Application Architecture of Semantic QA System

Ying-Hong Wang[1], Cheng-Horng Liao[2], Shih-Hao Huang[3], and Wen-Nan Wang[4,*]

Department of Computer Science & Information Engineering

Tamkang University, Tamshui, Taipei County, Taiwan

[1]inhon@mail.tku.edu.tw, [2]jingo@cs.tku.edu.tw,

[3]793190074@s93.tku.edu.tw, [4]892190033@s92.tku.edu.tw

Received 15 June 2006; Revised 22 July 2006 ; Accepted 28 August 2006

Abstract. In this paper, we seek to use a Link Grammar Parser to develop a Semantic QA System. This system will make use of Link Grammar Parser to analyses the user's problems. After analyzing the learner's grammar, it will expand the Semantic dictionary via WordNet and subsequently seeks answer in the system's Ontology or searches the internet for more appropriate answers. In addition, it will provide http links to the answer as supplement information. The implementation environment will use Spring Framework as development platform. This research will be focused on the software system architecture. As it is foreseeable that the entire system will be sizable, and that education methods and QA system inevitably evolve throughout times, it is crucial that the design of this platform is easily extensible. We have two principles here, the system could be extensible and the components could be plug-in. In this paper, we defined all modules and use API to combine all modules by phases. All the modules should be implemented by interfaces, and provide appropriate service methods.

Keywords: Semantic QA, Semantic Strategy, Semantic API

1 Introduction

We aimed to develop a semantic question and to answer system base on Link Grammar Parser. This system is used to help end-user to learn e-learning courses effectively, and we tried to make a linkage between the learner and course. Because the popularization of java technology, such as: OOA (Object Oriented Analysis) and OOD (Object Oriented Design) are valued by software developers.

A major factor in the invention of Object-Oriented approach is to remove some of the flaws encountered with the procedural approach. In OOP, the data is treated as a critical element and does not allow that it flows freely. It bounds data closely to the functions that operate on it and protect it from accidental modification from outside functions. OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects. A major advantage of OOP is code reusability. Since the design of semantic QA system is complex, we tried to use OO approach to implement the system.

2 Related Works

2.1 OOA/OOD

OOA is the challenge for understanding the problem domain, and then the system's responsibilities in that light. For us, analysis is the study of a problem domain, leading to a specification of externally observable behavior; a complete, consistent, and feasible statement of what is needed; coverage of both functional and quantified operational characteristics (e.g. reliability, availability, performance).

The practice of taking a specification of externally available behavior and adding details needed for actual computer system implementation, including human interaction, task management, and data management details.

* Correspondence Author

2.2 Design Pattern

There are numbers of Design Patterns in the OO world. Here we are referring to the book "Design Patterns: Elements of Reusable Object-Oriented Software". Table 1 show that there are 23 patterns with three characteristics in this book: [1]

Table 1. 23 kinds of design patterns

Creation Patterns	Structural Patterns	Behavioral Patterns	
1. Abstract Factory	6. Adapter	13. Chain of Responsibility	20. State
2. Builder	7. Bridge	14. Command	21. Strategy
3. Factory Method	8. Composite	15. Interpreter	22. Template Method
4. Prototype Method	9. Decorator	16. Iterator	23. Visitor
5. Singleton	10. Façade	17. Mediator	
	11. Flyweight	18. Memento	
	12. Proxy	19. Observer	

In software engineering, a design pattern is a general repeatable solution to a commonly-occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code; it is a description or template for how to solve a problem that can be used in many different situations. Object-oriented design patterns typically show the relationships and interactions between classes or objects without specifying the final application classes or objects that are involved. Algorithms are not thought of as design patterns since they solve computational problems rather than design problems.

2.3 Spring Framework

Developing software applications is hard enough even with good tools and technologies. Implementing applications using platforms which promise everything but turn out to be heavy-weight, hard to control and not very efficient during the development cycle to make it even harder. Spring provides a light-weight solution for building enterprise-ready applications, while still supporting the possibility of using declarative transaction management, remote access to your logic using RMI or web services, mailing facilities and various options in persisting your data to a database.

Spring [2] provides an MVC framework, transparent ways of integrating AOP into your software and a well-structured exception hierarchy including automatic mapping from proprietary exception hierarchies. Spring could potentially be a one-stop-shop for all your enterprise applications; however, Spring is modular, allowing you to use parts of it, without having to bring in the rest. You can use the bean container, with Struts on top, but you could also choose to just use the Hibernate integration or the JDBC abstraction layer. Spring is non-intrusive, meaning dependencies on the framework are generally none or absolutely minimal, depending on the area of use.

Spring contains a lot of functionality and features, which are well-organized in seven modules shown in the figure 1. This section discusses each one of modules in turn.

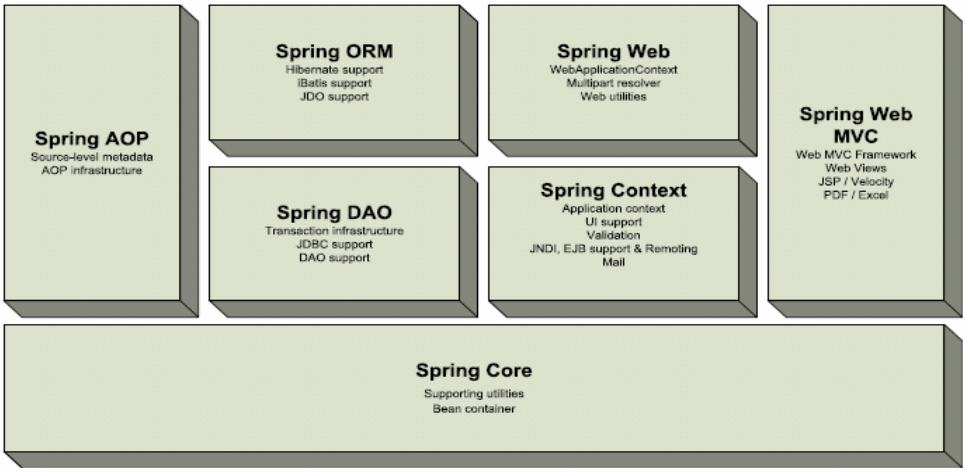

Fig. 1. Modules of Spring Frameworks

The Core package is the most fundamental part of the framework and provides the Dependency Injection features allowing you to manage bean container functionality. The basic concept here is the BeanFactory, which provides a factory pattern removing the need for programmatic singletons and allows you to decouple the configuration and specification of dependencies from your actual program logic.

On top of the Core package sits the Context package, providing a way to access beans in a framework-style manner, somewhat resembles a JNDI-registry. The context package inherits its features from the beans package and adds support for text messaging using e.g. resource bundles, event-propagation, resource-loading and transparent creation of contexts, for example, a servlet container.

The DAO package provides a JDBC-abstraction layer that removes the need to do tedious JDBC coding and parsing of database-vendor specific error codes. Also, the JDBC package provides a way to do programmatic as well as declarative transaction management, not only for classes implementing special interfaces, but for all your POJOs (plain old java objects). The ORM package provides integration layers for popular object-relational mapping APIs, including JDO, Hibernate and iBatis.

Using the ORM package you can use all those O/R-mappers in combination with all the other features Spring offers, like simple declarative transaction management mentioned before Spring's AOP package provides an AOP Alliance compliant aspect-oriented programming implementation allowing you to define. For example, method-interceptors and pointcuts cleanly decouple code implementing functionality that should logically speaking be separated. Using source-level metadata functionality you can incorporate all kinds of behavioral information into your code, a little like .NET attributes.

2.4 Link Grammar

The Link Grammar Parser [3] is a syntactic parser of English, based on link grammar, an original theory of English syntax. Given a sentence, the system assigns to it a syntactic structure, which consists of a set of labeled links connecting pairs of words. The parser also produces a "constituent" representation of a sentence (showing noun phrases, verb phrases, etc.).

The parser has a dictionary of about 60000 word forms. It has coverage of a wide variety of syntactic constructions, including many rare and idiomatic ones. The parser is robust; it is able to skip over portions of the sentence that it cannot understand, and assigns some structure to the rest of the sentence. It is able to handle unknown vocabulary, to make intelligent guesses from context and to spell about the syntactic categories of unknown words. It has knowledge of capitalization, numerical expressions, and a variety of punctuation symbols.

2.5 WordNet and Ontology

WordNet [4] is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. WordNet was developed by the Cognitive Science Laboratory at Princeton University under the direction of Professor George A.Miller.

Ontology [5] is used to describe structural Domain Knowledge. It maintains relationships between knowledges. We call this knowledge “node”, and actually it is out persistence for whole system. We could use several ways to implementation the ontology that we want, such as RDF, DAML [6], KAON [7], and so on.

3 System Architecture Design

We defined four main modules for the Semantic QA system shown as figure 2. There are four system models in the system architecture:

(1). SemQA Service acts as the service center.

(2). SentenceGrammarParser Service uses Link Grammar to process most sentence grammar checks to distinguish the relationship between words and sentence grammar structure

(3). KeywordExtractionProcessor carries out explanation and extension of the vocabulary to be aimed at the keywords of existing Ontology.

(4). Ontology Service: we choose the Data Structure from many distance learning courses as the target courses to set up the system Ontology. The Ontology Service is designed to find the semantic answer from the ontology according to the relevant keyword generated by model 2 and 3.

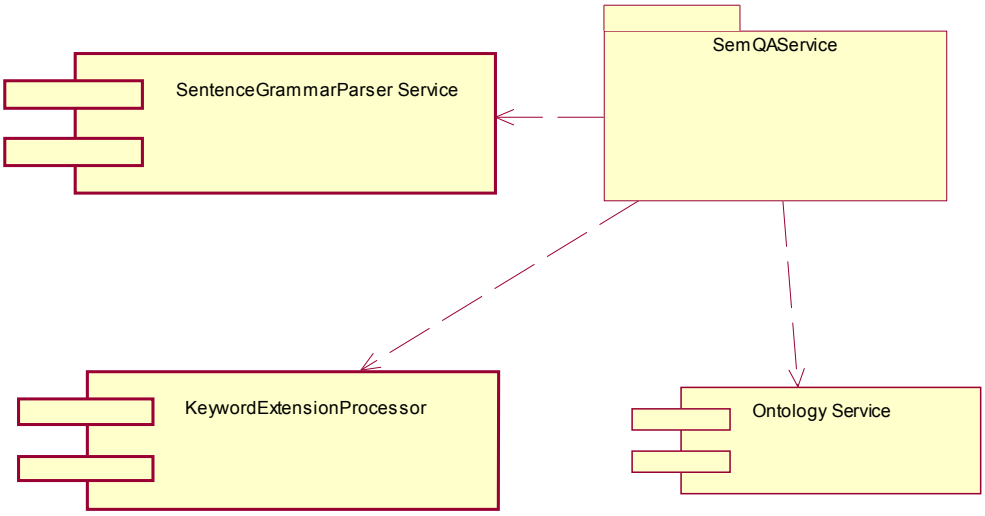

Fig. 2. System Modules of Semantic QA System

Following by the principle, "The Single Responsibility," we keep only one reason to modify the module. Every single module should take one and only one business responsibility.

Following "The Interface Segregation Principle," we implement every module by the interface that we defined. Each interface just exports the needed parts to other modules; we hide the complex parts from the end users. One famous design pattern is used to describe like "Façade."

According to the principle, whole system is defined as four main services which are: SemQAService, SentenceGrammarService, KeyWordExtensionProcessor and OntologyService. We use Spring IoC Container [8] to declare these four modules into XML files.

SemQAService is the façade of Semantic QA System; Figure 3 shows this concept by the class diagram.

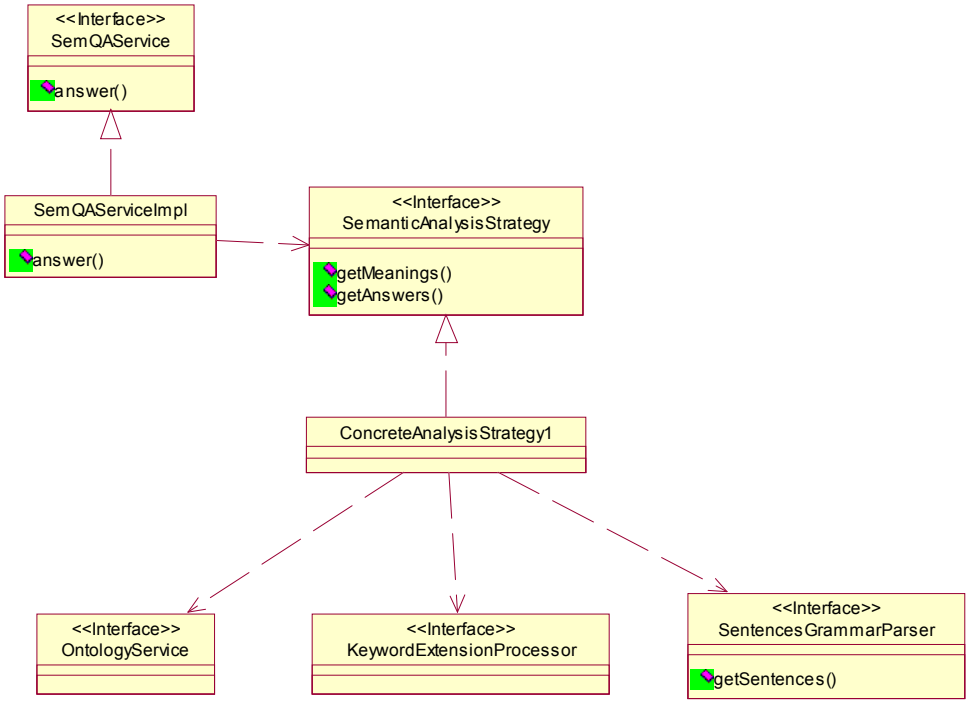

Fig. 3. Semantic QA System Class Diagram

We use Model-View-Controller pattern for our web framework, and this interface, SemQAService, separates the view and model. Viewers call the model by this interface. It means we don't keep business login into any view module. By the same way we may have various view implementations here, it may be web based or client-server. Figure 4 shows that we provide various kinds of API to access SemQAService, which includes RMI, Web Services or local java API.

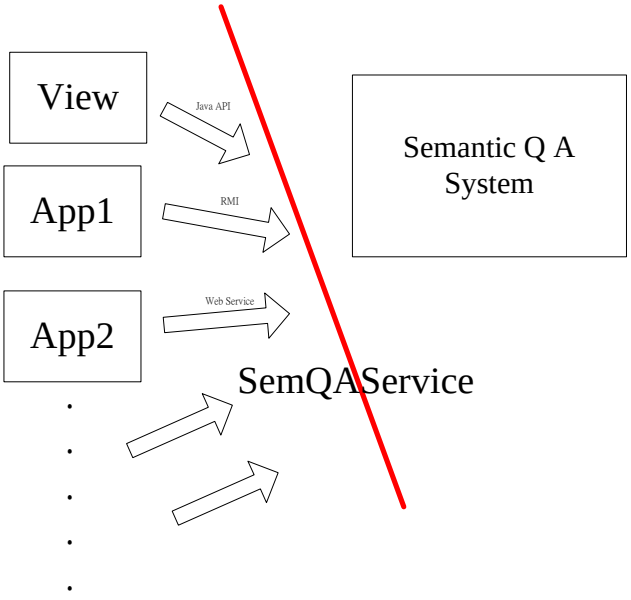

Fig. 4. API for Semantic QA System

And all the system activities of SemQA Service are listed as follows:
(1). To get question sentence from user
(2). To parses the sentence and get its grammar.
(3). To get related words by verbs and nouns.
(4). For semantic analysis.
(5). To find answers.
(6). To response answer to user.

And then we use Link Grammar to implement SentencesGrammarParser. It builds a constituent tree after parse the sentences; the figure 5 shows the conceptual classes of this class diagram.

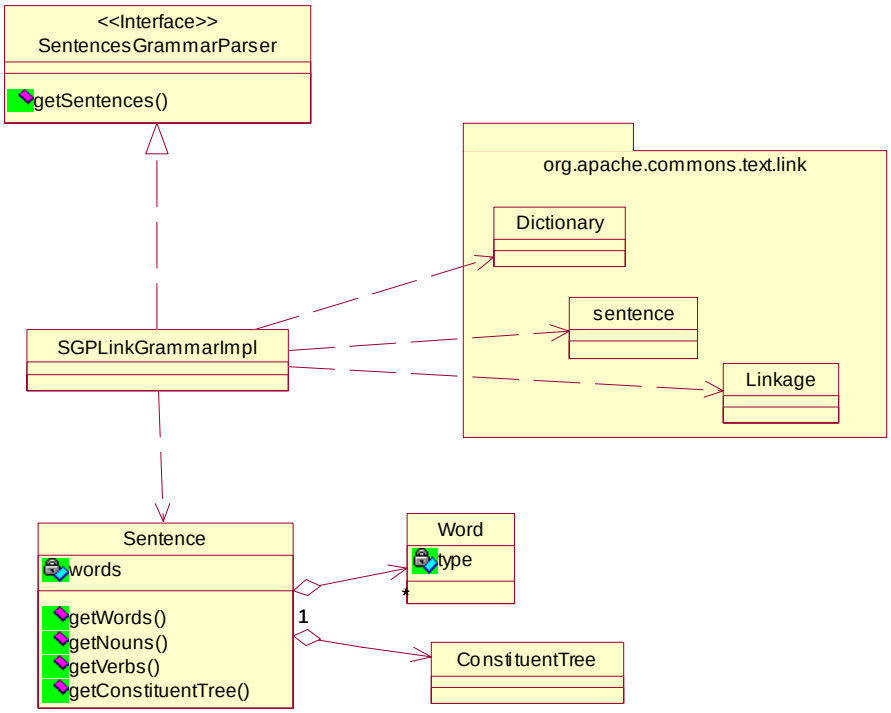

Fig. 5. Conceptual class diagram of SentenceGrammarParser

KeywordExtensionProcessor is used to get more information from the sentences, and it can help us to clarify the appropriate semantic. Here we use WordNet to implement. Figure 6 shows the conceptual class diagram.

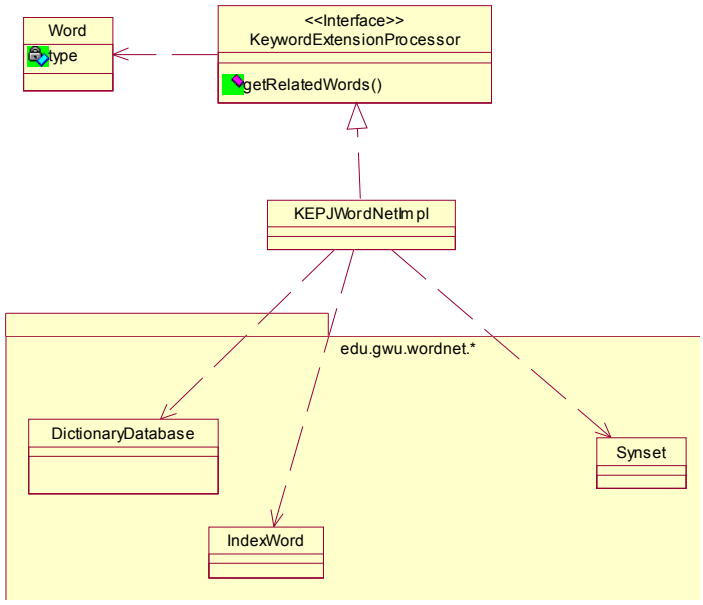

Fig. 6. KeyWordExtensionProcessor

OntologyService is used to describe domain knowledge. It provides the answer for us. Here we show the conceptual class diagram by Figure 7.

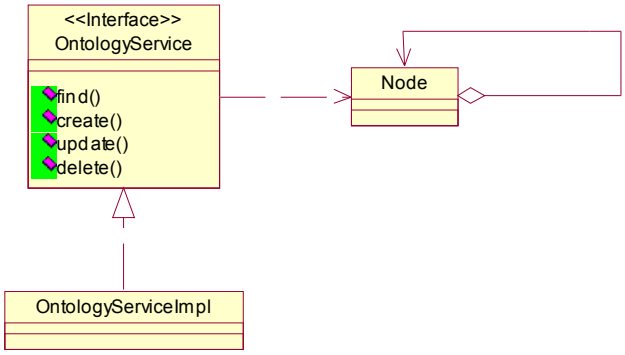

Fig. 7. OntologyService

4 System Implementation

To implement the system design described in session 3, we use SUN J2EE to develop the whole system. We use Spring Framework to build each parts of the system. And the system deployed by several XML files, the main XML file is applicationContext.xml which is main architecture of all system. It is showed as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

<bean id="qaService"
class="tw.edu.tku.tkse.elearning.semqa.model.service.impl.SemQAServiceImp
l" />

</beans>
```

All the other modules of this system will be implemented by the same way. This also shows the relationships in a XML file.

5 Conclusions

We developed a Semantic QA system here. This system assists e-learning system to become more like a real world teaching and learning environment. Since it is not the last day of e-learning developing, we do not build a final system here, instead, we design a system that prepare for changes. It should be extensible and flexible and it can be easy to plug-in. Here we use Spring Framework as its framework, and we provide several API for end-use and other application platform. We didn't do too much in the phase, but we define some clear modules at the beginning which will help to shape the direction of future development.

6 Acknowledgement

This research was funded by the National Science Council of the Republic of China under the Contracts No: NSC-NSC 94-2520-S-032-003 and NSC 95-2520-S-032-001. The authors also appreciate the instrument support of TamKang Software Engineering Group at TamKang University, Taiwan, ROC.

References

[1] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addson-Wesley Professional, 1995.

[2] Rod Johnson, Juergen Hoeller, Alef Arendsen, Colin Sampaleanu, Darren Davison, Dmitriy Kopylenko, Thomas Risberg and Mark Pollack, "Spring – java/j2ee Application Framework Reference Documentation," Version 1.1.2, *http://www.springframework.org/docs/spring-reference.pdf*.

[3] D. Sleator, D. Temperley, and J. Lafferty, "Link Grammar," *http://www.link.cs.cmu.edu/link/*.

[4] G. Miller, "WordNet, a lexical database for the English language," *http://www.cogsci.princeton.edu/~wn/*.

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

[6] The Defense Advanced Research Projects Agency, "The DARPA Agent Markup Language," *http://www.daml.org/*, 2000.

[7] FZI and AIFB, "KAON is an open-source ontology management infrastructure targeted for business applications," *http://kaon.semanticweb.org.*

[8] M. Fowler, "Inversion of Control Containers and the Dependency Injection pattern," *http://www.martinfowler.com/articles/injection.html*, 2004.