
  <rdfs:domain rdf:resource="#Component"/>
  <rdfs:range rdf:resource="#Component_Model"/>
  <owl:inverseOf rdf:resource="#isComponentModelOf"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isOsOf">
  <rdfs:domain rdf:resource="#Component_OS"/>
  <rdfs:range rdf:resource="#Component"/>
  <owl:inverseOf rdf:resource="#hasOS"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasCPURequirement">
  <owl:inverseOf rdf:resource="#isCPURequirementOf"/>
  <rdfs:range rdf:resource="#Component_CPU_Requirements"/>
  <rdfs:domain rdf:resource="#Component"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasMemoryRequirement">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="isMemoryRequirementOf"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Component"/>
  <rdfs:range rdf:resource="#Component_Memory_Requirements"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isDiskRequirementOf">
  <rdfs:range rdf:resource="#Component"/>
  <rdfs:domain rdf:resource="#Component_Disk_Requirements"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasDiskRequirement"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isMemoryRequirementOf">
  <rdfs:range rdf:resource="#Component"/>
  <owl:inverseOf rdf:resource="#hasMemoryRequirement"/>
  <rdfs:domain rdf:resource="#Component_Memory_Requirements"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasDiskRequirement">
  <rdfs:range rdf:resource="#Component_Disk_Requirements"/>
  <owl:inverseOf rdf:resource="#isDiskRequirementOf"/>
  <rdfs:domain rdf:resource="#Component"/>
</owl:ObjectProperty>
<Component_Visual_Basic_2008 rdf:ID="ezCryto_.NET"/>
<Component rdf:ID="Component_6"/>
<Component_Visual_Basic_2008 rdf:ID="PowerTCP_FTP_for_.NET">
  <hasDiskRequirement>
    <Component_Disk_Requirements rdf:ID="Disk_2MB">
      <isDiskRequirementOf rdf:resource="#PowerTCP_FTP_for_.NET"/>
    </Component_Disk_Requirements>
  </hasDiskRequirement>
  <hasOS>
    <Component_Windows_2000 rdf:ID="Windows_2000">
      <isOsOf rdf:resource="#PowerTCP_FTP_for_.NET"/>
    </Component_Windows_2000>
  </hasOS>
  <hasOS>
    <Component_Windows_ME rdf:ID="Windows_ME">
      <isOsOf rdf:resource="#PowerTCP_FTP_for_.NET"/>
    </Component_Windows_ME>
  </hasOS>
  <hasOS>
    <Component_Windows_NT rdf:ID="Windows_NT">
      <isOsOf rdf:resource="#PowerTCP_FTP_for_.NET"/>
    </Component_Windows_NT>
  </hasOS>
</Component_Visual_Basic_2008>
<Component_Visual_Basic_2008 rdf:ID="Netrix_DOM_Editor">
  <hasOS>
    <Component_Windows_XP rdf:ID="Windows_XP">
      <isOsOf rdf:resource="#Netrix_DOM_Editor"/>
    </Component_Windows_XP>
  </hasOS>
  <hasOS>
    <Component_Windows_Vista rdf:ID="Windows_Vista">
      <isOsOf rdf:resource="#Netrix_DOM_Editor"/>
```

```
        </Component_Windows_Vista>
      </hasOS>
      <hasDiskRequirement>
        <Component_Disk_Requirements rdf:ID="Disk_7MB">
          <isDiskRequirementOf rdf:resource="#Netrix_DOM_Editor"/>
        </Component_Disk_Requirements>
      </hasDiskRequirement>
    </Component_Visual_Basic_2008>
    <Component_Visual_Basic_2005 rdf:ID="Rainbow_PDF_Server_Based_Converter">
      <hasCPURequirement>
        <Component_CPU_Requirements rdf:ID="Pentium_42.3GHz">
          <isCPURequirementOf rdf:resource="#Rainbow_PDF_Server_Based_Converter"/>
        </Component_CPU_Requirements>
      </hasCPURequirement>
      <hasDiskRequirement>
        <Component_Disk_Requirements rdf:ID="Disk_1GB">
          <isDiskRequirementOf rdf:resource="#Rainbow_PDF_Server_Based_Converter"/>
        </Component_Disk_Requirements>
      </hasDiskRequirement>
      <hasMemoryRequirement>
        <Component_Memory_Requirements rdf:ID="Memory_1GB">
          <isMemoryRequirementOf rdf:resource="#Rainbow_PDF_Server_Based_Converter"/>
        </Component_Memory_Requirements>
      </hasMemoryRequirement>
      <hasOS>
        <Component_Sun_Solaris rdf:ID="Sun_Solaris">
          <isOsOf rdf:resource="#Rainbow_PDF_Server_Based_Converter"/>
        </Component_Sun_Solaris>
      </hasOS>
      <hasOS>
        <Component_RedHat rdf:ID="RedHat">
          <isOsOf rdf:resource="#Rainbow_PDF_Server_Based_Converter"/>
        </Component_RedHat>
      </hasOS>
    </Component_Visual_Basic_2005>
    <Component_Delphi rdf:ID="Xceed_Ultimate_Suite_2008">
      <hasMemoryRequirement>
        <Component_Memory_Requirements rdf:ID="Memory_32MB">
          <isMemoryRequirementOf rdf:resource="#Xceed_Ultimate_Suite_2008"/>
        </Component_Memory_Requirements>
      </hasMemoryRequirement>
      <hasDiskRequirement>
        <Component_Disk_Requirements rdf:ID="Disk_28MB">
          <isDiskRequirementOf rdf:resource="#Xceed_Ultimate_Suite_2008"/>
        </Component_Disk_Requirements>
      </hasDiskRequirement>
    </Component_Delphi>
  </rdf:RDF>
  <!-- Created with Protege (with OWL Plugin 3.4, Build 533)    http://protege.stanford.edu -->
```

A.3 Intrinsic Model

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:p2="http://www.owl-ontologies.com/Ontology1226299693.owl#520.00"
    xmlns:p1="http://www.owl-ontologies.com/Ontology1226299693.owl#194.00"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    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/Ontology1226299693.owl#"
    xmlns:p4="http://www.owl-ontologies.com/Ontology1226299693.owl#877.00"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:p3="http://www.owl-ontologies.com/Ontology1226299693.owl#1301.00"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xml:base="http://www.owl-ontologies.com/Ontology1226299693.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Component_Java">
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_Cplusplus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_.NET"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Component_Type"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_VBX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_JavaScript_AJAX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Componen_VCL"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_DLL"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_Flash_Flex"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_ActiveX_COM"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_ActiveX_EXE">
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_ActiveX_OCX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_ActiveX_DLL"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_ActiveX_.NET_Ready"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_ActiveX_COM"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Component_.NET_Class">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_.NET"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_.NET_WinForm"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_ASP_.NET_WebForm"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_ASP_.NET_AJAX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_.NET_Web_Service"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_.NET_Compact_Framework"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_WPF"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Component_Silverlight"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_Price">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Component"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Component_ASP_.NET_AJAX">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_.NET"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_.NET_WinForm"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_ASP_.NET_WebForm"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_.NET_Class"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_.NET_Web_Service"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_.NET_Compact_Framework"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WPF"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Silverlight"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Component_Cplusplus">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_Type"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Flash_Flex"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Java"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_DLL"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_.NET"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_ActiveX_COM"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_VBX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_JavaScript_AJAX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Componen_VCL"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Component_.NET_Compact_Framework">
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_.NET_WinForm"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_ASP_.NET_WebForm"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_ASP_.NET_AJAX"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Class"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_.NET_Web_Service"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WPF"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Silverlight"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_.NET"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Component_Vender">
    <rdfs:subClassOf rdf:resource="#Component"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_ActiveX_.NET_Ready">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_ActiveX_COM"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_ActiveX_OCX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_ActiveX_DLL"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_ActiveX_EXE"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_.NET">
    <owl:disjointWith rdf:resource="#Component_Cplusplus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_ActiveX_COM"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Java"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_DLL"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_JavaScript_AJAX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Componen_VCL"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Flash_Flex"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_VBX"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_Type"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Component_ActiveX_DLL">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Component_ActiveX_COM"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_ActiveX_OCX"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_ActiveX_EXE"/>
    <owl:disjointWith rdf:resource="#Component_ActiveX_.NET_Ready"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_Type">
    <rdfs:subClassOf rdf:resource="#Component"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Enterprise_JavaBean">
    <rdfs:subClassOf rdf:resource="#Component_Java"/>
  </owl:Class>
```

```
<owl:Class rdf:about="#Component_Silverlight">
  <rdfs:subClassOf rdf:resource="#Component_.NET"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_.NET_WinForm"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_ASP_.NET_WebForm"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_ASP_.NET_AJAX"/>
  <owl:disjointWith rdf:resource="#Component_.NET_Class"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_.NET_Web_Service"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_.NET_Compact_Framework"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_WPF"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="Component_WebLogic_Workshop_JWS_Control">
  <rdfs:subClassOf rdf:resource="#Component_Java"/>
</owl:Class>
<owl:Class rdf:about="#Component_ActiveX_COM">
  <owl:disjointWith>
    <owl:Class rdf:about="#Componen_VCL"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_Java"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_JavaScript_AJAX"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#Component_Type"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_Flash_Flex"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_DLL"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_.NET"/>
  <owl:disjointWith rdf:resource="#Component_Cplusplus"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_VBX"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Component_.NET_WinForm">
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_ASP_.NET_WebForm"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_ASP_.NET_AJAX"/>
  <owl:disjointWith rdf:resource="#Component_.NET_Class"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_.NET_Web_Service"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_.NET_Compact_Framework"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Component_WPF"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Component_Silverlight"/>
  <rdfs:subClassOf rdf:resource="#Component_.NET"/>
</owl:Class>
<owl:Class rdf:ID="Component_Version">
  <rdfs:subClassOf rdf:resource="#Component"/>
</owl:Class>
<owl:Class rdf:ID="Component_JavaBean">
  <rdfs:subClassOf rdf:resource="#Component_Java"/>
</owl:Class>
<owl:Class rdf:ID="Component_Name">
  <rdfs:subClassOf rdf:resource="#Component"/>
</owl:Class>
<owl:Class rdf:about="#Component_.NET_Web_Service">
  <owl:disjointWith rdf:resource="#Component_.NET_WinForm"/>
```

```
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_ASP_.NET_WebForm"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_ASP_.NET_AJAX"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Class"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Compact_Framework"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WPF"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Silverlight"/>
    <rdfs:subClassOf rdf:resource="#Component_.NET"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_DLL">
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Flash_Flex"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Java"/>
    <rdfs:subClassOf rdf:resource="#Component_Type"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_VBX"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_ActiveX_COM"/>
    <owl:disjointWith rdf:resource="#Component_.NET"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_JavaScript_AJAX"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Cplusplus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Componen_VCL"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Component_Java_Class">
    <rdfs:subClassOf rdf:resource="#Component_Java"/>
  </owl:Class>
  <owl:Class rdf:about="#Componen_VCL">
    <owl:disjointWith rdf:resource="#Component_DLL"/>
    <rdfs:subClassOf rdf:resource="#Component_Type"/>
    <owl:disjointWith rdf:resource="#Component_Cplusplus"/>
    <owl:disjointWith rdf:resource="#Component_.NET"/>
    <owl:disjointWith rdf:resource="#Component_ActiveX_COM"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_JavaScript_AJAX"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_Flash_Flex"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_VBX"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Java"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_Flash_Flex">
    <owl:disjointWith rdf:resource="#Component_Cplusplus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_JavaScript_AJAX"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Java"/>
    <owl:disjointWith rdf:resource="#Component_.NET"/>
    <owl:disjointWith rdf:resource="#Component_ActiveX_COM"/>
    <owl:disjointWith rdf:resource="#Component_DLL"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_VBX"/>
    </owl:disjointWith>
    <rdfs:subClassOf rdf:resource="#Component_Type"/>
    <owl:disjointWith rdf:resource="#Componen_VCL"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_ActiveX_OCX">
    <owl:disjointWith rdf:resource="#Component_ActiveX_DLL"/>
    <owl:disjointWith rdf:resource="#Component_ActiveX_EXE"/>
    <owl:disjointWith rdf:resource="#Component_ActiveX_.NET_Ready"/>
    <rdfs:subClassOf rdf:resource="#Component_ActiveX_COM"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Java_Servlet">
    <rdfs:subClassOf rdf:resource="#Component_Java"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Date">
    <rdfs:subClassOf rdf:resource="#Component"/>
  </owl:Class>
  <owl:Class rdf:ID="Component_Java_Applet">
    <rdfs:subClassOf rdf:resource="#Component_Java"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_ASP_.NET_WebForm">
    <rdfs:subClassOf rdf:resource="#Component_.NET"/>
    <owl:disjointWith rdf:resource="#Component_.NET_WinForm"/>
    <owl:disjointWith rdf:resource="#Component_ASP_.NET_AJAX"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Class"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Web_Service"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Compact_Framework"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_WPF"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_Silverlight"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_JavaScript_AJAX">
    <owl:disjointWith rdf:resource="#Componen_VCL"/>
    <owl:disjointWith rdf:resource="#Component_Cplusplus"/>
    <owl:disjointWith rdf:resource="#Component_Flash_Flex"/>
    <owl:disjointWith rdf:resource="#Component_Java"/>
    <owl:disjointWith rdf:resource="#Component_ActiveX_COM"/>
    <owl:disjointWith rdf:resource="#Component_.NET"/>
    <rdfs:subClassOf rdf:resource="#Component_Type"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Component_VBX"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Component_DLL"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_WPF">
    <owl:disjointWith rdf:resource="#Component_.NET_WinForm"/>
    <owl:disjointWith rdf:resource="#Component_ASP_.NET_WebForm"/>
    <owl:disjointWith rdf:resource="#Component_ASP_.NET_AJAX"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Class"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Web_Service"/>
    <owl:disjointWith rdf:resource="#Component_.NET_Compact_Framework"/>
    <owl:disjointWith rdf:resource="#Component_Silverlight"/>
    <rdfs:subClassOf rdf:resource="#Component_.NET"/>
  </owl:Class>
  <owl:Class rdf:about="#Component_VBX">
    <owl:disjointWith rdf:resource="#Component_DLL"/>
    <owl:disjointWith rdf:resource="#Component_.NET"/>
    <owl:disjointWith rdf:resource="#Component_Java"/>
    <owl:disjointWith rdf:resource="#Component_ActiveX_COM"/>
    <owl:disjointWith rdf:resource="#Component_Cplusplus"/>
    <owl:disjointWith rdf:resource="#Component_Flash_Flex"/>
    <owl:disjointWith rdf:resource="#Component_JavaScript_AJAX"/>
    <rdfs:subClassOf rdf:resource="#Component_Type"/>
    <owl:disjointWith rdf:resource="#Componen_VCL"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="hasName">
    <rdfs:domain rdf:resource="#Component"/>
    <rdfs:range rdf:resource="#Component_Name"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="isNameOf"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasPrice">
    <rdfs:domain rdf:resource="#Component"/>
    <rdfs:range rdf:resource="#Component_Price"/>
    <owl:inverseOf>
```

```
      <owl:ObjectProperty rdf:ID="isPriceOf"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasVender">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="isVenderOf"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#Component_Vender"/>
    <rdfs:domain rdf:resource="#Component"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasVersion">
    <rdfs:domain rdf:resource="#Component"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="isVersionOf"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#Component_Version"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#isVersionOf">
    <rdfs:domain rdf:resource="#Component_Version"/>
    <rdfs:range rdf:resource="#Component"/>
    <owl:inverseOf rdf:resource="#hasVersion"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#isVenderOf">
    <rdfs:range rdf:resource="#Component"/>
    <owl:inverseOf rdf:resource="#hasVender"/>
    <rdfs:domain rdf:resource="#Component_Vender"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#isPriceOf">
    <rdfs:domain rdf:resource="#Component_Price"/>
    <owl:inverseOf rdf:resource="#hasPrice"/>
    <rdfs:range rdf:resource="#Component"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#isNameOf">
    <rdfs:range rdf:resource="#Component"/>
    <rdfs:domain rdf:resource="#Component_Name"/>
    <owl:inverseOf rdf:resource="#hasName"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasDate">
    <rdfs:domain rdf:resource="#Component"/>
    <rdfs:range rdf:resource="#Component_Date"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="isDateOf"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#isDateOf">
    <rdfs:domain rdf:resource="#Component_Date"/>
    <rdfs:range rdf:resource="#Component"/>
    <owl:inverseOf rdf:resource="#hasDate"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasA"/>
  <owl:AllDifferent/>
  <Component_.NET_Class rdf:ID="Netrix_DOM_Editor">
    <hasVender>
      <Component_Vender rdf:ID="Guru_Components">
        <isVenderOf rdf:resource="#Netrix_DOM_Editor"/>
      </Component_Vender>
    </hasVender>
    <hasVersion>
      <Component_Version rdf:ID="V1.6">
        <isVersionOf rdf:resource="#Netrix_DOM_Editor"/>
      </Component_Version>
    </hasVersion>
    <hasPrice>
      <Component_Price rdf:about="# 877.00">
        <isPriceOf rdf:resource="#Netrix_DOM_Editor"/>
      </Component_Price>
    </hasPrice>
  </Component_.NET_Class>
  <Component_WPF rdf:ID="Xceed_Ultimate_Suite_2008">
```

```
    <hasPrice>
      <Component_Price rdf:about="# 520.00">
        <isPriceOf rdf:resource="#Xceed_Ultimate_Suite_2008"/>
      </Component_Price>
    </hasPrice>
    <hasVender>
      <Component_Vender rdf:ID="Xceed_Software">
        <isVenderOf rdf:resource="#Xceed_Ultimate_Suite_2008"/>
      </Component_Vender>
    </hasVender>
    <hasVersion>
      <Component_Version rdf:ID="V5">
        <isVersionOf rdf:resource="#Xceed_Ultimate_Suite_2008"/>
      </Component_Version>
    </hasVersion>
  </Component_WPF>
  <Component_Java_Class rdf:ID="Rainbow_PDF_Server_Based_Converter">
    <hasVersion>
      <Component_Version rdf:ID="V2.0">
        <isVersionOf rdf:resource="#Rainbow_PDF_Server_Based_Converter"/>
      </Component_Version>
    </hasVersion>
    <hasPrice>
      <Component_Price rdf:about="#1301.00">
        <isPriceOf rdf:resource="#Rainbow_PDF_Server_Based_Converter"/>
      </Component_Price>
    </hasPrice>
    <hasVender>
      <Component_Vender rdf:ID="Antenna_House">
        <isVenderOf rdf:resource="#Rainbow_PDF_Server_Based_Converter"/>
      </Component_Vender>
    </hasVender>
  </Component_Java_Class>
  <Component_.NET_Class rdf:ID="ezCrypto_.NET">
    <hasPrice>
      <Component_Price rdf:about="#拢65.00">
        <isPriceOf rdf:resource="#ezCrypto_.NET"/>
      </Component_Price>
    </hasPrice>
    <hasVersion>
      <Component_Version rdf:ID="V2.0.2">
        <isVersionOf rdf:resource="#ezCrypto_.NET"/>
      </Component_Version>
    </hasVersion>
    <hasVender>
      <Component_Vender rdf:ID="Component_Designs">
        <isVenderOf rdf:resource="#ezCrypto_.NET"/>
      </Component_Vender>
    </hasVender>
  </Component_.NET_Class>
  <Component_.NET_Class rdf:ID="Power_TCP_FTP_for_.NET">
    <hasVersion>
      <Component_Version rdf:ID="V3.0.4">
        <isVersionOf rdf:resource="#Power_TCP_FTP_for_.NET"/>
      </Component_Version>
    </hasVersion>
    <hasVender>
      <Component_Vender rdf:ID="Dart_Communications">
        <isVenderOf rdf:resource="#Power_TCP_FTP_for_.NET"/>
      </Component_Vender>
    </hasVender>
    <hasPrice>
      <Component_Price rdf:about="#194.00">
        <isPriceOf rdf:resource="#Power_TCP_FTP_for_.NET"/>
      </Component_Price>
    </hasPrice>
  </Component_.NET_Class>
</rdf:RDF>
```

```
<!-- Created with Protege (with OWL Plugin 3.4, Build 533)    http://protege.stanford.edu -->
```

Appendix B: The OWL Doc of the Financial Domain Ontology

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/ontologies/2011/6/Ontology1310633460953.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     ontologyIRI="http://www.semanticweb.org/ontologies/2011/6/Ontology1310633460953.owl">
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Declaration>
        <Class IRI="#Account"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Accounting_Integration"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Accounts_Receivables_&amp;_Payables"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Accounts_Receivables_and_Payables"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Adobe_Acrobat_Integration"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Advanced_Analytics"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Aggregation"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Allocations"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Alternative_Investment_Systems"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Analytics"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Asset_Management_Systems"/>
    </Declaration>
    <Declaration>
```

```
        <Class IRI="#Association_Class"/>
</Declaration>
<Declaration>
        <Class IRI="#Authorisation"/>
</Declaration>
<Declaration>
        <Class IRI="#Authorising"/>
</Declaration>
<Declaration>
        <Class IRI="#Automated_Document_Mailings"/>
</Declaration>
<Declaration>
        <Class IRI="#Automated_PDF_Creation_and_Collation"/>
</Declaration>
<Declaration>
        <Class IRI="#Automated_Price_Updates"/>
</Declaration>
<Declaration>
        <Class IRI="#Automated_Workflow"/>
</Declaration>
<Declaration>
        <Class IRI="#Basic_Portfolio_Management_"/>
</Declaration>
<Declaration>
        <Class IRI="#Business_Intelligence"/>
</Declaration>
<Declaration>
        <Class IRI="#Calculation"/>
</Declaration>
<Declaration>
        <Class IRI="#Calendar_/_Schedule_"/>
</Declaration>
<Declaration>
        <Class IRI="#Capital_Markets_Systems"/>
</Declaration>
<Declaration>
        <Class IRI="#Capturing"/>
</Declaration>
<Declaration>
        <Class IRI="#Card_Authorisation"/>
</Declaration>
<Declaration>
        <Class IRI="#Card_Processing_/_Payments"/>
</Declaration>
<Declaration>
        <Class IRI="#Cash_Flow_Forecasting"/>
</Declaration>
<Declaration>
        <Class IRI="#Cash_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Charting"/>
</Declaration>
<Declaration>
        <Class IRI="#Charting_and_Graphing"/>
</Declaration>
<Declaration>
        <Class IRI="#Charting_and_Graphing_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Check-in_&amp;_Check-out"/>
</Declaration>
<Declaration>
        <Class IRI="#Chip_&amp;_PIN"/>
</Declaration>
<Declaration>
        <Class IRI="#Clearing_and_Settlement_Processes"/>
</Declaration>
<Declaration>
```

```
        <Class IRI="#Client_Acquisition"/>
</Declaration>
<Declaration>
        <Class IRI="#Client_Wealth_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Collaborating"/>
</Declaration>
<Declaration>
        <Class IRI="#Communication"/>
</Declaration>
<Declaration>
        <Class IRI="#Communication_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#Consultancy"/>
</Declaration>
<Declaration>
        <Class IRI="#Contact_&amp;_Document_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Contributions_Processing"/>
</Declaration>
<Declaration>
        <Class IRI="#Correspondence_Tracking"/>
</Declaration>
<Declaration>
        <Class IRI="#Credit_Administration"/>
</Declaration>
<Declaration>
        <Class IRI="#Customer_Relationship_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Customized_User_Dashboards"/>
</Declaration>
<Declaration>
        <Class IRI="#Dashboard"/>
</Declaration>
<Declaration>
        <Class IRI="#Dashboards_and_Visualization"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Conversion"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Conversion_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Distribution_Systems"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Editing"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Management_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Management_Systems"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Security"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Security_MVICS"/>
</Declaration>
<Declaration>
```

```
        <Class IRI="#Data_Solutions"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Transfer"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Transfer_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Validation"/>
</Declaration>
<Declaration>
        <Class IRI="#Data_Validation_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Database_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Database_Management_1"/>
</Declaration>
<Declaration>
        <Class IRI="#Database_Management_2"/>
</Declaration>
<Declaration>
        <Class IRI="#Database_Management_3"/>
</Declaration>
<Declaration>
        <Class IRI="#Database_Management_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Database_Security_1"/>
</Declaration>
<Declaration>
        <Class IRI="#Database_Security_2"/>
</Declaration>
<Declaration>
        <Class IRI="#Deal_Flow"/>
</Declaration>
<Declaration>
        <Class IRI="#Dealing_and_Order_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Deliver"/>
</Declaration>
<Declaration>
        <Class IRI="#Distributions_Processing"/>
</Declaration>
<Declaration>
        <Class IRI="#Distributions_Processing_2"/>
</Declaration>
<Declaration>
        <Class IRI="#Distributions_Processing_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#Document_Importing"/>
</Declaration>
<Declaration>
        <Class IRI="#Document_Log"/>
</Declaration>
<Declaration>
        <Class IRI="#Document_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Document_Profiling"/>
</Declaration>
<Declaration>
        <Class IRI="#Document_Scanning"/>
</Declaration>
<Declaration>
```

```
        <Class IRI="#E-mail"/>
</Declaration>
<Declaration>
        <Class IRI="#Electronic_Commerce"/>
</Declaration>
<Declaration>
        <Class IRI="#Email"/>
</Declaration>
<Declaration>
        <Class IRI="#Email_Archive"/>
</Declaration>
<Declaration>
        <Class IRI="#Email_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Email_Rules"/>
</Declaration>
<Declaration>
        <Class IRI="#Employee_Profile"/>
</Declaration>
<Declaration>
        <Class IRI="#Employee_Transfer,_Promotions_&amp;_Increments"/>
</Declaration>
<Declaration>
        <Class IRI="#Fees_and_Commissions"/>
</Declaration>
<Declaration>
        <Class IRI="#Financial_Accounting"/>
</Declaration>
<Declaration>
        <Class IRI="#Fund_Investment_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Hedge_Funds"/>
</Declaration>
<Declaration>
        <Class IRI="#Import_Transaction_History"/>
</Declaration>
<Declaration>
        <Class IRI="#Information_Search"/>
</Declaration>
<Declaration>
        <Class IRI="#Investment_Records_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Investor_Registration"/>
</Declaration>
<Declaration>
        <Class IRI="#Investor_Reporting"/>
</Declaration>
<Declaration>
        <Class IRI="#Knowledge_Delivery"/>
</Declaration>
<Declaration>
        <Class IRI="#Leave_Management_System"/>
</Declaration>
<Declaration>
        <Class IRI="#MS_Word_Integration"/>
</Declaration>
<Declaration>
        <Class IRI="#Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Management_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#Management_Fees"/>
</Declaration>
<Declaration>
```

```
        <Class IRI="#Management_Reporting"/>
</Declaration>
<Declaration>
        <Class IRI="#Mangement_2"/>
</Declaration>
<Declaration>
        <Class IRI="#Markers_and_Trend_Lines"/>
</Declaration>
<Declaration>
        <Class IRI="#Market_Surveillance_and_Monitoring"/>
</Declaration>
<Declaration>
        <Class IRI="#Marketing_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#Marketing_Tools"/>
</Declaration>
<Declaration>
        <Class IRI="#Menu"/>
</Declaration>
<Declaration>
        <Class IRI="#Microfinance_/_Microbanking"/>
</Declaration>
<Declaration>
        <Class IRI="#Modelling"/>
</Declaration>
<Declaration>
        <Class IRI="#Modelling_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Modelling_Software"/>
</Declaration>
<Declaration>
        <Class IRI="#Monitoring"/>
</Declaration>
<Declaration>
        <Class IRI="#Mutual_Funds_/_Unit_trusts"/>
</Declaration>
<Declaration>
        <Class IRI="#Navigation"/>
</Declaration>
<Declaration>
        <Class IRI="#Navigation_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#Notifications_&amp;_Approvals"/>
</Declaration>
<Declaration>
        <Class IRI="#Notifications_&amp;_Approvals_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#Outlook_and_Email_Integration"/>
</Declaration>
<Declaration>
        <Class IRI="#PDF"/>
</Declaration>
<Declaration>
        <Class IRI="#PDF_Creation"/>
</Declaration>
<Declaration>
        <Class IRI="#PDF_Creator"/>
</Declaration>
<Declaration>
        <Class IRI="#Payment_System"/>
</Declaration>
<Declaration>
        <Class IRI="#Payment_System_3"/>
</Declaration>
<Declaration>
```

```
        <Class IRI="#Payment_system_1"/>
</Declaration>
<Declaration>
        <Class IRI="#Payment_system_2"/>
</Declaration>
<Declaration>
        <Class IRI="#Payroll"/>
</Declaration>
<Declaration>
        <Class IRI="#Payroll_Management_System"/>
</Declaration>
<Declaration>
        <Class IRI="#Performance_Analytics"/>
</Declaration>
<Declaration>
        <Class IRI="#Planning"/>
</Declaration>
<Declaration>
        <Class IRI="#Portfolio_/_Fund_Management_Systems"/>
</Declaration>
<Declaration>
        <Class IRI="#Portfolio_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Portfolio_Management_1"/>
</Declaration>
<Declaration>
        <Class IRI="#Price_Alerts"/>
</Declaration>
<Declaration>
        <Class IRI="#Private_Equity"/>
</Declaration>
<Declaration>
        <Class IRI="#Product_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Profile"/>
</Declaration>
<Declaration>
        <Class IRI="#Purchasing_and_Transaction"/>
</Declaration>
<Declaration>
        <Class IRI="#Query_and_Analysis"/>
</Declaration>
<Declaration>
        <Class IRI="#Real_Estate_Investment_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Records_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Redemptions"/>
</Declaration>
<Declaration>
        <Class IRI="#Reference_Data_Systems"/>
</Declaration>
<Declaration>
        <Class IRI="#Reimbursement_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Reporting"/>
</Declaration>
<Declaration>
        <Class IRI="#Reporting_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Reporting_MVICS_Association_Class"/>
</Declaration>
<Declaration>
```

```
        <Class IRI="#Retrieve_Transactions"/>
</Declaration>
<Declaration>
        <Class IRI="#Revenue_Analytics"/>
</Declaration>
<Declaration>
        <Class IRI="#Risk_Analytics"/>
</Declaration>
<Declaration>
        <Class IRI="#Scheduling_and_Calendaring"/>
</Declaration>
<Declaration>
        <Class IRI="#Search"/>
</Declaration>
<Declaration>
        <Class IRI="#Search_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Search_and_navigation"/>
</Declaration>
<Declaration>
        <Class IRI="#Settlement"/>
</Declaration>
<Declaration>
        <Class IRI="#Settlement_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#Site_Navigation"/>
</Declaration>
<Declaration>
        <Class IRI="#Statement"/>
</Declaration>
<Declaration>
        <Class IRI="#Subscriptions"/>
</Declaration>
<Declaration>
        <Class IRI="#Switching"/>
</Declaration>
<Declaration>
        <Class IRI="#Switching_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#System_Administration"/>
</Declaration>
<Declaration>
        <Class IRI="#Task_Manager"/>
</Declaration>
<Declaration>
        <Class IRI="#Text_and_Word_Processing"/>
</Declaration>
<Declaration>
        <Class IRI="#Text_and_Word_Processing_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Time_/_Date"/>
</Declaration>
<Declaration>
        <Class IRI="#Time_Management_System"/>
</Declaration>
<Declaration>
        <Class IRI="#Toolbar"/>
</Declaration>
<Declaration>
        <Class IRI="#Tracking"/>
</Declaration>
<Declaration>
        <Class IRI="#Trade_Order_and_Risk_Management"/>
</Declaration>
<Declaration>
```

```
        <Class IRI="#Transfer"/>
</Declaration>
<Declaration>
        <Class IRI="#Transfers_and_Defaults"/>
</Declaration>
<Declaration>
        <Class IRI="#UserAdministration"/>
</Declaration>
<Declaration>
        <Class IRI="#User_Administration"/>
</Declaration>
<Declaration>
        <Class IRI="#User_Administration_Agg"/>
</Declaration>
<Declaration>
        <Class IRI="#User_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Validating"/>
</Declaration>
<Declaration>
        <Class IRI="#Version_Control"/>
</Declaration>
<Declaration>
        <Class IRI="#Version_Control_MVICS"/>
</Declaration>
<Declaration>
        <Class IRI="#Wealth_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Wealth_Planning"/>
</Declaration>
<Declaration>
        <Class IRI="#Web_Interface"/>
</Declaration>
<Declaration>
        <Class IRI="#Wholesale_/_Commercial_Banking_Systems"/>
</Declaration>
<Declaration>
        <Class IRI="#Workflow"/>
</Declaration>
<Declaration>
        <Class IRI="#Workflow-Based_Task_Routing"/>
</Declaration>
<Declaration>
        <Class IRI="#Workflow_Management"/>
</Declaration>
<Declaration>
        <Class IRI="#Yield_Calculations"/>
</Declaration>
<Declaration>
        <ObjectProperty IRI="#IsA"/>
</Declaration>
<EquivalentClasses>
        <Class IRI="#Account"/>
        <Class IRI="#Accounting_Integration"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Accounts_Receivables_and_Payables"/>
        <Class IRI="#Payment_system_1"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Advanced_Analytics"/>
        <Class IRI="#Analytics"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Allocations"/>
        <Class IRI="#Data_Management_MVICS"/>
</EquivalentClasses>
```

```
<EquivalentClasses>
        <Class IRI="#Analytics"/>
        <Class IRI="#Cash_Flow_Forecasting"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Analytics"/>
        <Class IRI="#Client_Acquisition"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Analytics"/>
        <Class IRI="#Customer_Relationship_Management"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Analytics"/>
        <Class IRI="#Performance_Analytics"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Analytics"/>
        <Class IRI="#Query_and_Analysis"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Analytics"/>
        <Class IRI="#Revenue_Analytics"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Analytics"/>
        <Class IRI="#Risk_Analytics"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Authorising"/>
        <Class IRI="#Card_Authorisation"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Automated_PDF_Creation_and_Collation"/>
        <Class IRI="#PDF_Creation"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Automated_Price_Updates"/>
        <Class IRI="#Distributions_Processing_Agg"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Automated_Price_Updates"/>
        <Class IRI="#Transfer"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Automated_Workflow"/>
        <Class IRI="#Workflow_Management"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Calculation"/>
        <Class IRI="#Yield_Calculations"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Calendar_/_Schedule_"/>
        <Class IRI="#Scheduling_and_Calendaring"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Cash_Management"/>
        <Class IRI="#Management_Agg"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Charting"/>
        <Class IRI="#Charting_and_Graphing_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Charting_and_Graphing_MVICS"/>
        <Class IRI="#Dashboards_and_Visualization"/>
</EquivalentClasses>
<EquivalentClasses>
```

```
        <Class IRI="#Charting_and_Graphing_MVICS"/>
        <Class IRI="#Markers_and_Trend_Lines"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Check-in_&amp;_Check-out"/>
        <Class IRI="#UserAdministration"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Chip_&amp;_PIN"/>
        <Class IRI="#Database_Security_2"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Clearing_and_Settlement_Processes"/>
        <Class IRI="#Settlement_Agg"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Client_Acquisition"/>
        <Class IRI="#Profile"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Client_Wealth_Management"/>
        <Class IRI="#Management_Agg"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Communication"/>
        <Class IRI="#Communication_Agg"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Consultancy"/>
        <Class IRI="#Database_Management_1"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Contributions_Processing"/>
        <Class IRI="#Payment_System_3"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Correspondence_Tracking"/>
        <Class IRI="#Tracking"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Customer_Relationship_Management"/>
        <Class IRI="#Email_MVICS"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Customer_Relationship_Management"/>
        <Class IRI="#Planning"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Customer_Relationship_Management"/>
        <Class IRI="#Search_MVICS"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Customer_Relationship_Management"/>
        <Class IRI="#Tracking"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Customer_Relationship_Management"/>
        <Class IRI="#UserAdministration"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Customer_Relationship_Management"/>
        <Class IRI="#Workflow"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Customized_User_Dashboards"/>
        <Class IRI="#User_Administration_Agg"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Dashboard"/>
        <Class IRI="#Dashboards_and_Visualization"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Data_Management_Systems"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Dealing_and_Order_Management"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Employee_Transfer,_Promotions_&amp;_Increments"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Import_Transaction_History"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Investment_Records_Management"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Management_Fees"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Price_Alerts"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Records_Management"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Management_MVICS"/>
        <Class IRI="#Wealth_Planning"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Transfer_MVICS"/>
        <Class IRI="#Knowledge_Delivery"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Data_Validation_MVICS"/>
        <Class IRI="#Validating"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Database_Security_1"/>
        <Class IRI="#Settlement"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Distributions_Processing"/>
        <Class IRI="#Distributions_Processing_2"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Document_Importing"/>
        <Class IRI="#Document_Management"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Document_Log"/>
        <Class IRI="#Document_Management"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Document_Management"/>
        <Class IRI="#Document_Scanning"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Document_Profiling"/>
        <Class IRI="#Profile"/>
```

```
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Email_Archive"/>
    <Class IRI="#Email_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Email_MVICS"/>
    <Class IRI="#Email_Rules"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Email_MVICS"/>
    <Class IRI="#Outlook_and_Email_Integration"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Employee_Profile"/>
    <Class IRI="#Profile"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Fees_and_Commissions"/>
    <Class IRI="#Payment_system_1"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Information_Search"/>
    <Class IRI="#Search_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Investor_Reporting"/>
    <Class IRI="#Reporting_MVICS_Association_Class"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Leave_Management_System"/>
    <Class IRI="#Management_Agg"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#MS_Word_Integration"/>
    <Class IRI="#Text_and_Word_Processing_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Management_Agg"/>
    <Class IRI="#Payroll_Management_System"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Management_Agg"/>
    <Class IRI="#Reimbursement_Management"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Management_Agg"/>
    <Class IRI="#Task_Manager"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Management_Agg"/>
    <Class IRI="#User_Management"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Management_Reporting"/>
    <Class IRI="#Reporting_MVICS_Association_Class"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Mangement_2"/>
    <Class IRI="#Product_Management"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Market_Surveillance_and_Monitoring"/>
    <Class IRI="#Monitoring"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Marketing_Agg"/>
    <Class IRI="#Marketing_Tools"/>
</EquivalentClasses>
```

```
<EquivalentClasses>
    <Class IRI="#Modelling"/>
    <Class IRI="#Modelling_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Modelling_MVICS"/>
    <Class IRI="#Modelling_Software"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Navigation"/>
    <Class IRI="#Search_and_navigation"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Navigation_Agg"/>
    <Class IRI="#Site_Navigation"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Notifications_&amp;_Approvals"/>
    <Class IRI="#Notifications_&amp;_Approvals_Agg"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Payment_System"/>
    <Class IRI="#Payment_System_3"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Payment_system_1"/>
    <Class IRI="#Reference_Data_Systems"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Payment_system_1"/>
    <Class IRI="#Subscriptions"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Payment_system_2"/>
    <Class IRI="#Redemptions"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Planning"/>
    <Class IRI="#Wealth_Planning"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Portfolio_Management"/>
    <Class IRI="#Portfolio_Management_1"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Query_and_Analysis"/>
    <Class IRI="#Search_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Retrieve_Transactions"/>
    <Class IRI="#Search_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Search"/>
    <Class IRI="#Search_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Search_MVICS"/>
    <Class IRI="#Search_and_navigation"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Statement"/>
    <Class IRI="#Wealth_Planning"/>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#Switching"/>
    <Class IRI="#Switching_Agg"/>
</EquivalentClasses>
<EquivalentClasses>
```

```
        <Class IRI="#Time_/_Date"/>
        <Class IRI="#Time_Management_System"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Transfer"/>
        <Class IRI="#Transfers_and_Defaults"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Version_Control"/>
        <Class IRI="#Version_Control_MVICS"/>
    </EquivalentClasses>
    <EquivalentClasses>
        <Class IRI="#Workflow"/>
        <Class IRI="#Workflow-Based_Task_Routing"/>
    </EquivalentClasses>
    <SubClassOf>
        <Class IRI="#Account"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Accounting_Integration"/>
        <Class IRI="#Financial_Accounting"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Accounts_Receivables_&amp;_Payables"/>
        <Class IRI="#Financial_Accounting"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Accounts_Receivables_and_Payables"/>
        <Class IRI="#Payroll"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Adobe_Acrobat_Integration"/>
        <Class IRI="#Deliver"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Advanced_Analytics"/>
        <Class IRI="#Business_Intelligence"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Aggregation"/>
        <Class abbreviatedIRI=":Thing"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Allocations"/>
        <Class IRI="#Hedge_Funds"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Alternative_Investment_Systems"/>
        <Class IRI="#Asset_Management_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Analytics"/>
        <Class IRI="#Aggregation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Authorisation"/>
        <Class IRI="#Card_Processing_/_Payments"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Authorising"/>
        <Class IRI="#Authorisation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Automated_Document_Mailings"/>
        <Class IRI="#Deal_Flow"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Automated_PDF_Creation_and_Collation"/>
        <Class IRI="#Deal_Flow"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Automated_Price_Updates"/>
        <Class IRI="#Basic_Portfolio_Management_"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Automated_Workflow"/>
        <Class IRI="#Collaborating"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Basic_Portfolio_Management_"/>
        <Class IRI="#Portfolio_/_Fund_Management_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Business_Intelligence"/>
        <Class IRI="#Data_Solutions"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Calculation"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Calendar_/_Schedule_"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Capital_Markets_Systems"/>
        <Class IRI="#Asset_Management_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Capturing"/>
        <Class IRI="#Contact_&amp;_Document_Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Card_Authorisation"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Card_Processing_/_Payments"/>
        <Class IRI="#Wholesale_/_Commercial_Banking_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Cash_Flow_Forecasting"/>
        <Class IRI="#Alternative_Investment_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Cash_Management"/>
        <Class IRI="#Fund_Investment_Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Charting"/>
        <Class IRI="#Fund_Investment_Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Charting_and_Graphing"/>
        <Class IRI="#Distributions_Processing_2"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Charting_and_Graphing_MVICS"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Check-in_&amp;_Check-out"/>
        <Class IRI="#Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Chip_&amp;_PIN"/>
        <Class IRI="#Card_Processing_/_Payments"/>
```

```
</SubClassOf>
<SubClassOf>
      <Class IRI="#Clearing_and_Settlement_Processes"/>
      <Class IRI="#Capital_Markets_Systems"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Client_Acquisition"/>
      <Class IRI="#Wealth_Management"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Client_Wealth_Management"/>
      <Class IRI="#Wealth_Management"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Collaborating"/>
      <Class IRI="#Contact_&amp;_Document_Management"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Communication"/>
      <Class IRI="#Electronic_Commerce"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Communication_Agg"/>
      <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Consultancy"/>
      <Class IRI="#Card_Processing_/_Payments"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Contact_&amp;_Document_Management"/>
      <Class IRI="#Alternative_Investment_Systems"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Contributions_Processing"/>
      <Class IRI="#Private_Equity"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Correspondence_Tracking"/>
      <Class IRI="#Deal_Flow"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Credit_Administration"/>
      <Class IRI="#Wholesale_/_Commercial_Banking_Systems"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Customer_Relationship_Management"/>
      <Class IRI="#Data_Solutions"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Customized_User_Dashboards"/>
      <Class IRI="#Deal_Flow"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Dashboard"/>
      <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Dashboards_and_Visualization"/>
      <Class IRI="#Business_Intelligence"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Conversion"/>
      <Class IRI="#Payment_System_3"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Conversion_MVICS"/>
      <Class IRI="#Payment_System_3"/>
</SubClassOf>
```

```
<SubClassOf>
      <Class IRI="#Data_Distribution_Systems"/>
      <Class IRI="#Data_Solutions"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Editing"/>
      <Class IRI="#Payment_System_3"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Management"/>
      <Class IRI="#Distributions_Processing_2"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Management_MVICS"/>
      <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Management_Systems"/>
      <Class IRI="#Data_Distribution_Systems"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Security"/>
      <Class IRI="#Payment_System_3"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Security_MVICS"/>
      <Class IRI="#Transfer"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Solutions"/>
      <Class abbreviatedIRI=":Thing"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Transfer"/>
      <Class IRI="#Transfer"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Transfer_MVICS"/>
      <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Validation"/>
      <Class IRI="#Payment_System_3"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Data_Validation_MVICS"/>
      <Class IRI="#Transfer"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Database_Management"/>
      <Class IRI="#Payment_System_3"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Database_Management_1"/>
      <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Database_Management_2"/>
      <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Database_Management_3"/>
      <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Database_Management_MVICS"/>
      <Class IRI="#Transfer"/>
</SubClassOf>
<SubClassOf>
```

```
        <Class IRI="#Database_Security_1"/>
        <Class IRI="#Aggregation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Database_Security_2"/>
        <Class IRI="#Aggregation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Deal_Flow"/>
        <Class IRI="#Private_Equity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Dealing_and_Order_Management"/>
        <Class IRI="#Fund_Investment_Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Deliver"/>
        <Class IRI="#Contact_&amp;_Document_Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Distributions_Processing"/>
        <Class IRI="#Private_Equity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Distributions_Processing_2"/>
        <Class IRI="#Aggregation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Distributions_Processing_Agg"/>
        <Class IRI="#Aggregation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Document_Importing"/>
        <Class IRI="#Capturing"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Document_Log"/>
        <Class IRI="#Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Document_Management"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Document_Profiling"/>
        <Class IRI="#Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Document_Scanning"/>
        <Class IRI="#Capturing"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#E-mail"/>
        <Class IRI="#Transfer"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Electronic_Commerce"/>
        <Class IRI="#Wholesale_/_Commercial_Banking_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Email"/>
        <Class IRI="#Payment_System_3"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Email_Archive"/>
        <Class IRI="#Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Email_MVICS"/>
        <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Email_Rules"/>
        <Class IRI="#Management"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Employee_Profile"/>
        <Class IRI="#Payroll"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Employee_Transfer,_Promotions_&amp;_Increments"/>
        <Class IRI="#Payroll"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Fees_and_Commissions"/>
        <Class IRI="#Fund_Investment_Management"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Financial_Accounting"/>
        <Class IRI="#Alternative_Investment_Systems"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Fund_Investment_Management"/>
        <Class IRI="#Private_Equity"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Hedge_Funds"/>
        <Class IRI="#Portfolio_/_Fund_Management_Systems"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Import_Transaction_History"/>
        <Class IRI="#Basic_Portfolio_Management_"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Information_Search"/>
        <Class IRI="#Data_Distribution_Systems"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Investment_Records_Management"/>
        <Class IRI="#Basic_Portfolio_Management_"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Investor_Registration"/>
        <Class IRI="#Hedge_Funds"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Investor_Reporting"/>
        <Class IRI="#Deal_Flow"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Knowledge_Delivery"/>
        <Class IRI="#Data_Distribution_Systems"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Leave_Management_System"/>
        <Class IRI="#Payroll"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#MS_Word_Integration"/>
        <Class IRI="#Deal_Flow"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Management"/>
        <Class IRI="#Contact_&amp;_Document_Management"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Management_Agg"/>
        <Class IRI="#Aggregation"/>
```

```
</SubClassOf>
<SubClassOf>
        <Class IRI="#Management_Fees"/>
        <Class IRI="#Hedge_Funds"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Management_Reporting"/>
        <Class IRI="#Private_Equity"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Mangement_2"/>
        <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Markers_and_Trend_Lines"/>
        <Class IRI="#Basic_Portfolio_Management_"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Market_Surveillance_and_Monitoring"/>
        <Class IRI="#Capital_Markets_Systems"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Marketing_Agg"/>
        <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Marketing_Tools"/>
        <Class IRI="#Electronic_Commerce"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Menu"/>
        <Class IRI="#Distributions_Processing_2"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Microfinance_/_Microbanking"/>
        <Class IRI="#Wholesale_/_Commercial_Banking_Systems"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Modelling"/>
        <Class IRI="#Fund_Investment_Management"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Modelling_MVICS"/>
        <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Modelling_Software"/>
        <Class IRI="#Data_Distribution_Systems"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Monitoring"/>
        <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Mutual_Funds_/_Unit_trusts"/>
        <Class IRI="#Asset_Management_Systems"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Navigation"/>
        <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Navigation_Agg"/>
        <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Notifications_&amp;_Approvals"/>
        <Class IRI="#Collaborating"/>
</SubClassOf>
```

```
<SubClassOf>
	<Class IRI="#Notifications_&amp;_Approvals_Agg"/>
	<Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Outlook_and_Email_Integration"/>
	<Class IRI="#Deal_Flow"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#PDF"/>
	<Class IRI="#Distributions_Processing_2"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#PDF_Creation"/>
	<Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#PDF_Creator"/>
	<Class IRI="#Deliver"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Payment_System"/>
	<Class IRI="#Electronic_Commerce"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Payment_System_3"/>
	<Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Payment_system_1"/>
	<Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Payment_system_2"/>
	<Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Payroll"/>
	<Class IRI="#Financial_Accounting"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Payroll_Management_System"/>
	<Class IRI="#Payroll"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Performance_Analytics"/>
	<Class IRI="#Fund_Investment_Management"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Planning"/>
	<Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Portfolio_/_Fund_Management_Systems"/>
	<Class IRI="#Asset_Management_Systems"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Portfolio_Management"/>
	<Class IRI="#Fund_Investment_Management"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Portfolio_Management_1"/>
	<Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Price_Alerts"/>
	<Class IRI="#Basic_Portfolio_Management_"/>
</SubClassOf>
<SubClassOf>
```

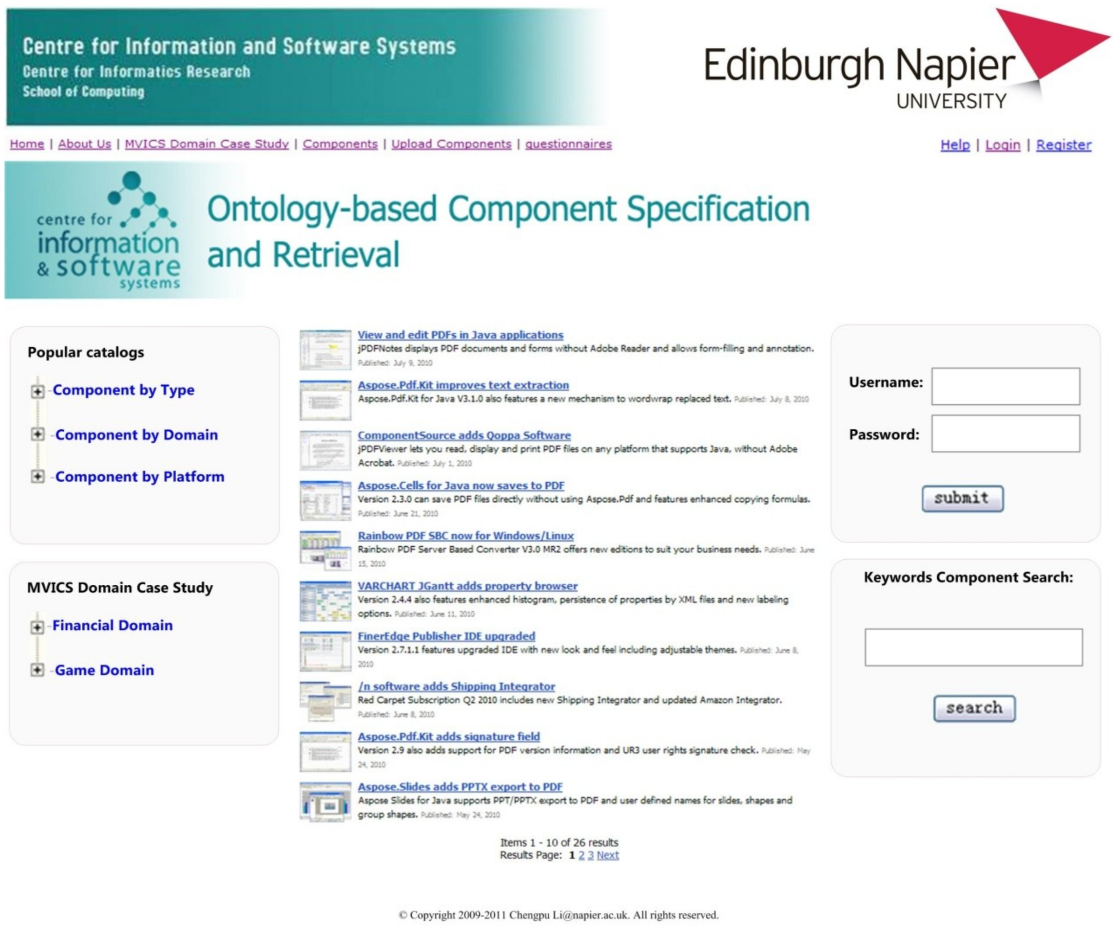
```
        <Class IRI="#Private_Equity"/>
        <Class IRI="#Alternative_Investment_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Product_Management"/>
        <Class IRI="#Electronic_Commerce"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Profile"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Purchasing_and_Transaction"/>
        <Class IRI="#Authorisation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Query_and_Analysis"/>
        <Class IRI="#Business_Intelligence"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Real_Estate_Investment_Management"/>
        <Class IRI="#Alternative_Investment_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Records_Management"/>
        <Class IRI="#Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Redemptions"/>
        <Class IRI="#Hedge_Funds"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Reference_Data_Systems"/>
        <Class IRI="#Data_Distribution_Systems"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Reimbursement_Management"/>
        <Class IRI="#Payroll"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Reporting"/>
        <Class IRI="#Transfer"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Reporting_MVICS"/>
        <Class IRI="#Payment_System_3"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Reporting_MVICS_Association_Class"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Retrieve_Transactions"/>
        <Class IRI="#Basic_Portfolio_Management_"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Revenue_Analytics"/>
        <Class IRI="#Fund_Investment_Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Risk_Analytics"/>
        <Class IRI="#Fund_Investment_Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Scheduling_and_Calendaring"/>
        <Class IRI="#Deal_Flow"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Search"/>
        <Class IRI="#Contact_&amp;_Document_Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Search_MVICS"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Search_and_navigation"/>
        <Class IRI="#Business_Intelligence"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Settlement"/>
        <Class IRI="#Card_Processing_/_Payments"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Settlement_Agg"/>
        <Class IRI="#Aggregation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Site_Navigation"/>
        <Class IRI="#Electronic_Commerce"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Statement"/>
        <Class IRI="#Aggregation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Subscriptions"/>
        <Class IRI="#Hedge_Funds"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Switching"/>
        <Class IRI="#Authorisation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Switching_Agg"/>
        <Class IRI="#Aggregation"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#System_Administration"/>
        <Class IRI="#Payment_System_3"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Task_Manager"/>
        <Class IRI="#Management"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Text_and_Word_Processing"/>
        <Class IRI="#Distributions_Processing_2"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Text_and_Word_Processing_MVICS"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Time_/_Date"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Time_Management_System"/>
        <Class IRI="#Payroll"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Toolbar"/>
        <Class IRI="#Distributions_Processing_2"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Tracking"/>
        <Class IRI="#Association_Class"/>
```

```
</SubClassOf>
<SubClassOf>
    <Class IRI="#Trade_Order_and_Risk_Management"/>
    <Class IRI="#Capital_Markets_Systems"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Transfer"/>
    <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Transfers_and_Defaults"/>
    <Class IRI="#Private_Equity"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#UserAdministration"/>
    <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#User_Administration"/>
    <Class IRI="#Payment_System_3"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#User_Administration_Agg"/>
    <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#User_Management"/>
    <Class IRI="#Electronic_Commerce"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Validating"/>
    <Class IRI="#Authorisation"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Version_Control"/>
    <Class IRI="#Management"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Version_Control_MVICS"/>
    <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Wealth_Management"/>
    <Class IRI="#Asset_Management_Systems"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Wealth_Planning"/>
    <Class IRI="#Wealth_Management"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Web_Interface"/>
    <Class IRI="#Collaborating"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Wholesale_/_Commercial_Banking_Systems"/>
    <Class abbreviatedIRI=":Thing"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Workflow"/>
    <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Workflow-Based_Task_Routing"/>
    <Class IRI="#Deal_Flow"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Workflow_Management"/>
    <Class IRI="#Association_Class"/>
</SubClassOf>
```

```
        <SubClassOf>
            <Class IRI="#Yield_Calculations"/>
            <Class IRI="#Basic_Portfolio_Management_"/>
        </SubClassOf>
        <DisjointClasses>
            <Class IRI="#Asset_Management_Systems"/>
            <Class IRI="#Data_Solutions"/>
        </DisjointClasses>
        <DisjointClasses>
            <Class IRI="#Asset_Management_Systems"/>
            <Class IRI="#Wholesale_/_Commercial_Banking_Systems"/>
        </DisjointClasses>
        <DisjointClasses>
            <Class IRI="#Card_Processing_/_Payments"/>
            <Class IRI="#Microfinance_/_Microbanking"/>
        </DisjointClasses>
        <DisjointClasses>
            <Class IRI="#Credit_Administration"/>
            <Class IRI="#Microfinance_/_Microbanking"/>
        </DisjointClasses>
        <DisjointClasses>
            <Class IRI="#Data_Solutions"/>
            <Class IRI="#Wholesale_/_Commercial_Banking_Systems"/>
        </DisjointClasses>
        <DisjointClasses>
            <Class IRI="#Electronic_Commerce"/>
            <Class IRI="#Microfinance_/_Microbanking"/>
        </DisjointClasses>
        <FunctionalObjectProperty>
            <ObjectProperty IRI="#IsA"/>
        </FunctionalObjectProperty>
        <TransitiveObjectProperty>
            <ObjectProperty IRI="#IsA"/>
        </TransitiveObjectProperty>
    </Ontology>
    <!-- Generated by the OWL API (version 3.2.3.22702) http://owlapi.sourceforge.net -->
```

Appendix C: The MVICS Project Homepage

C.1 The Homepage


Centre for Information and Software Systems
Centre for Informatics Research
School of Computing

Edinburgh Napier UNIVERSITY

Home | About Us | MVICS Domain Case Study | Components | Upload Components | questionnaires

Help | Login | Register

centre for information & software systems

Ontology-based Component Specification and Retrieval

Popular catalogs

- Component by Type
- Component by Domain
- Component by Platform

MVICS Domain Case Study

- Financial Domain
- Game Domain

View and edit PDFs in Java applications
jPDFNotes displays PDF documents and forms without Adobe Reader and allows form-filling and annotation. Published: July 9, 2010

Aspose.Pdf.Kit improves text extraction
Aspose.Pdf.Kit for Java V3.1.0 also features a new mechanism to wordwrap replaced text. Published: July 8, 2010

ComponentSource adds Qoppa Software
jPDFViewer lets you read, display and print PDF files on any platform that supports Java, without Adobe Acrobat. Published: July 1, 2010

Aspose.Cells for Java now saves to PDF
Version 2.3.0 can save PDF files directly without using Aspose.Pdf and features enhanced copying formulas. Published: June 21, 2010

Rainbow PDF SBC now for Windows/Linux
Rainbow PDF Server Based Converter V3.0 MR2 offers new editions to suit your business needs. Published: June 15, 2010

VARCHART JGantt adds property browser
Version 2.4.4 also features enhanced histogram, persistence of properties by XML files and new labeling options. Published: June 11, 2010

FinerEdge Publisher IDE upgraded
Version 2.7.1.1 features upgraded IDE with new look and feel including adjustable themes. Published: June 8, 2010

/n software adds Shipping Integrator
Red Carpet Subscription Q2 2010 includes new Shipping Integrator and updated Amazon Integrator. Published: June 8, 2010

Aspose.Pdf.Kit adds signature field
Version 2.9 also adds support for PDF version information and UR3 user rights signature check. Published: May 24, 2010

Aspose.Slides adds PPTX export to PDF
Aspose Slides for Java supports PPT/PPTX export to PDF and user defined names for slides, shapes and group shapes. Published: May 24, 2010

Items 1 - 10 of 26 results
Results Page: **1** 2 3 Next

Username:

Password:

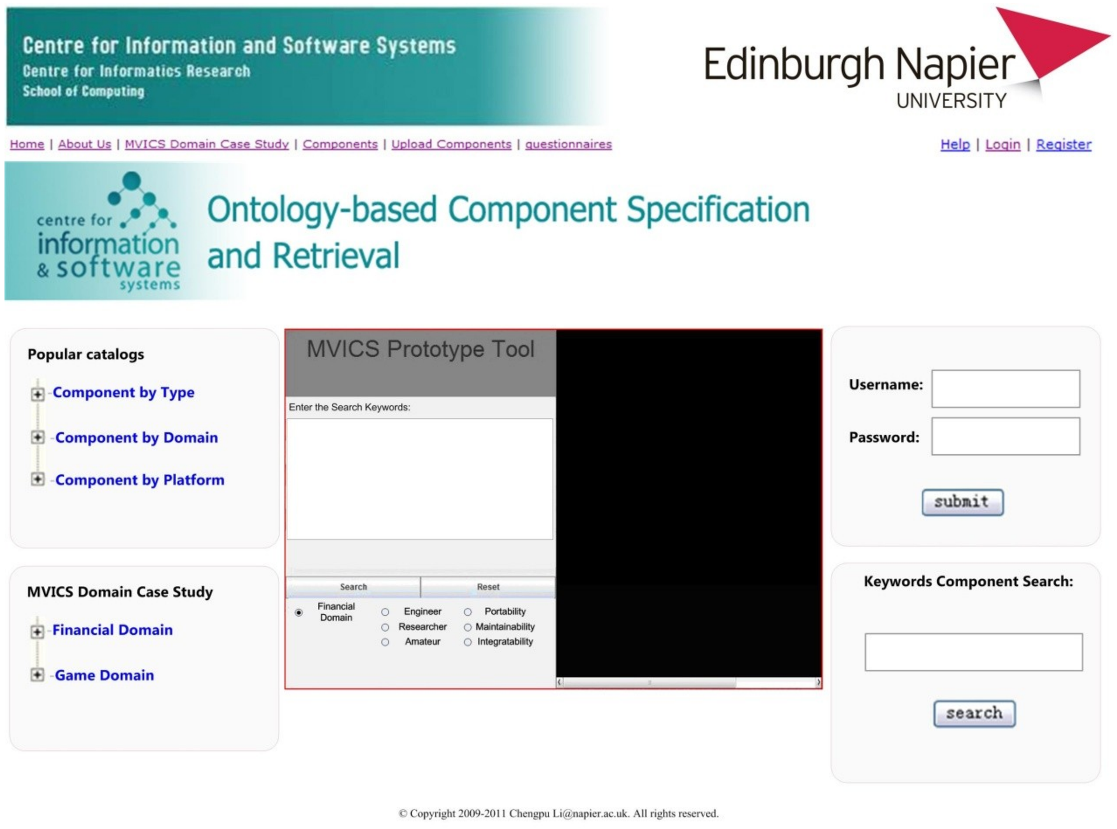
submit

Keywords Component Search:

search

© Copyright 2009-2011 Chengpu Li@napier.ac.uk. All rights reserved.

C.2 The Online Prototype Tool



Centre for Information and Software Systems
Centre for Informatics Research
School of Computing

Edinburgh Napier UNIVERSITY

Home | About Us | MVICS Domain Case Study | Components | Upload Components | questionnaires

Help | Login | Register

centre for information & software systems

Ontology-based Component Specification and Retrieval

Popular catalogs

- Component by Type
- Component by Domain
- Component by Platform

MVICS Domain Case Study

- Financial Domain
- Game Domain

MVICS Prototype Tool

Enter the Search Keywords:

Search | Reset

[✓ Financial Domain] Engineer Researcher Amateur Portability Maintainability Integratability

Username:

Password:

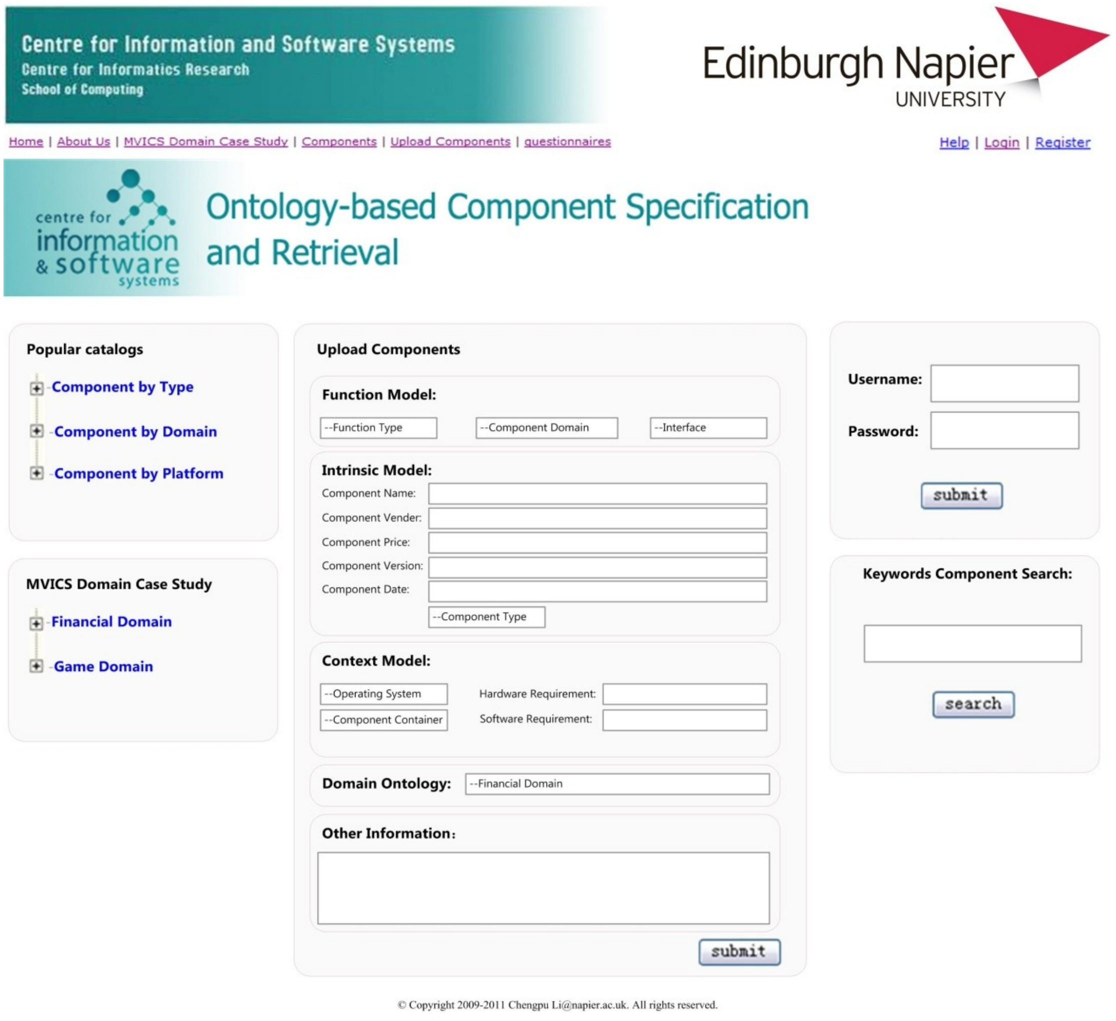
submit

Keywords Component Search:

search

© Copyright 2009-2011 Chengpu Li@napier.ac.uk. All rights reserved.

C.3 The MVICS-based Component Specification Form



Centre for Information and Software Systems
Centre for Informatics Research
School of Computing

Edinburgh Napier UNIVERSITY

Home | About Us | MVICS Domain Case Study | Components | Upload Components | questionnaires

Help | Login | Register

centre for information & software systems

Ontology-based Component Specification and Retrieval

Popular catalogs

- Component by Type
- Component by Domain
- Component by Platform

MVICS Domain Case Study

- Financial Domain
- Game Domain

Upload Components

Function Model: --Function Type | --Component Domain | --Interface

Intrinsic Model:

Component Name:
Component Vender:
Component Price:
Component Version:
Component Date:
--Component Type

Context Model:

--Operating System | Hardware Requirement:
--Component Container | Software Requirement:

Domain Ontology: --Financial Domain

Other Information:

submit

Username:

Password:

submit

Keywords Component Search:

search

© Copyright 2009-2011 Chengpu Li@napier.ac.uk. All rights reserved.

Appendix D: The OWL Doc of the RAG Ontology

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/ontologies/2011/6/Ontology1310792126375.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     ontologyIRI="http://www.semanticweb.org/ontologies/2011/6/Ontology1310792126375.owl">
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Declaration>
        <Class IRI="#Acceleration"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Action"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Action_Control"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Agg"/>
    </Declaration>
    <Declaration>
        <Class IRI="#All-Terrain_Vehicle"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Anti-decompilation"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Anti-modification"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Anti-plug_in"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Asphalt"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Ass"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Back"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Background"/>
```

```
</Declaration>
<Declaration>
    <Class IRI="#Background_Data_Processing"/>
</Declaration>
<Declaration>
    <Class IRI="#Background_Music"/>
</Declaration>
<Declaration>
    <Class IRI="#Building"/>
</Declaration>
<Declaration>
    <Class IRI="#Button_Dubbing"/>
</Declaration>
<Declaration>
    <Class IRI="#Car_Dubbing"/>
</Declaration>
<Declaration>
    <Class IRI="#Car_Model"/>
</Declaration>
<Declaration>
    <Class IRI="#Cinema"/>
</Declaration>
<Declaration>
    <Class IRI="#Citizen"/>
</Declaration>
<Declaration>
    <Class IRI="#City"/>
</Declaration>
<Declaration>
    <Class IRI="#City_Nature"/>
</Declaration>
<Declaration>
    <Class IRI="#Cloudy"/>
</Declaration>
<Declaration>
    <Class IRI="#Country_Nature"/>
</Declaration>
<Declaration>
    <Class IRI="#Countryside"/>
</Declaration>
<Declaration>
    <Class IRI="#Data_Processing_Agg"/>
</Declaration>
<Declaration>
    <Class IRI="#Database_Management"/>
</Declaration>
<Declaration>
    <Class IRI="#Database_Management_1"/>
</Declaration>
<Declaration>
    <Class IRI="#Day_Time"/>
</Declaration>
<Declaration>
    <Class IRI="#Day_and_Night"/>
</Declaration>
<Declaration>
    <Class IRI="#Desert"/>
</Declaration>
<Declaration>
    <Class IRI="#Desert_Plant_"/>
</Declaration>
<Declaration>
    <Class IRI="#Desert_Road"/>
</Declaration>
<Declaration>
    <Class IRI="#Dirt"/>
</Declaration>
<Declaration>
    <Class IRI="#Dubbing"/>
```

```
</Declaration>
<Declaration>
        <Class IRI="#F1"/>
</Declaration>
<Declaration>
        <Class IRI="#Farm"/>
</Declaration>
<Declaration>
        <Class IRI="#Farmhouse"/>
</Declaration>
<Declaration>
        <Class IRI="#Flat"/>
</Declaration>
<Declaration>
        <Class IRI="#Forward"/>
</Declaration>
<Declaration>
        <Class IRI="#Game_Control"/>
</Declaration>
<Declaration>
        <Class IRI="#Gymnasium"/>
</Declaration>
<Declaration>
        <Class IRI="#Hard_Court"/>
</Declaration>
<Declaration>
        <Class IRI="#Highway"/>
</Declaration>
<Declaration>
        <Class IRI="#Jungle_ATV"/>
</Declaration>
<Declaration>
        <Class IRI="#Left"/>
</Declaration>
<Declaration>
        <Class IRI="#Main_Road"/>
</Declaration>
<Declaration>
        <Class IRI="#Mall"/>
</Declaration>
<Declaration>
        <Class IRI="#Material"/>
</Declaration>
<Declaration>
        <Class IRI="#Motor_Home"/>
</Declaration>
<Declaration>
        <Class IRI="#Motorcycle"/>
</Declaration>
<Declaration>
        <Class IRI="#Mountain_ATV"/>
</Declaration>
<Declaration>
        <Class IRI="#Move"/>
</Declaration>
<Declaration>
        <Class IRI="#Night_Time"/>
</Declaration>
<Declaration>
        <Class IRI="#Path"/>
</Declaration>
<Declaration>
        <Class IRI="#Physical_Data_Control"/>
</Declaration>
<Declaration>
        <Class IRI="#Public_Facilities"/>
</Declaration>
<Declaration>
        <Class IRI="#Racing_Car"/>
```

```
</Declaration>
<Declaration>
	<Class IRI="#Racing_Field"/>
</Declaration>
<Declaration>
	<Class IRI="#Racing_Track"/>
</Declaration>
<Declaration>
	<Class IRI="#Rain"/>
</Declaration>
<Declaration>
	<Class IRI="#Right"/>
</Declaration>
<Declaration>
	<Class IRI="#SUV"/>
</Declaration>
<Declaration>
	<Class IRI="#Security"/>
</Declaration>
<Declaration>
	<Class IRI="#Security_Affair_Administration"/>
</Declaration>
<Declaration>
	<Class IRI="#Sence"/>
</Declaration>
<Declaration>
	<Class IRI="#Server_Speed_Control"/>
</Declaration>
<Declaration>
	<Class IRI="#Server_Speed_Optimization"/>
</Declaration>
<Declaration>
	<Class IRI="#Skyscraper"/>
</Declaration>
<Declaration>
	<Class IRI="#Snow"/>
</Declaration>
<Declaration>
	<Class IRI="#Sound"/>
</Declaration>
<Declaration>
	<Class IRI="#Special_Effects_Dubbing"/>
</Declaration>
<Declaration>
	<Class IRI="#Speed"/>
</Declaration>
<Declaration>
	<Class IRI="#Stock_Car"/>
</Declaration>
<Declaration>
	<Class IRI="#Stop"/>
</Declaration>
<Declaration>
	<Class IRI="#Street"/>
</Declaration>
<Declaration>
	<Class IRI="#Sunny"/>
</Declaration>
<Declaration>
	<Class IRI="#Truck"/>
</Declaration>
<Declaration>
	<Class IRI="#Ture"/>
</Declaration>
<Declaration>
	<Class IRI="#UI_Data_Processing"/>
</Declaration>
<Declaration>
	<Class IRI="#Villa"/>
```

```
</Declaration>
<Declaration>
      <Class IRI="#Weather"/>
</Declaration>
<Declaration>
      <Class IRI="#Width"/>
</Declaration>
<EquivalentClasses>
      <Class IRI="#Anti-decompilation"/>
      <Class IRI="#Security"/>
</EquivalentClasses>
<EquivalentClasses>
      <Class IRI="#Anti-modification"/>
      <Class IRI="#Security"/>
</EquivalentClasses>
<EquivalentClasses>
      <Class IRI="#Anti-plug_in"/>
      <Class IRI="#Security"/>
</EquivalentClasses>
<EquivalentClasses>
      <Class IRI="#Background_Data_Processing"/>
      <Class IRI="#Data_Processing_Agg"/>
</EquivalentClasses>
<EquivalentClasses>
      <Class IRI="#Data_Processing_Agg"/>
      <Class IRI="#UI_Data_Processing"/>
</EquivalentClasses>
<EquivalentClasses>
      <Class IRI="#Database_Management"/>
      <Class IRI="#Database_Management_1"/>
</EquivalentClasses>
<SubClassOf>
      <Class IRI="#Acceleration"/>
      <Class IRI="#Action_Control"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Action"/>
      <Class IRI="#Game_Control"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Action_Control"/>
      <Class IRI="#Game_Control"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Agg"/>
      <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#All-Terrain_Vehicle"/>
      <Class IRI="#Car_Model"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Anti-decompilation"/>
      <Class IRI="#Security_Affair_Administration"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Anti-modification"/>
      <Class IRI="#Security_Affair_Administration"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Anti-plug_in"/>
      <Class IRI="#Security_Affair_Administration"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Asphalt"/>
      <Class IRI="#Material"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Ass"/>
```

```
        <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Back"/>
        <Class IRI="#Move"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Background_Data_Processing"/>
        <Class IRI="#Physical_Data_Control"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Background_Music"/>
        <Class IRI="#Sound"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Building"/>
        <Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Button_Dubbing"/>
        <Class IRI="#Dubbing"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Car_Dubbing"/>
        <Class IRI="#Dubbing"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Car_Model"/>
        <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Cinema"/>
        <Class IRI="#Building"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Citizen"/>
        <Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#City"/>
        <Class IRI="#Sence"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#City_Nature"/>
        <Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Cloudy"/>
        <Class IRI="#Weather"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Country_Nature"/>
        <Class IRI="#Countryside"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Countryside"/>
        <Class IRI="#Sence"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Data_Processing_Agg"/>
        <Class IRI="#Agg"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Database_Management"/>
        <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Database_Management_1"/>
        <Class IRI="#Agg"/>
```

```
</SubClassOf>
<SubClassOf>
      <Class IRI="#Day_Time"/>
      <Class IRI="#Day_and_Night"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Day_and_Night"/>
      <Class IRI="#Action_Control"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Desert"/>
      <Class IRI="#Sence"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Desert_Plant_"/>
      <Class IRI="#Desert"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Desert_Road"/>
      <Class IRI="#Desert"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Dirt"/>
      <Class IRI="#Material"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Dubbing"/>
      <Class IRI="#Sound"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#F1"/>
      <Class IRI="#Racing_Car"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Farm"/>
      <Class IRI="#Countryside"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Farmhouse"/>
      <Class IRI="#Countryside"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Flat"/>
      <Class IRI="#Building"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Forward"/>
      <Class IRI="#Move"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Game_Control"/>
      <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Gymnasium"/>
      <Class IRI="#Building"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Hard_Court"/>
      <Class IRI="#Racing_Car"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Highway"/>
      <Class IRI="#Street"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Jungle_ATV"/>
      <Class IRI="#All-Terrain_Vehicle"/>
</SubClassOf>
```

```
<SubClassOf>
	<Class IRI="#Left"/>
	<Class IRI="#Ture"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Main_Road"/>
	<Class IRI="#Street"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Mall"/>
	<Class IRI="#Building"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Material"/>
	<Class IRI="#Racing_Track"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Motor_Home"/>
	<Class IRI="#Car_Model"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Motorcycle"/>
	<Class IRI="#Car_Model"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Mountain_ATV"/>
	<Class IRI="#All-Terrain_Vehicle"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Move"/>
	<Class IRI="#Action"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Night_Time"/>
	<Class IRI="#Day_and_Night"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Path"/>
	<Class IRI="#Street"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Physical_Data_Control"/>
	<Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Public_Facilities"/>
	<Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Racing_Car"/>
	<Class IRI="#Car_Model"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Racing_Field"/>
	<Class IRI="#Sence"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Racing_Track"/>
	<Class IRI="#Background"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Rain"/>
	<Class IRI="#Weather"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Right"/>
	<Class IRI="#Ture"/>
</SubClassOf>
<SubClassOf>
```

```
        <Class IRI="#SUV"/>
        <Class IRI="#All-Terrain_Vehicle"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Security"/>
        <Class IRI="#Ass"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Security_Affair_Administration"/>
        <Class abbreviatedIRI="owl:Thing"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Sence"/>
        <Class IRI="#Background"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Server_Speed_Control"/>
        <Class IRI="#Security_Affair_Administration"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Server_Speed_Optimization"/>
        <Class IRI="#Security_Affair_Administration"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Skyscraper"/>
        <Class IRI="#Building"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Snow"/>
        <Class IRI="#Weather"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Sound"/>
        <Class abbreviatedIRI="owl:Thing"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Special_Effects_Dubbing"/>
        <Class IRI="#Dubbing"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Speed"/>
        <Class IRI="#Action_Control"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Stock_Car"/>
        <Class IRI="#Racing_Car"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Stop"/>
        <Class IRI="#Action"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Street"/>
        <Class IRI="#City"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Sunny"/>
        <Class IRI="#Weather"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Truck"/>
        <Class IRI="#Car_Model"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Ture"/>
        <Class IRI="#Action"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#UI_Data_Processing"/>
        <Class IRI="#Physical_Data_Control"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Villa"/>
        <Class IRI="#Building"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Weather"/>
        <Class IRI="#Background"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Width"/>
        <Class IRI="#Racing_Track"/>
    </SubClassOf>
</Ontology>
<!-- Generated by the OWL API (version 3.2.3.22702) http://owlapi.sourceforge.net -->
```

Appendix E: The OWL Doc of the Gameloft Ontology

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/ontologies/2011/6/Ontology1310792126375.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     ontologyIRI="http://www.semanticweb.org/ontologies/2011/6/Ontology1310792126375.owl">
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Declaration>
        <Class IRI="#Acceleration"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Action"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Action_Control"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Aggregation"/>
    </Declaration>
    <Declaration>
        <Class IRI="#All-Terrain_Vehicle"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Animal"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Anti-decompilation"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Anti-modification"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Anti-plug_in"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Asphalt"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Association_Class"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Back"/>
    </Declaration>
```

```
<Declaration>
	<Class IRI="#Background"/>
</Declaration>
<Declaration>
	<Class IRI="#Background_Data_Processing"/>
</Declaration>
<Declaration>
	<Class IRI="#Background_Music"/>
</Declaration>
<Declaration>
	<Class IRI="#Building"/>
</Declaration>
<Declaration>
	<Class IRI="#Button_Dubbing"/>
</Declaration>
<Declaration>
	<Class IRI="#Car_Dubbing"/>
</Declaration>
<Declaration>
	<Class IRI="#Car_Model"/>
</Declaration>
<Declaration>
	<Class IRI="#Cartoon_Character"/>
</Declaration>
<Declaration>
	<Class IRI="#Character"/>
</Declaration>
<Declaration>
	<Class IRI="#Chart"/>
</Declaration>
<Declaration>
	<Class IRI="#Charting_and_Graphing_MVICS"/>
</Declaration>
<Declaration>
	<Class IRI="#Chess"/>
</Declaration>
<Declaration>
	<Class IRI="#Cinema"/>
</Declaration>
<Declaration>
	<Class IRI="#Citizen"/>
</Declaration>
<Declaration>
	<Class IRI="#City"/>
</Declaration>
<Declaration>
	<Class IRI="#City_Nature"/>
</Declaration>
<Declaration>
	<Class IRI="#Cloudy"/>
</Declaration>
<Declaration>
	<Class IRI="#Country_Nature"/>
</Declaration>
<Declaration>
	<Class IRI="#Countryside"/>
</Declaration>
<Declaration>
	<Class IRI="#Data_Display"/>
</Declaration>
<Declaration>
	<Class IRI="#Data_Processing_Agg"/>
</Declaration>
<Declaration>
	<Class IRI="#Database_Management"/>
</Declaration>
<Declaration>
	<Class IRI="#Database_Management_1"/>
</Declaration>
```

```
<Declaration>
	<Class IRI="#Day_Time"/>
</Declaration>
<Declaration>
	<Class IRI="#Day_and_Night"/>
</Declaration>
<Declaration>
	<Class IRI="#Desert"/>
</Declaration>
<Declaration>
	<Class IRI="#Desert_Plant_"/>
</Declaration>
<Declaration>
	<Class IRI="#Desert_Road"/>
</Declaration>
<Declaration>
	<Class IRI="#Diagram"/>
</Declaration>
<Declaration>
	<Class IRI="#Diagram_MVICS"/>
</Declaration>
<Declaration>
	<Class IRI="#Digit"/>
</Declaration>
<Declaration>
	<Class IRI="#Dirt"/>
</Declaration>
<Declaration>
	<Class IRI="#Dubbing"/>
</Declaration>
<Declaration>
	<Class IRI="#Else_Ethnic"/>
</Declaration>
<Declaration>
	<Class IRI="#Ethnic"/>
</Declaration>
<Declaration>
	<Class IRI="#F1"/>
</Declaration>
<Declaration>
	<Class IRI="#Farm"/>
</Declaration>
<Declaration>
	<Class IRI="#Farmhouse"/>
</Declaration>
<Declaration>
	<Class IRI="#Female"/>
</Declaration>
<Declaration>
	<Class IRI="#Flat"/>
</Declaration>
<Declaration>
	<Class IRI="#Forward"/>
</Declaration>
<Declaration>
	<Class IRI="#Game_Control"/>
</Declaration>
<Declaration>
	<Class IRI="#Gender"/>
</Declaration>
<Declaration>
	<Class IRI="#Graphic_Rendering"/>
</Declaration>
<Declaration>
	<Class IRI="#Gymnasium"/>
</Declaration>
<Declaration>
	<Class IRI="#Hard_Court"/>
</Declaration>
```

```
<Declaration>
	<Class IRI="#Highway"/>
</Declaration>
<Declaration>
	<Class IRI="#Human"/>
</Declaration>
<Declaration>
	<Class IRI="#Jungle_ATV"/>
</Declaration>
<Declaration>
	<Class IRI="#Left"/>
</Declaration>
<Declaration>
	<Class IRI="#Main_Road"/>
</Declaration>
<Declaration>
	<Class IRI="#Male"/>
</Declaration>
<Declaration>
	<Class IRI="#Mall"/>
</Declaration>
<Declaration>
	<Class IRI="#Material"/>
</Declaration>
<Declaration>
	<Class IRI="#Melanoderm"/>
</Declaration>
<Declaration>
	<Class IRI="#Meter"/>
</Declaration>
<Declaration>
	<Class IRI="#Motor_Home"/>
</Declaration>
<Declaration>
	<Class IRI="#Motorcycle"/>
</Declaration>
<Declaration>
	<Class IRI="#Mountain_ATV"/>
</Declaration>
<Declaration>
	<Class IRI="#Move"/>
</Declaration>
<Declaration>
	<Class IRI="#Night_Time"/>
</Declaration>
<Declaration>
	<Class IRI="#Path"/>
</Declaration>
<Declaration>
	<Class IRI="#Physical_Data_Control"/>
</Declaration>
<Declaration>
	<Class IRI="#Plot_Control"/>
</Declaration>
<Declaration>
	<Class IRI="#Poker"/>
</Declaration>
<Declaration>
	<Class IRI="#Public_Facilities"/>
</Declaration>
<Declaration>
	<Class IRI="#Puzzle_Character"/>
</Declaration>
<Declaration>
	<Class IRI="#Racing_Car"/>
</Declaration>
<Declaration>
	<Class IRI="#Racing_Field"/>
</Declaration>
```

```
<Declaration>
        <Class IRI="#Racing_Track"/>
</Declaration>
<Declaration>
        <Class IRI="#Rain"/>
</Declaration>
<Declaration>
        <Class IRI="#Regulation_Control"/>
</Declaration>
<Declaration>
        <Class IRI="#Right"/>
</Declaration>
<Declaration>
        <Class IRI="#SUV"/>
</Declaration>
<Declaration>
        <Class IRI="#Security"/>
</Declaration>
<Declaration>
        <Class IRI="#Security_Affair_Administration"/>
</Declaration>
<Declaration>
        <Class IRI="#Sence"/>
</Declaration>
<Declaration>
        <Class IRI="#Server_Speed_Control"/>
</Declaration>
<Declaration>
        <Class IRI="#Server_Speed_Optimization"/>
</Declaration>
<Declaration>
        <Class IRI="#Skyscraper"/>
</Declaration>
<Declaration>
        <Class IRI="#Snow"/>
</Declaration>
<Declaration>
        <Class IRI="#Sound"/>
</Declaration>
<Declaration>
        <Class IRI="#Special_Effects_Dubbing"/>
</Declaration>
<Declaration>
        <Class IRI="#Speed"/>
</Declaration>
<Declaration>
        <Class IRI="#Stock_Car"/>
</Declaration>
<Declaration>
        <Class IRI="#Stop"/>
</Declaration>
<Declaration>
        <Class IRI="#Street"/>
</Declaration>
<Declaration>
        <Class IRI="#Sunny"/>
</Declaration>
<Declaration>
        <Class IRI="#Truck"/>
</Declaration>
<Declaration>
        <Class IRI="#Ture"/>
</Declaration>
<Declaration>
        <Class IRI="#UI_Data_Processing"/>
</Declaration>
<Declaration>
        <Class IRI="#Uno"/>
</Declaration>
```

```
<Declaration>
        <Class IRI="#Villa"/>
</Declaration>
<Declaration>
        <Class IRI="#Visual_Effects"/>
</Declaration>
<Declaration>
        <Class IRI="#Weather"/>
</Declaration>
<Declaration>
        <Class IRI="#White_Race"/>
</Declaration>
<Declaration>
        <Class IRI="#Width"/>
</Declaration>
<Declaration>
        <Class IRI="#Yellow_Race"/>
</Declaration>
<EquivalentClasses>
        <Class IRI="#Anti-decompilation"/>
        <Class IRI="#Security"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Anti-modification"/>
        <Class IRI="#Security"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Anti-plug_in"/>
        <Class IRI="#Security"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Background_Data_Processing"/>
        <Class IRI="#Data_Processing_Agg"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Chart"/>
        <Class IRI="#Charting_and_Graphing_MVICS"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Data_Processing_Agg"/>
        <Class IRI="#UI_Data_Processing"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Database_Management"/>
        <Class IRI="#Database_Management_1"/>
</EquivalentClasses>
<EquivalentClasses>
        <Class IRI="#Diagram"/>
        <Class IRI="#Diagram_MVICS"/>
</EquivalentClasses>
<SubClassOf>
        <Class IRI="#Acceleration"/>
        <Class IRI="#Action_Control"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Action"/>
        <Class IRI="#Game_Control"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Action_Control"/>
        <Class IRI="#Game_Control"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Aggregation"/>
        <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#All-Terrain_Vehicle"/>
        <Class IRI="#Car_Model"/>
```

```
</SubClassOf>
<SubClassOf>
      <Class IRI="#Animal"/>
      <Class IRI="#Character"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Anti-decompilation"/>
      <Class IRI="#Security_Affair_Administration"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Anti-modification"/>
      <Class IRI="#Security_Affair_Administration"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Anti-plug_in"/>
      <Class IRI="#Security_Affair_Administration"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Asphalt"/>
      <Class IRI="#Material"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Back"/>
      <Class IRI="#Move"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Background"/>
      <Class IRI="#Graphic_Rendering"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Background_Data_Processing"/>
      <Class IRI="#Physical_Data_Control"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Background_Music"/>
      <Class IRI="#Sound"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Building"/>
      <Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Button_Dubbing"/>
      <Class IRI="#Dubbing"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Car_Dubbing"/>
      <Class IRI="#Dubbing"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Car_Model"/>
      <Class IRI="#Character"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Cartoon_Character"/>
      <Class IRI="#Character"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Character"/>
      <Class IRI="#Graphic_Rendering"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Chart"/>
      <Class IRI="#Data_Display"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Charting_and_Graphing_MVICS"/>
      <Class IRI="#Association_Class"/>
</SubClassOf>
```

```
<SubClassOf>
        <Class IRI="#Chess"/>
        <Class IRI="#Puzzle_Character"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Cinema"/>
        <Class IRI="#Building"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Citizen"/>
        <Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#City"/>
        <Class IRI="#Sence"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#City_Nature"/>
        <Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Cloudy"/>
        <Class IRI="#Weather"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Country_Nature"/>
        <Class IRI="#Countryside"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Countryside"/>
        <Class IRI="#Sence"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Data_Display"/>
        <Class IRI="#Graphic_Rendering"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Data_Processing_Agg"/>
        <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Database_Management"/>
        <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Database_Management_1"/>
        <Class IRI="#Aggregation"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Day_Time"/>
        <Class IRI="#Day_and_Night"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Day_and_Night"/>
        <Class IRI="#Action_Control"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Desert"/>
        <Class IRI="#Sence"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Desert_Plant_"/>
        <Class IRI="#Desert"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Desert_Road"/>
        <Class IRI="#Desert"/>
</SubClassOf>
<SubClassOf>
```

```
        <Class IRI="#Diagram"/>
        <Class IRI="#Data_Display"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Diagram_MVICS"/>
        <Class IRI="#Association_Class"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Digit"/>
        <Class IRI="#Data_Display"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Dirt"/>
        <Class IRI="#Material"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Dubbing"/>
        <Class IRI="#Sound"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Else_Ethnic"/>
        <Class IRI="#Ethnic"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Ethnic"/>
        <Class IRI="#Human"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#F1"/>
        <Class IRI="#Racing_Car"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Farm"/>
        <Class IRI="#Countryside"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Farmhouse"/>
        <Class IRI="#Countryside"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Female"/>
        <Class IRI="#Gender"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Flat"/>
        <Class IRI="#Building"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Forward"/>
        <Class IRI="#Move"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Game_Control"/>
        <Class abbreviatedIRI="owl:Thing"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Gender"/>
        <Class IRI="#Human"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Graphic_Rendering"/>
        <Class abbreviatedIRI="owl:Thing"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Gymnasium"/>
        <Class IRI="#Building"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#Hard_Court"/>
        <Class IRI="#Racing_Car"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Highway"/>
        <Class IRI="#Street"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Human"/>
        <Class IRI="#Character"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Jungle_ATV"/>
        <Class IRI="#All-Terrain_Vehicle"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Left"/>
        <Class IRI="#Ture"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Main_Road"/>
        <Class IRI="#Street"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Male"/>
        <Class IRI="#Gender"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Mall"/>
        <Class IRI="#Building"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Material"/>
        <Class IRI="#Racing_Track"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Melanoderm"/>
        <Class IRI="#Ethnic"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Meter"/>
        <Class IRI="#Data_Display"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Motor_Home"/>
        <Class IRI="#Car_Model"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Motorcycle"/>
        <Class IRI="#Car_Model"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Mountain_ATV"/>
        <Class IRI="#All-Terrain_Vehicle"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Move"/>
        <Class IRI="#Action"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Night_Time"/>
        <Class IRI="#Day_and_Night"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Path"/>
        <Class IRI="#Street"/>
</SubClassOf>
<SubClassOf>
        <Class IRI="#Physical_Data_Control"/>
        <Class abbreviatedIRI="owl:Thing"/>
```

```
</SubClassOf>
<SubClassOf>
      <Class IRI="#Plot_Control"/>
      <Class IRI="#Game_Control"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Poker"/>
      <Class IRI="#Puzzle_Character"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Public_Facilities"/>
      <Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Puzzle_Character"/>
      <Class IRI="#Character"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Racing_Car"/>
      <Class IRI="#Car_Model"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Racing_Field"/>
      <Class IRI="#Sence"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Racing_Track"/>
      <Class IRI="#Background"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Rain"/>
      <Class IRI="#Weather"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Regulation_Control"/>
      <Class IRI="#Game_Control"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Right"/>
      <Class IRI="#Ture"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#SUV"/>
      <Class IRI="#All-Terrain_Vehicle"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Security"/>
      <Class IRI="#Association_Class"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Security_Affair_Administration"/>
      <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Sence"/>
      <Class IRI="#Background"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Server_Speed_Control"/>
      <Class IRI="#Security_Affair_Administration"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Server_Speed_Optimization"/>
      <Class IRI="#Security_Affair_Administration"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#Skyscraper"/>
      <Class IRI="#Building"/>
</SubClassOf>
```

```
<SubClassOf>
	<Class IRI="#Snow"/>
	<Class IRI="#Weather"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Sound"/>
	<Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Special_Effects_Dubbing"/>
	<Class IRI="#Dubbing"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Speed"/>
	<Class IRI="#Action_Control"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Stock_Car"/>
	<Class IRI="#Racing_Car"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Stop"/>
	<Class IRI="#Action"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Street"/>
	<Class IRI="#City"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Sunny"/>
	<Class IRI="#Weather"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Truck"/>
	<Class IRI="#Car_Model"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Ture"/>
	<Class IRI="#Action"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#UI_Data_Processing"/>
	<Class IRI="#Physical_Data_Control"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Uno"/>
	<Class IRI="#Puzzle_Character"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Villa"/>
	<Class IRI="#Building"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Visual_Effects"/>
	<Class IRI="#Graphic_Rendering"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Weather"/>
	<Class IRI="#Background"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#White_Race"/>
	<Class IRI="#Ethnic"/>
</SubClassOf>
<SubClassOf>
	<Class IRI="#Width"/>
	<Class IRI="#Racing_Track"/>
</SubClassOf>
<SubClassOf>
```

```
        <Class IRI="#Yellow_Race"/>
        <Class IRI="#Ethnic"/>
    </SubClassOf>
</Ontology>
<!-- Generated by the OWL API (version 3.2.3.22702) http://owlapi.sourceforge.net -->
```