ISSN: 1945-3116(Print), 1945-3124(Online) Volume 2, Number 1, April 2009

Journal of Software Engineering and Applications

Chief Editor : Dr. Ruben Prieto-Diaz

www.scirp.org/journal/jsea

Scientific Research Publishing

Journal Editorial Board

http://www.scirp.org/journal/jsea

Editor-in-Chief

Dr. Ruben Prieto-Diaz	James Madison University, USA

Editorial Board (According to Alphabet)

Dr. Jiannong Cao	Hong Kong Polytechnic University, China
Dr. Zonghua Gu	Hong Kong University of Science and Technology, China
Dr. Keqing He	Wuhan University, China
Dr. Mingzhi Mao	SUN YAT-SEN University, China
Dr. Kasi Periyasamy	University of Wisconsin-La Crosse, La Crosse, USA
Dr. Juergen Rilling	Concordia University, Canada
Dr. Philip Sheu	University of California at Irvine, USA
Dr. Jian Wang	Chinese Academy of Sciences, China
Dr. Shi Ying	Wuhan University, China
Dr. Mark A. Yoder	Electrical and Computer Engineering, USA

Editorial Assistant

Xiaoqian Qi	Wuhan University, Wuhan, China

Guest Reviewers (According to Alphabet)

Harry Agius
Paul Ashford
Aladdin Ayesh
Guo Bo
Riadh Dhaou
Dawei Ding
K.L. Edwards
Omar Elkeelany
Jun-Bong Eom
Zhi Fang
Ivan G. Guardiola
Kwan Hee Han
Chul Kim
Min-Sung Kim
Fang Liu
Hao Ma
Giorgio Di Natale
Haruhiko Ogasawara
Silvia Pfeiffer
Mahmudur Rahman
Yoan Shin
Joseph Y. H. So
Janusz Stoklosa
Shirshu Varma
Shuenn-Shyang Wang
Simon Wu
Tai-Chi Wu
Cholatip Yawut
Wei Zhang

J. Software Engineering & Applications, **2009, 2**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

CONTENTS

Volume 2 Number 1 April 2009

Journal of Software Engineering and Applications (JSEA)

Journal Information

SUBSCRIPTIONS

The *Journal of Software Engineering and Applications* (Online at Scientific Research Publishing, www.SciRP.org) is published quarterly by Scientific Research Publishing, Inc. 5005 Paseo Segovia, Irvine, CA 92603-3334, USA.

E-mail: jsea@scirp.org

Subscription rates: Volume 2 2009
Print: $50 per copy.
Electronic: free, available on www.SciRP.org.
To subscribe, please contact Journals Subscriptions Department, E-mail: jsea@scirp.org

Sample copies: If you are interested in subscribing, you may obtain a free sample copy by contacting Scientific Research Publishing, Inc. at the above address.

SERVICES

Advertisements
Advertisement Sales Department, E-mail: jsea@scirp.org

Reprints (minimum quantity 100 copies)
Reprints Co-ordinator, Scientific Research Publishing, Inc. 5005 Paseo Segovia, Irvine, CA 92603-3334, USA.
E-mail: jsea@scirp.org

COPYRIGHT

Copyright© 2009 Scientific Research Publishing, Inc.

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as described below, without the permission in writing of the Publisher.

Copying of articles is not permitted except for personal and internal use, to the extent permitted by national copyright law, or under the terms of a license issued by the national Reproduction Rights Organization.

Requests for permission for other kinds of copying, such as copying for general distribution, for advertising or promotional purposes, for creating new collective works or for resale, and other enquiries should be addressed to the Publisher.

Statements and opinions expressed in the articles and communications are those of the individual contributors and not the statements and opinion of Scientific Research Publishing, Inc. We assume no responsibility or liability for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained herein. We expressly disclaim any implied warranties of merchantability or fitness for a particular purpose. If expert assistance is required, the services of a competent professional person should be sought.

PRODUCTION INFORMATION

For manuscripts that have been accepted for publication, please contact:
E-mail: jsea@scirp.org

J. Software Engineering & Applications, **2009, 2:1-12**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

Design Pattern Representation for Safety-Critical Embedded Systems

Ashraf Armoush*, Falk Salewski*, Stefan Kowalewski*

*Embedded Software Laboratory, RWTH Aachen University, Aachen, Germany
Email: {armoush, salewski, kowalewski}@embedded.rwth-aachen.de

Received January 9th, 2009; revised February 12th, 2009; accepted March 25th, 2009.

ABSTRACT

Design Patterns, which give abstract solutions to commonly recurring design problems, have been widely used in the software and hardware domain. As non-functional requirements are an important aspect in the design of safety-critical embedded systems, this work focuses on the integration of non-functional implications in an existing design pattern concept. We propose a pattern representation for safety-critical embedded application design methods by including fields for the implications and side effects of the represented design pattern on the non-functional requirements of the overall systems. The considered requirements include safety, reliability, modifiability, cost, and execution time.

Keywords: *Design Pattern*, *Embedded Systems*, *Non-Functional Requirements*, *Safety-Critical Systems*

1. Introduction

Design pattern, originally proposed in [1] by the architect Christopher Alexander, is a universal approach to describe common solutions to widely recurring design problems. Ever since, this concept has been applied in several domains of hardware design (electronics) and also became popular in the software domain after the success of the book Design Patterns: Element of Reusable Object-Oriented Software by Gama *et al.* [2].

As the concept of design pattern aims at supporting designers and system architects in their choice of suitable solutions for commonly recurring design problems, this concept might also be useful to support the design of safety-critical embedded systems. The design of these systems is considered to be a complex process, as hardware and software components have to be considered during the design as well as potential interactions between hardware and/or software components. Moreover, not only functional requirements[1] have to be fulfilled by these systems. Failures in safety-critical systems could result in critical situations that may lead to serious injury or loss of life or unacceptable damage to the environment. Therefore, also the non-functional requirement *safety* has to be considered in these systems to assure that the risk of hazards is acceptable low in the considered system. To support the design of safe devices, safety measures are given by international safety standards as the IEC61508 [3]. Beside life cycle and process requirements, also different measures for the design of software and hardware components are recommended. These safety measures have typically an impact on the cost, the reliability, the real-time behavior of the system, and on the modifiability of the resulting system. Depending on the application domain of the later embedded system, these non-functional requirements are of great importance. For this reason, non-functional requirements should be considered during the design of any safety-critical system.

While current concepts of design pattern exist for many different application domains, they typically lack a consideration of potential side effects on non-functional requirements. In order to integrate these side effects into the pattern concept, we propose an extended template for an effective design pattern representation for safety-critical applications. This pattern representation includes the traditional pattern concept in combination with an extension describing the implications and side effects with respect to the non-functional requirements. While this concept has been described briefly in [4] before, this work focuses on the application of our approach. Thus, two example patterns are included to illustrate the proposed representation of design patterns for software and hardware components in safety-critical applications.

2. Related Work

The field of design pattern is large and still rapidly growing. Many researches have focused on the use of design pattern in the software domain [2,5,6,7,8,9], but further research is still needed in the domain of safety-critical embedded systems to integrate the non-functional requirements in design patterns. In his books [10] and [11], Bruce Douglass proposed several design patterns

[1]Note: functional requirements (FR) represent the required functionality itself while non-functional requirements (NFR) describe additional qualities that have to be achieved by the developed system (e.g. safety, reliability, modifiability, cost and execution time)

for safety-critical systems based on well known fault tolerant design methods and by integrating some modification to increase the safety level on these patterns. Gross and Yu [12] discuss the relationship between non-functional requirements and design patterns, and propose a systematic approach for evaluating design patterns with respect to non-functional requirements. They propose the use of design patterns for establishing traces between non-functional goals in a goal tree such as a soft goal interdependency graph (SIG) and the system design. Cleland-Huang *et al.* [13,14] enhance the patterns defined by Gross and Yu [12] through defining a model for establishing traceability between certain types of non-functional requirements and design and code artifacts, through the use of design patterns as intermediary objects. Xu *et al.* [15] classified the dependability needs into three types of requirements and proposed an architectural pattern that allows requirements engineers and architects to map dependability requirements into three corresponding types of architectural components. Konord *et al.* [16,17] describe a research of how the principle of design pattern can be applied to requirements specifications, which they term requirements patterns for embedded systems. They include a constraints field in the pattern template to show the functional and non-functional restrictions that are applied to the system.

In comparison to our work, none of the aforementioned approaches show clearly the implications on the non-functional requirements as part of the pattern. These patterns and the other developed patterns focus on the traditional structure of the pattern that includes: context, problem and solution. The use of non-functional requirements in these approaches is restricted to the requirement analysis phase of the design process. In these approaches, neither a relative measure nor an indication for the implications of the patterns on the non-functional requirements, were given. To improve these approaches, we propose a new template representation in Section 4 to show the implications of the represented patterns on the non-functional requirements.

3. Design Pattern Template

In this section, the template pattern we propose for the representation of design patterns for safety-critical embedded applications is described. As depicted in Figure 1, the upper part of the template includes the traditional representation of a design pattern while a listing of the pattern implications on the non-functional requirements is given in the *Implication* section. Moreover, further support is given by stating implementation issues, summarizing the consequences and side effects as well as a listing of related patterns.

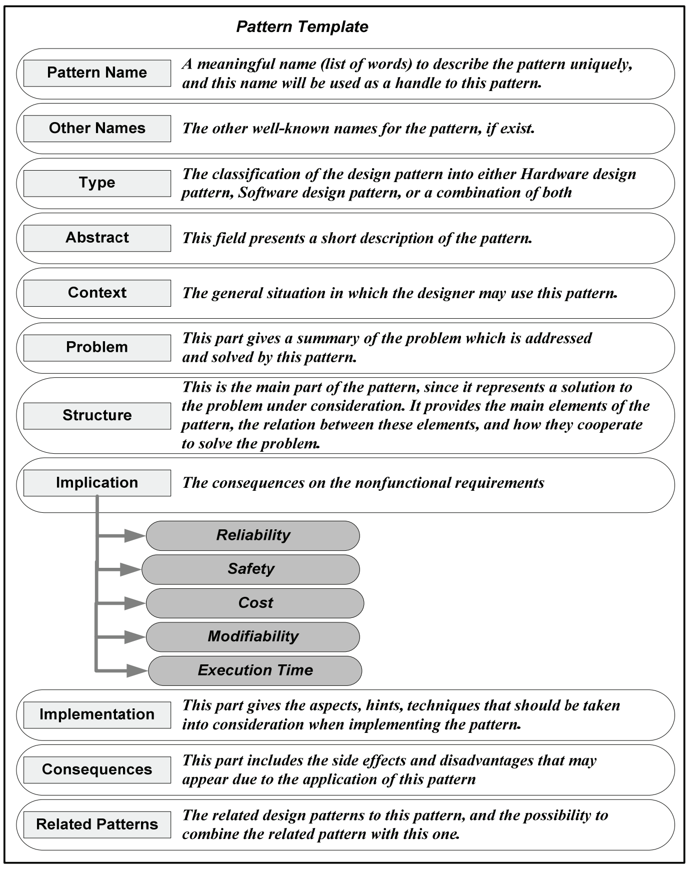

Pattern Template
Pattern Name
A meaningful name (list of words) to describe the pattern uniquely, and this name will be used as a handle to this pattern.
Other Names
The other well-known names for the pattern, if exist.
Type
The classification of the design pattern into either Hardware design pattern, Software design pattern, or a combination of both
Abstract
This field presents a short description of the pattern.
Context
The general situation in which the designer may use this pattern.
Problem
This part gives a summary of the problem which is addressed and solved by this pattern.
Structure
This is the main part of the pattern, since it represents a solution to the problem under consideration. It provides the main elements of the pattern, the relation between these elements, and how they cooperate to solve the problem.
Implication
The consequences on the nonfunctional requirements
Reliability
Safety
Cost
Modifiability
Execution Time
Implementation
This part gives the aspects, hints, techniques that should be taken into consideration when implementing the pattern.
Consequences
This part includes the side effects and disadvantages that may appear due to the application of this pattern
Related Patterns
The related design patterns to this pattern, and the possibility to combine the related pattern with this one.

Figure 1. The design pattern template

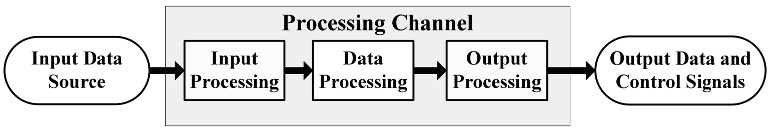


Figure 2. Basic system without specfic safety requirements

The proposed design template includes a part for pattern implications on the non-functional requirements reliability, safety, cost, modifiability and execution time. To allow a suitable description of these implications, the changes/improvements of using the corresponding design pattern are represented relative to a basic simple system (see Figure 2). This basic system has a given reliability (R_{old}), a given cost, a given modifiability and is resulting in a given executing time. Moreover, this basic system has no specific safety measurements.

4. The Implication On Non-Functional Requirements

While the main part of the design pattern proposed does not differ from well known approaches [18,19,20,21], the part for the implications on the non-functional requirements is described in this section. As mentioned above, the implications are stated relative to the basic system without any specific safety method. In the following, the determination of the five implications on non-functional requirements is described:

Reliability: In this context, reliability is defined as the probability that of a system or component to perform its required functions correctly under stated conditions for a specified period of time. This part of implications describes the relative improvement in the system's reliability relative to the maximum possible improvement[2] in reliability, which is defined in the following equation:

$$\text{Reliability Improvement}=\frac{R_{new}-R_{old}}{1-R_{old}}\times 100\% \qquad (1)$$

R_{new}: The reliability after using this pattern.

R_{old}: The reliability of the basic system.

Safety: The safety of a system is usually determined by the residual risk of operating this system (see e.g. [3]). Therefore, the notion of risk can be used as a measure for the assessment of safety-critical systems. The problem concerning design patterns is that they describe an abstract solution to a commonly recurring design problem. As it is not related to a specific application or to a specific case, it is difficult to determine an actual value for the possible residual risk without considering a concrete application. To allow an indication of the safety that can be achieved by the application of a specific design pattern, existing recommendations given in safety standards are used. In detail, it is stated to which Safety Integrity Level (SIL) the pattern is recommended in a given safety standard. The safety integrity levels used here include the levels SIL1 to SIL4 as they are defined in the standard IEC61508 [3]. Additionally, the notation SIL0 is used in this template to describe a system without specific safety requirements. If measures are described in design patterns that are not included in current safety standards, these measures have to be assessed in an appropriate manner, e.g. by comparing them to measures with known recommendations.

***Cost*:** The implications on costs include:

- The recurring cost per unit, which reflects the additional costs resulting from additional or specific hardware components required by the design pattern.
- The development cost of applying this pattern.

***Modifiability*:** This implication describes the degree to which the system developed according to this design pattern can be modified and changed.

***Impact on execution time*:** With this implication, the effect of the pattern on the total time of execution at runtime is indicated. It shows the execution time overhead that is resulting from the application of this pattern in the worst and the average cases.

The application of the design pattern proposed, especially the use of the implication part introduced briefly in this section, is described in form of two example patterns in the following section.

5. Example Patterns

Two example patterns are presented in this section to illustrate the application of the proposed approach: The first pattern is a hardware and software pattern that is expected to be suitable for complex and highly safety-critical systems (Safety Executive Pattern). The second pattern is a hybrid[3] software fault tolerance method intended to increase the reliability of the standard N-version programming approach (Acceptance Voting Pattern).

5.1 EXAMPLE 1

In this example pattern, the pattern originally described in [10] is presented in our extended pattern representation including also implications on non-functional requirements.

Pattern Name
Safety Executive Pattern (SEP)

Other Names
Safety Kernel Pattern

[2]Note: the maximum possible improvement in reliability is the difference between the reliability of the basic system and the maximum value for reliability which is equal to 1 (Ideal case without failures).

[3]Note: A pattern is called *hybrid* if the pattern is composed of at least two other patterns.

Type

Hardware and Software

Abstract

The Safety Executive Pattern can be considered as an extension of the Watchdog Pattern[4] targeting the problem that a shutdown of the system by the actuation channel itself might be critical in the presence of faults (shutdown might fail or take too long). This problem occurs especially in those systems in which a complicated series of steps involving several components is necessary to reach a fail-safe state. Therefore, the Safety Executive Pattern uses a watchdog in combination with an additional safety executive component, which is responsible for the shutdown of the system as soon as the watchdog sends a shutdown signal (see also Figure 3. The safety executive pattern). If the system has a safe state, the actuation channel is shutdown via the safety executive component. Otherwise, the safety executive component has to delegate all actuations necessary to an additional fail-safe processing channel.

Context

The application of this pattern is suitable in the following context:

• The considered actuation channel requires a risk reduction by safety measures.

• The considered system has at least one safe state. If this is not the case, an additional fail-safe processing channel has to be applied to overtake necessary actions.

• A shutdown of the actuation channel is complex. As an example, this is the case if several safety-related system actions have to be controlled simultaneously.

• A sufficient determination of failures in the actuation channel can be achieved by a watchdog.

Problem

Provision of a centralized and consistent method for monitoring and controlling the execution of a complex safety measure (shutdown or switch over to redundant unit in case of failures).

Pattern Structure

The Safety Executive Pattern is based on an actuation channel to perform the required functionality and an optional fail-safe processing channel that is dedicated to the execution and control of the fail-safe processing (see also Figure 3). The central part of this pattern is the existence of a centralized safety executive component coordinating all safety-measures required to shut down the system or to switch over to the fail-safe processing channel. The safety executive component can also be used to control multiple actuation channels in the system that each may have multiple channels.

The components of the pattern depicted in Figure 3 are described below:

• Input Data Source: This component represents the source of information that is used as input to the system under consideration. Typically, this data comes either from the system user or from external sensors that are used to monitor environmental variables such as: temperature, pressure, speed, light, etc...

• Actuator(s): Actuators are the physical devices that perform the action of the channel like: motors, switches, heaters, signals, or any other device that performs a specific action. Often, there are more than one actuator in a single channel.

• Actuation Channel: This channel represents a subsystem that performs dedicated tasks in the overall system by taking an input data from the input data source, performs some transformation on this data, and then uses the results to generate suitable commands to drive the actuators.

• Fail-Safe Processing Channel: This is an optional channel; it is dedicated to the execution and control of the fail-safe processing. In the presence of a fault in the actuation channel, the safety executive turns off the actuation channel, and the fail-safe channel takes over. If the System doesn't have a fail-Safe Channel, then the actuation channels must have at least one reachable safe states.

• Data Acquisition (Input Processing): This part of the channel collects the raw data from the input data source and may perform some data formatting or transfor- mations.

• Data Processing (Transformation): This part may contain multiple data transformation components; where each one performs a single transformation or processing on the received data to execute the desired algorithm in order to generate the required control signals. The final component of this part sends the computed output to the output processing unit.

• Output Processing: This unit takes the computed data from the data transformation unit and generates the final data and the control signals to drive the actuators. It can be considered as a device driver for the actuator.

• Integrity Check (Optional): This is an optional component that is invoked by the watchdog to run a periodic Built-In Test (BIT) to verify all or a portion of the internal functionality of the actuation channel.

• Time Base: This is an independent timing source (timing circuit) that is used to drive the watchdog. This time source has to be separate from the one used to drive the actuation channel.

• Watchdog (WD): The watchdog receives liveness messages (strokes) from the components of the actuation channel in a predefined timeframe. If a stroke comes too late or out of sequence, the watchdog considers this situation as a fault in the actuation channel and it issues a shutdown signal to the actuation channel or initiates a

[4] Note: The *Watchdog Pattern* is based on two components, the actuation channel and a supervisor, called watchdog. The actuation channel typically triggers the watchdog at defined time intervals to demonstrate that the actuation channel is still active. More advanced approaches include more sophisticated interactions between the actuation channel and the watchdog to allow a higher degree of fault coverage (see e.g. [11]).

corrective action through sending a command signal to the optional integrity check. If the system contains multiple actuation channels, then it may contain multiple watchdogs, one per actuation channel.

• Safety Executive: This is the main component in this safety executive pattern. It tracks and coordinates all safety monitoring to ensure the execution of safety action when appropriate. It consists of a safety coordinator that controls safety measures and safety policies. The safety executive component captures the shutdown signal from the watchdog in the case of failure in the actuation channel.

• Safety Coordinator: The safety coordinator is used to control and coordinate the safety processing that is managed by the safety measures. It also executes the control algorithms that are specified by the safety policies.

• Safety Measures: Include the detailed description of the safety measures. The safety coordinator may control multiple safety measures.

• Safety Policies: Each safety policy specifies a strategy or control algorithm for the safety coordinator. It involves a complicated sequence of steps that involve multiple safety measures. The reason for the separation of the coordinator from the safety policies is to make the process of changing and adapting a safety policy easier.

Implication

This section describes the implication of this pattern relative to the basic system without a specific safety method.

• **Reliability**

Let us have the following notations:

R_{AC}: the reliability of the actuation channel. ($R_{old} = R_{AC}$)

R_{SC}: the reliability of the fail-safe processing channel.

R_{SE}: the reliability of the safety executive component.

C: the coverage factor which is defined as: the probability that a fault in an actuation channel will be identified by the safety executive and the fail-safe processing channel will be activated.

Assume that the watchdogs are carefully designed with reliability≈1.

The safety executive pattern will continue to work without system failure as long as one of the following two conditions holds:

◎ There is no fault in the *actuation channel*.

◎ There is a fault in the *actuation channel* and the watchdog detects this fault and the *safety executive* initiates a shutdown or activates the fail-safe processing channel.

The new reliability after using this pattern (R_{new}) is equal to:

$$R \ = R_{AC} + CR_{SE}(1 - R_{AC})R_{SC} \qquad (2)$$

In this equation, the first term represents the reliability of the actuation channel while the second term represents the reliability of the remaining parts in the case of failure in the actuation channel.

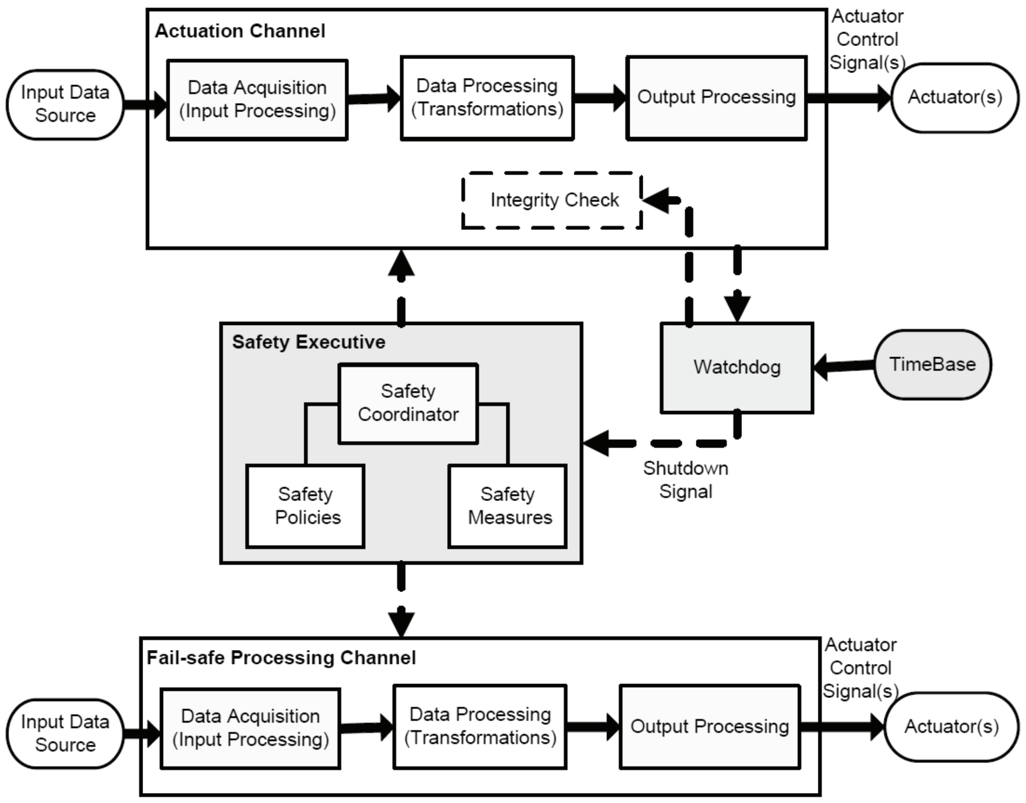


Figure 3. The safety executive pattern

The percentage improvement in reliability relative to the maximum possible improvement is equal to:

$$= \frac{R_{AC} + CR_{SE}(1 - R_{AC})R_{SC} - R_{AC}}{1 - R_{AC}} \times 100\% \qquad (3)$$

$$= CR_{SE}R_{SC} \times 100\% \qquad (4)$$

• Safety**:**

The safety executive pattern includes the following four design techniques: program sequence monitoring with a watchdog, test by redundant hardware (the watchdog that initiates the integrity check and BITs), safety bag techniques[5], and graceful degradation[6]. According to the standard IEC 61508 [3], the recommendation for these techniques is shown Table 1.

In general, we think that the combination of these techniques and the development cost makes the safety executive pattern suitable and highly recommended only for very high critical applications with high safety integrity levels (SIL4 and SIL3) and recommended for lower levels (SIL2 and SIL 1).

• **Cost:**

This pattern is an expensive pattern with very high cost since it consists of different components that involve high recurring and development cost.

○ Recurring cost: It includes the cost of the following:

◆ The actuation channel.

◆ The fail-safe processing channel (if present).

◆ The safety executive component.

◆ Watchdogs and their independent timing source.

○ Development cost: In general, the development cost for this pattern is very high since it includes a development of three different systems (channels) that include different architectures and different designs.

• **Modifiability:**

There are two types of possible modifications:

1) Actuation Channel: It is very simple to modify this pattern by adding extra functionality to the actuation channel. The only things that should be done: is to know whether the new components need to send stroke messages to the watchdog.

2) Safety policy: One of the main features of this pattern is the centralized safety processing which is performed by the Safety Executive Component. The Safety Executive separates the coordinator from the safety policies to simplify the change and modification of the safety policy and to make it easier.

[5]Note: A safety bag is an external monitor, implemented on an independent hardware component to ensure that the system performs safe actions [3].

[6]Note: The safety degradation is a technique that gives priorities to the various functions to be carried out by the system. The design ensures that if there are insufficient resources to carry out all the system functions, the higher priority functions (the safety functions in our case) are carried out in preference to the lower once [3].

[7]Note: a processing unit is a (programmable) electronic device executing a certain function. Typical examples are programmable logic controllers, microcontrollers, and FPGAs. Moreover, an electronic device might include more than one processing unit (e.g. multi-core architectures). In this case, the analysis of the independence between these units requires special care.

Table 1. Recommendations for safety integrity levels

Techniques	SIL1	SIL2	SIL3	SIL4
Program sequence monitoring (WD)	HR	HR	HR	HR
Test by redundant hardware	R	R	R	R
Safety bag techniques	—	R	R	R
Graceful degradation	R	R	HR	HR

• **Impact on execution time:**

The actuation channel and the safety executive have different CPUs and different memories, and they run simultaneously in parallel. Thus, there is no effect for the safety executive component on the actuation channel during the normal operation of the system except the execution of the periodic built in tests (BITs).

Implementation

The following points should be taken into consideration during the implementation of this pattern:

• The actuation channel, the safety executive, and the fail-safe processing channel run separately in parallel, therefore each channel will run on its own processing unit[7] and own memory.

• The safety-critical information must be protected against data corruption, e.g. by using CRCs or any other method to detect data errors.

• The watchdog component is simple and often implemented as a separate hardware device. It is capable of detecting a variety of hardware and software fault. However, its actual diagnostic coverage depends on the *i*ntegrity check implemented in the actuation channel.

• To provide protection from faults in a common time base, separate timing sources must be used for the watchdog, the safety executive and the actuation channel.

Consequences and Side Effects

The main drawback of this pattern is the high complexity of this pattern for implementation. Therefore it is used for complex and highly safety-critical systems.

Related Patterns

The safety executive pattern is used for complex safety-critical applications and it covers a large set of features, provided by of the other patterns, such as sequence monitoring provide by watchdog, switch-to-backup as in the fail-safe channel. For simpler systems with simpler safety requirements, other simpler patterns, such as Watchdog pattern, Sanity Check pattern and Monitor Actuator pattern [11], can be used.

5.2 EXAMPLE 2

This example pattern describes the pattern originally described in [22] including the standard pattern components as well as implications on non-functional requirements.

Pattern Name

Acceptance Voting Pattern

Other Names

Type
Software

Abstract
The Acceptance Voting pattern is a hybrid pattern that incorporates the *N-Version Programming Pattern* with *Acceptance Tests* (AT) used by the *Recovery Block Pattern*. Similar to the normal N-version programming approach, this pattern is based on two or more diverse software versions. Traditionally, these versions are functionally equivalent and are developed by independent teams from the same initial specification [23]. Moreover, approaches to increase the diversity of the resulting software versions could be applied already in the specification (functional diversity, see e.g. [40]). In case of this pattern, the output of each version is presented to an acceptance test to determine if the output is reasonable. The outputs that pass the acceptance test are used as inputs to a dynamic voter, which is executed to produce the correct output according to a specific voting scheme. Depending on the applications, the diverse software versions can be executed on parallel hardware or sequentially on a single hardware device.

Context
The application of this pattern is suitable in the following context:

• Tolerance of software faults is required for safety reasons (acceptance test)

• High reliability of the system's output is required (several software versions)

• The correctness of the results delivered by the diverse software versions can be checked by an acceptance test.

• The development of diverse software versions is possible (additional development costs, additional organizational effort, and sufficient number of developers for the development of each version).

Problem
Enable the tolerance of software faults that may remain after the software development to target safety and reliability requirements.

Pattern Structure
The Acceptance Voting Pattern (AV) is a hybrid pattern, which represents an extension of the N-version programming approach by incorporating this approach with the acceptance test used in the recovery block approach. This pattern includes N diverse software versions that are typically executed in parallel to perform the required task. The output of each version is tested for correctness using an acceptance test and only those results that pass the acceptance test are used by the voting algorithm to generate the final result. The goal of the Acceptance Voting pattern is to increase the system's reliability through a combination of a fault detection scheme provided by the acceptance test and a fault masking scheme provided by N-version programming with voting.

The structure of this pattern is shown in Figure 4 and the function of each component is described below:

• Input Data Source: This instance represents the source of information that is used as input to the designed system. Typically, this data comes either from a system user or from external sensors used to monitor environmental variables such as temperature, pressure, speed, or light.

• Output Data and Control Signals: The output data of the voter module represents the final output data of the designed system. This data may contain control signals to activate actuators as motors, switches, heaters, or messages for other components outside the system.

• Version 1, 2...N: These are diverse software versions implemented to fulfill the specified functionality. Typically, these versions result from an independent implementation by independent teams of software developers, based on the same initial specification. Thus, these versions are performing roughly the same functionality on the input data to produce the final result. Further aspects of generating these diverse software versions can be found in the *implementation section* below. Usually, these diverse versions are executed in parallel on different hardware devices to generate ***N*** outputs and each of these outputs is presented to an acceptance test to check them for correctness. Those results that pass the acceptance test are processed by the dynamic voter module to determine the output data by applying a specific voting strategy.

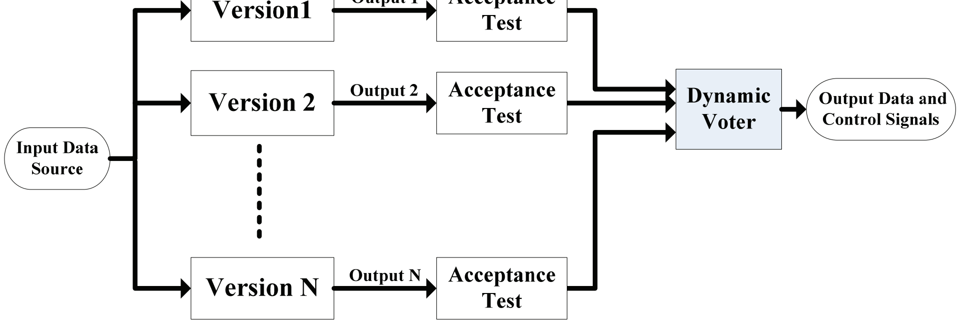


Figure 4. The acceptance voting pattern

• Acceptance Test (AT): This part of the software is executed on the outcome of each version to confirm that the result is reasonable and fulfills defined requirements given in the software specification. The acceptance test returns either true or false and may have several components. Moreover, it may include checks for runtime errors [24] and mechanisms for implicit error detection. Various implementations of the acceptance test are possible ranging from simple reasonableness checks to complex high-coverage validators [25].

The reliability of the system's output depends greatly on the quality of the acceptance test (especially an acceptance test that detects faults in a correct output reduces the system's reliability). Thus, this test should be carefully designed and it is desirable that the acceptance test is simple as well as easy to verify.

• Dynamic Voter: The voter reads the outputs that pass the acceptance test and uses these results as inputs to the voting algorithm in order to determine the final output and control signals. The voter that is used in this pattern should be dynamic[8] due to the variable number of inputs that ranges from 0 to N. Depending on the number of outputs that pass the acceptance test, the voter may include the following different actions:

− In the case when no output passes the acceptance test, it reports an overall system failure.

− In the case when one output passes the acceptance test, it just forwards this output.

− In the case when two outputs pass the acceptance test, the signal is only forwarded if both are equal (or the difference is within a defined tolerance). In the case of inequality, the action depends on the required level of safety and reliability, either an output is selected according to a predefined order or an exception is raised to indicate a failure.

− When the number of outputs that pass the acceptance test is more than two, a voting technique is executed to generate the final result.

Several voting techniques exist that can be used for voting as majority voting (most commonly used technique), consensus voting [26], and maximum likelihood voting [27]. The selection of these techniques depends on the type of data, the deviation in the outputs of the versions, the type of agreement [28], the output space cardinality size[9], the functionality of the voter [29], the reliability of the different versions, and perhaps even further factors.

[8]Note: A voter is considered as *dynamic* if it accepts varying numbers of input signals.

[9]Note: The cardinality size of an output space is the number of possible different values for an output.

[10]Note: While the assumption of failure independence is not realistic for practical software implementations, this assumption eases the calculation presented. The simplified calculation presented here already allows certain reliability evaluations. Moreover, a dependency term could be included into the calculations if an explicit consideration of dependencies between the software versions should be considered. Further aspects of software diversity can be found in the *implementation section* of this pattern description.

Implication

This section describes the implication of this pattern using a majority voting approach relative to the basic system without any specific safety method.

• **Reliability:**

Let us have the following notations:

f: the probability of failure in a version due to a bug in its implementation.

E: the event that the output of a version is erroneous. ($P\{E\}=f$)

T: the event that the acceptance test reports that the output is wrong.

N: the total number of different independent versions.

n: the number of versions that pass the acceptance test.

m: the agreement number which is equal to $\lceil (n+1)/2 \rceil$ for the majority voting.

P_{TP}: the probability that a version will pass the acceptance test, given that the outcome of a version is correct. (True Positive case) ($P_{TP} = P\{\overline{T} \mid \overline{E}\}$).

P_{FN}: the probability that a version will fail the acceptance test, given that the outcome of a version is correct. (False Negative case) ($P_{FN} = P\{T \mid \overline{E}\} = 1 - P_{TP}$).

P_{TN}: the probability that a version will fail the acceptance test, given that the outcome of a version is wrong. (True Negative case) ($P_{TN} = P\{T \mid E\}$).

P_{FP}: the probability that a version will pass the acceptance test, given that the outcome of a version is wrong. (False Positive case) ($P_{FP} = P\{\overline{T} \mid E\} = 1 - P_{TN}$).

R_{old}: the reliability of system software with single version ($R_{old} = R$).

R_{new}: the reliability of system software with acceptance voting pattern.

Assumptions:

− The voter is carefully designed and can be considered as fault free.

− The majority voting technique is used in the voter software component.

− The failures in the different versions are statically independent[10] and the different versions have the same probability of failure (f) and the same reliability ($R_i = R$). At any given time, if the number of versions' outputs that pass the acceptance test and participate the voting is n, then these outputs can be grouped into two parts:

− Correct Outputs (True Positive outputs that pass AT) with probability=$R*P_{TP}$.

− Incorrect Outputs(False Positive outputs that contain undetected faults) with probability = $(1-R)\cdot(1-P_{TN}) = (1-R)\cdot P_{FP}$.

The probability that an output passes the test is equal to:

$$P\{\overline{T}\} = RP_{TP} + (1-R)P_{FP} \tag{5}$$

The probability that an output does not pass the test is equal to:

$$p\{T\} = RP_{FN} + (1-R)P_{TN} \tag{6}$$

The Probability that the voter gives a correct output, given that n outputs passed the test, is equal to

$$= \sum_{i=m}^{n} \binom{n}{i} (RP_{TP})^i [(1-R)P_{FP}]^{n-i} \tag{7}$$

The probability that n outputs from the total number of outputs pass the acceptance test and give a correct result in the majority voting is:

$$= \left[\binom{N}{n} \left(\sum_{i=m}^{n} \binom{n}{i} (RP_{TP})^i [(1-R)P_{FP}]^{n-i} \right) (RP_{FN} + (1-R)P_{TN})^{N-n} \right] \tag{8}$$

The number of versions n that pass the acceptance to produce a correct result can be 1, 2…N. Therefore, the new reliability after using this pattern (R_{new}) is equal to:

$$R_{new} = \sum_{n=1}^{N} \left[\binom{N}{n} \left(\sum_{i=m}^{n} \binom{n}{i} (RP_{TP})^i [(1-R)P_{FP}]^{n-i} \right) (RP_{FN} + (1-R)P_{TN})^{N-n} \right] \tag{9}$$

Finally, the percentage improvement in software reliability relative to the maximum possible improvement is equal to:

$$= \frac{R_{new} - R_{old}}{1 - R_{old}} \times 100\% = \frac{R_{new} - R}{1 - R} \times 100\% \tag{10}$$

As shown in Equation (9) and (10), the reliability improvement in this pattern depends on the reliability and number of versions N, and on the performance and the effectiveness of the acceptance test used. The acceptance test should be carefully designed, reasonably simple, highly reliable, and with a high error detection coverage in order to mask the faulty outputs from participating in the voting step.

• **Safety:**

The presented pattern includes the concepts of diverse programming and fault detection with acceptance test and voting. According to the software requirements in the standard IEC 61508-3 [3], the recommendations for these techniques are shown in Table 2.

According to the last table, we think that this pattern is suitable and highly recommended only for very high critical applications with high safety integrity levels (SIL4 and SIL3), recommended for lower level (SIL2) and with no recommendation for SIL1.

Table 2. Recommendations for safety integrity levels

Techniques	SIL1	SIL2	SIL3	SIL4
Diverse programming	R	R	R	HR
Fault detection and diagnosis (Voting)	–	R	HR	HR
Fault detection and diagnosis (Acceptance Test)	–	R	HR	HR

• **Cost:**

In comparison to the basic system, this pattern is resulting in high additional costs. These costs can be divided into two parts.

– Recurring cost: includes the cost for the N different hardware units that are used for the parallel execution of the N-version software. So, the recurring cost will be N times (N*100%) comparing to the recurring cost for the basic system that includes a single version. In this pattern, the voter and the acceptance test are implemented in software. Therefore, this pattern includes additional recurring cost for the used memory.

– Development cost: The development of N diverse software versions will cost more than the development of single version software. An estimation of the development cost of N-version software has to consider the following aspects:

◆ The N versions have the same specification, and only one set of specification has to be developed [30].

◆ The cost for developing N versions prior to the verification and the validation phase is N times the cost for developing a single version [31].

◆ The management of an N-version project imposes overhead not found in traditional software development [30].

◆ The different versions can be used to validate each other [30]. While this approach could be used to decrease the cost for using verification and validation tools, it is not recommended as all versions implemented might contain a similar or even the same fault.

Exact information about the additional costs of creating N version instead of a single version is limited. The estimated practical cost of development of multiversion system showed that the costs increase sub-linearly with the number of components [32]. Moreover, it is stated in [33] that each additional version costs about 75-80% of a single version.

In addition to the previous costs, this pattern includes extra cost for developing and verifying an effective acceptance test.

• **Modifiability:**

The following possible modifications have to be considered:

1) Modification of a single version: It is possible to modify a single version either to remove a newly discovered fault or to improve a poorly programmed function [34]. In this case, the initial specification remains without any modification and the modification of this version is similar to the modification of single version software following a standard fault removal procedure.

2) Modification of all member versions: The reason for modification of all ***N*** versions is either to add a new functionality or to improve the overall performance of the N-version software [34]. In this case, the initial specification has to be modified and all ***N*** versions must be modified and tested independently by independent teams. In general, the modification of N-version software is re-

markably more difficult than the modification of single version software.

3) Modification of the acceptance test (AT): The acceptance test can be considered as an independent software module that is checking the output of each of the N versions. Thus, this acceptance test can be easily modified without any influence on the different versions.

4) Modification of the voter: The separation of the voting module from the N versions and the acceptance test allows easy modification or changes of the voting technique.

• **Impact on Execution Time:**

The diverse software versions in this pattern are executed in parallel, ideally on N independent hardware devices. As the execution times of these software versions might differ as they are implemented differently, the voter has to wait for the outputs of all software versions to be checked by the acceptance test before applying the voting algorithm. Thus, the total time of execution is determined by the slowest version in addition to the typically relatively small time to execute the acceptance test and the voting algorithm. In general, if we can neglect the execution time of the acceptance test and the voter, then the execution time of this pattern is slightly equal to single version software.

It is also possible to execute the independent versions followed by the acceptance test and voting algorithm sequentially on a single hardware. However, the time of execution will increase by N times of a single version. This disadvantage[11] makes the sequential execution less attractive, especially for time critical applications.

Implementation

The acceptance voting pattern is a hybrid pattern that combines the idea of N-version programming and fault detection using an acceptance test. Therefore, the success of this pattern depends on three factors:

1) The quality of the acceptance test is an important factor. Thus, the acceptance test should be carefully designed to detect most of the possible software faults.

2) The N diverse software versions and especially the level of diversity between these versions to avoid common failures between different versions. In order to increase the level of diversity and the independence of the designed versions, the following have been recommended in [30]:

• The use of a complete, correct, and carefully documented specification to prevent an error in the specification from propagating to the different versions.

• The use of independent and isolated teams of programmers with diversity in their training and experience.

• The use of diverse algorithms and diverse implementation techniques.

[11]Note: Another disadvantage is that the execution on a single hardware can tolerate only few hardware faults (certain transient faults) while the approach on N different hardware devices can tolerate most transient and permanent hardware faults. However, even in this case faults have to be considered that could affect all N versions simultaneously.

• The use of diverse programming languages.

• The use of diverse compilers, development tools, and test methods.

With respect to N-version programming, it has to be noted that experiments have shown that developers which implement the same function independently tend to make the same faults. For this reason, the assumption of statistically independent failure behavior of the software versions does not hold [35,36]. Approaches modeling this dependency structure (e.g. [37]) and corresponding empirical studies [38,39] are known which allow certain (model-based) predictions of failure probabilities in systems built on N-version programming. The measures presented above in this section try to decrease the dependencies between the different software versions. The assumption is that different development methodologies lead to diversity in decision and thus diversity in the behavior of the resulting software. However, even with this increased effort, the absence of undesired dependencies between the diverse software versions cannot be guaranteed [36,40,41]. For this reason, it is recommended to apply N-version programming in combination with further software fault tolerance measures as the acceptance tests applied in this pattern.

3) The use of a suitable voting technique to implement the voting component such as:

• Majority Voting: It is the simplest and most common used method that is used to find the output, where at least $\lceil (n+1)/2 \rceil$ variant results agree.

• Plurality Voting (PV): It is a simple voter, that implements m-out-of-n voting, where m is less than a strict majority.

• Consensus Voting (CV) [26]: This voting method is used for multiversion software with a small output space. In this method, the result of the largest agreement number is chosen as correct output.

• Maximum Likelihood Voting (MLV) [27]: In this method, the voter uses the reliability of each software version to make a more accurate estimation of the most likely correct result.

• Adaptive Voting [42]: This technique introduces an individual weighting factor to each version which is later included in the voting procedure. These weighting factors are dynamically changeable to model and manage different quality levels of versions.

Consequences and Side Effects

Similar to the original N-version programming approach, the drawbacks of the *Acceptance Voting Pattern* are seen in the effort of developing N diverse software versions in addition to the high dependency on the initial specification which may propagate faults to all versions. With respect to safety, the problem of dependent faults in all N software versions is less critical in this pattern than in the original N-version programming approach, as the acceptance test included represents an additional measure to detect these faults.

Related Patterns
In comparison to the basic system, the Acceptance Voting Pattern allows improvements in the reliability and the safety of a software based system. As it is executed on different hardware devices, it is possible to combine this pattern with the Heterogeneous Design Pattern for the design of these diverse hardware units to deal with systematic hardware faults. This combination will improve the reliability and safety of the hardware as well as the software.

6. Conclusions

The design of safety-critical embedded applications requires an integration of the commonly used software and hardware design methods. Therefore, the use of design pattern is very promising in this application domain, if the specific properties of embedded systems are considered in the pattern representation. In this paper, we proposed an extended pattern representation for the design of safety-critical embedded applications. This representation focuses on the implications and side effects of the represented design method on the non-functional requirements of the safety-critical embedded system including safety, reliability, modifiability, cost and execution time. Two example patterns have been used to show the effectiveness of the proposed pattern representation. We expect that this extended representation will guide the selection of a suitable design as it allows evaluating alternative patterns with respect to their implications.

7. Future Work

For a successful application of the proposed representation of design patterns for safety-critical embedded systems, an integration of a higher number of design patterns is desirable. For this reason, we currently construct a pattern catalogue based on the proposed representation by collecting and classifying commonly used hardware and software design methods. Moreover, it is intended to construct the catalogue such that an automatic recommendation of suitable design methods for a given application can be achieved in the future.

8. Acknowledgments

This work was supported by the German Academic Exchange Service (DAAD) under the program: Research Grants for Doctoral Candidates and Young Academics and Scientists.

REFERENCES

[1] C. Alexander, "A Pattern Language: Towns, Buildings, Construction," New York: Oxford University Press, 1977.

[2] E. Gama, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: Element of reusable object-oriented software," New York: Addison-Wesley, 1997.

[3] IEC61508 Functional safety for electrical/electronic/ programmable electronic safety-related systems, International Electrotechnical Commission, 1998.

[4] A. Armoush, F. Salewski, and S. Kowalewski, "Effective pattern representation for safety critical embedded systems," International Conference on Computer Science and Software Engineering (CSSE 2008), pp. 91–97, 2008.

[5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal "Pattern-oriented software architecture: A system of patterns," John Wiley & Sons, Inc., New York, NY, 1996.

[6] P. Coad, "Object-oriented patterns," Communications of the ACM, Vol. 35, pp. 152–159, 1992.

[7] K. Beck and W. Cunningham, "Using pattern languages for object-oriented programs," Presented at the OOPSLA-87 Workshop on Specification and Design for Object-Oriented Programming.

[8] J. Coplien, "Idioms and patterns as architectural literature," IEEE Software, Vol. 14, pp. 36–42, 1997.

[9] B. Appleton. "Patterns and software: Essential concept and terminology," available at <http://www.enteract.com /~bradapp/docs/patterns-intro.html>.

[10] B. P. Douglass, "Doing hard time: Developing real-time system with UML, objects, frameworks, and pattern," New York: Addison-Wesley, 1999.

[11] B. P. Douglass, "Real-time design patterns," New York: Addison-Wesley, 2003.

[12] D. Gross and E. Yu, "From non-functional requirements to design through patterns," Requirements Engineering, Vol. 6, No. 1, pp. 18–36, 2002.

[13] J. Cleland-Huang and D. Schmelzer, "Dynamically tracing non-functional requirements through design pattern invariants," Workshop on Traceability in Emerging Forms of Software Engineering, in conjunction with IEEE International Conference on Automated Software Engineering, 2003.

[14] J. Fletcher and J. Cleland-Huang, "Softgoal traceability patterns," in Proceedings of the 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006), pp. 363–374, 2006.

[15] L. Xu, H. Ziv, T. A. Alspaugh, and D. J. Richardson, "An architectural pattern for non-functional dependability requirements," Journal of Systems and Software, Vol. 79, No. 10, pp. 1370–1378, 2006.

[16] S. Konrad and B. Cheng, "Requirements patterns for embedded systems," in Proceedings of the IEEE Joint International Requirements Engineering Conference (RE'02), pp. 127–136, 2002.

[17] S. Konrad, B. Cheng, and L. Campbell, "Object analysis patterns for embedded systems," IEEE Transactions on Software Engineering, Vol. 30, No. 12, pp. 970–992, 2004.

[18] K. Wolf and C. Liu, "New clients with old servers" A Pattern Language for Client/Server Frameworks," in Pattern Languages of Program Design, J. Coplien and D. Schmidt, Eds. Reading, MA: Addison Wesley, pp. 55–64, 1955.

[19] D. Riehle and H. Züllighoven, "A pattern language for tool construction and integration based on the tools and materials metaphor," in Pattern Languages of Program Design, J. Coplien and D. Schmidt, Eds. Reading, MA: Addison Wesley, pp. 55–64, 1955.

[20] S. Adams, "Functionality ala carte," in Pattern Languages of Program Design, J. Coplien and D. Schmidt, Eds. Reading, MA: Addison Wesley, pp. 55–64, 1955.

[21] R. Lajoie and R. K. Keller, "Design and reuse in object-oriented frameworks: Patterns, contracts and motifs in concert," in Object-Oriented Technology for Database and Software Systems, V. Alagar and R, Missaoui, Eds.

Singapore: World Scientific Publishing, pp. 295–312, 1995.

[22] A. Athavale, "Performance evaluation of hybrid voting schemes," M. S. thesis, North Carolina State University, Department of Computer Science, 1989.

[23] A. Avizienis and L. Chen, "On the implementation of N-version programming for software fault tolerance during execution," in Proceedings of IEEE COMPSAC 77, pp. 149–155, 1977.

[24] N. Storey, "Safety-Critical Computer Systems," Boston: Addison-Wesley, 1996.

[25] B. Prahami, "Design of reliable software via general combination of N-Version Programming and Acceptance Testing," in Proceedings of 7th International Symposium on Software Reliability Engineering ISSRE'96, pp. 104–109, 1996.

[26] D. F. McAllister, C. E. Sun, and M. A. Vouk, "Reliability of voting in fault-tolerant software systems for small output spaces," IEEE Transactions on Reliability, Vol. 39, No. 5, pp. 524–534, 1990.

[27] Y. W. Leung, "Maximum likelihood voting for fault-tolerant software with finite output space," IEEE Transactions on Reliability, Vol. 44, No. 3, 1995.

[28] G. Latif-Shabgahi, J. M. Bass, and S. Bennett, "A taxonomy for software voting algorithms used in safety-critical systems," IEEE Transactions, Reliability, Vol. 53, No. 3, pp. 319–328, 2004.

[29] B. Parhami, "Voting algorithms," IEEE Transactions on Reliability, Vol. 43, pp. 617–629, 1994.

[30] I. Koren and C. M. Krishna, "Fault-tolerant systems," Elsevier, 2007.

[31] A. Avizienis, "The N-version approach to fault-tolerant software," IEEE Transactions on Software Engineering, Vol. 11, No. 12, pp. 1491–1501, 1985.

[32] F. Daniels, K. Kim and M. A. Vouk, "The reliable hybrid pattern: a generalized software fault tolerant design pattern," in Conference PloP'97, pp. 1–9, 1997.

[33] M. Lyu, "Handbook of software reliability engineering," New York: McGraw-Hill and IEEE Computer Society Press, 1996.

[34] A. Avizienis, "The methodology of N-version programming," in Software Fault Tolerance, M. Lyu, Ed. New York: Wiley, pp. 23–46, 1995.

[35] J. C. Knight and N. G. Leveson, "An experimental evaluation of the assumption of independence in multiversion programming," IEEE Transactions on Software Engineering, Vol. 12, pp. 96–109, 1986.

[36] F. Salewski, D. Wilking, and S. Kowalewski, "The effect of diverse hardware platforms on n-version programming in embedded systems-an empirical evaluation," in 3rd International Workshop on Dependable Embedded Systems (WDES'06), 2006.

[37] B. Littlewood and D. R. Miller, "Conceptual modeling of coincident failures in multiversion software," IEEE Transactions on Software Engineering, 1989.

[38] J. G. W. Bentley, P. G. Bishop, and M. J. P. van der Meulen, "An empirical exploration of the difficulty function," in Computer Safety, Reliability and Security (Safecomp), 2004.

[39] X. Cai and M. R. Lyu, "An empirical study on reliability modeling for diverse software systems," 15th International Symposium on Software Reliability Engineering (ISSRE), 2004.

[40] B. Littlewood, P. Popov and L. Strigini, "A note on modeling functional diversity," in Reliability Engineering and System Safety, 1999.

[41] F. Salewski and S. Kowalewski, "Achieving highly reliable embedded software: An empirical evaluation of different approaches," in Proceeding of 26th International Conference on Computer Safety, Reliability and Security (SAFECOMP'07), pp. 270–275, 2007.

[42] K. Kanoun, M. Kaaniche, C. Beounes, J. C. Laprie, and J. Arlat, "Reliability growth of fault tolerant software," IEEE Transactions on Reliability, Vol. 42, No. 2, 1993.

***J. Software Engineering & Applications*, 2009, 2:13-19**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

An Approach to Generation of Process-Oriented Requirements Specification

Jingbai Tian[1], Keqing He[1], Chong Wang[1], Huafeng Chen[1]

[1]State Key Lab of Software Engineering, Wuhan University, China
Email: Tianjingbai@live.cn

Received January 10th, 2009; revised February 21st, 2009; accepted March 19th, 2009.

ABSTRACT

In service-oriented computing, process model may serve as a link to connect users' requirements with Web Services. In this paper, we propose an approach and related key techniques to generate process-oriented requirements specification from user's goal. For this purpose, a requirements description language named SORL will be provided to capture users' requirements. Then, a unified requirements meta-modeling frame RPGS will be used to construct reusable domain assets, which is the basis of generating requirements specifications. Finally, a set of rules are defined to extract process control structures from users' requirements described with SORL, so that we can convert requirements description into process-oriented requirements specification smoothly.

Keywords: *Requirements Specification*, *Process Modeling*, *Process Control Stucture*, *Networked Software*

1. Introduction

Service-oriented computing (SOC) [1] paradigm is deemed as an active topic in recent years, because it provides a loosely coupled, highly interoperability and high levels of reuse application architecture. As a typical kind of SOC applications, networked software (NS) [2,3] is born as a software system whose structure and behavior can evolve dynamically. Following the idea of user-centric development, NS can satisfy users' demands dynamically by composing web services distributed among the network hosts.

When customizing personalized requirements of NS, one of the key problems is to elicit users' requirements precisely. In the network environment, it is difficult for users to describe their requirements correctly and completely. What's more, users' requirements are always with preference and rapid change. Traditional requirements elicitation methods collect initial requirements by memo of interview transcripts, which is hard to capture rapid changes of requirements in the network environment. Moreover, Users usually describe requirements with imprecise natural languages, and it's difficult for them to grasp formal or logical requirements description languages. Accordingly, a requirements description language called Service-Oriented Requirements Language (SORL) [4] was proposed in our previous works. It defined set of natural language patterns to describe requirements and supported online requirements elicitation.

In order to develop software rapidly, many previous researches emphasize on using domain acknowledge to support requirement assets reuse. In order to analyze and satisfy users' requirements in networked environment, we proposed a domain requirements metamodelling framework called RGPS (Goal-Role-Process-Service) [5]. It is tended to model common requirements assets in a domain, including domain ontology and four types of domain models. Domain ontology consists of entity ontology and operation ontology, which represents noun terms and verb terms respectively. Domain models can resides in role layer, goal layer, process layer and service layer. Role layer describes organizations, roles, participants and the relationships between them. Goal layer addresses users' requirements based on SORL and provides a goal based hierarchy for requirements refinement. Process layer and service layer present the processes and web services that can fulfill users' requirements.

Based on RGPS framework, users' requirements in SORL can be translated into customized goal model and process chain step by step. Finally, these models expressed with OWL can be saved as the corresponding Web service-based requirements specifications. During the process of translating, process model plays a very important role. Because process can not only give a detailed perspective on how a service operates, but serve as a bridge between users' requirements and service composition. According to the refining process in RPGS, goals can be refined as operational goals, which can be mapped to some certain processes. Compared with goal model, process model contains control structures. In order to generate complete process chain from initial requirements, a control structure extraction method is expected. In this paper, an approach to generate process-oriented requirements specification is proposed to extract control structure from language patterns and business rules. This approach can be used to generate a process-based requirements specification from users' requirements and support web services composition in NS.

The rest of this paper is organized as follows: Section 2 introduces the theoretical foundation of our approach. Section 3 gives the overview of the requirements specification generation approach. Section 4 states the process control structure extraction method with a case study. Section 5 discusses related works, followed by the conclusions and future work in Section 6.

2. Theoretical Foundation

2.1 SORL: A Requirements Description Language

Natural language pattern is the key to information extraction. Based on natural language patterns, SORL is designed for online requirements elicitation. Figure 1 illustrates the requirements description metamodel in SORL. There are four layers in the syntactic structure of SORL, which are sentence, pattern, phrase and word. More specifically, Word includes concepts in the domain ontology and key words defined in sentence and pattern (e.g. preposition and adverb).

There are 10 types of patterns in SORL, covering the description of functional goals, qualitative nonfunctional goals, quantitative non-functional goals, and precondition and post condition of a goal, etc.

The SORL sentence, which can be simple sentence or compound sentence, is composed of patterns. Simple sentence can be used to describe an activity, a constraint, or a state. Compound sentence is composed of natural language pattern, which can be *Loop Sentence*, *Choose Sentence* and *Trigger Sentence*. In this paper, we will pay more attention to compound sentences. More details of SORL are provided in [6].

2.2 RGPS: A Domain Metamodeling Frame

In order to develop software rapidly, reuse of both code and components should be taken into account. Common requirements resources are also important asserts in software development, which should be given to recovery and reuse. Many researches recognized the potential benefit of requirements reuse [7,8], and provided corresponding methods. In [9], for example, ontology was used to represent common requirements in certain domains. In order to sort and reuse requirements assets in the networked environment, a unified requirements meta-modeling frame named RPGS is proposed. The frame is consisted of domain ontology and domain model. Domain ontology includes entities and operations, which can be imported by domain models. There are four meta-models in RGPS:

Role metamodel describes the user roles of networked softwares, the actors who play roles, the contexts where actors are, intercourse between different roles, and the business rules to be complied with.

Goal metamodel describes the requirements goals of users, including functional, non-functional goal, and semantic relationships between them (e.g. depend, conflict, etc.). The goals can be refined to operational goal further, which can be realized by business processes. The process of goal refinement will not halt until all the goal leaves are operational goals.

Process metamodel defines information of business processes, including functional description such as IOPE (input, output, precondition and effect), and non-functional description such as *Qos expectation* and individualized *business context expectation*. A process can realize at least one functional goal and promote realization of non-functional goals. A process can be either an atomic process or a composite process, and a composite process has its own control structures.

Service metamodel provides the solution based on service resources, with functional and non-functional attributes. A service-based solution can be used to realize the specified business process.

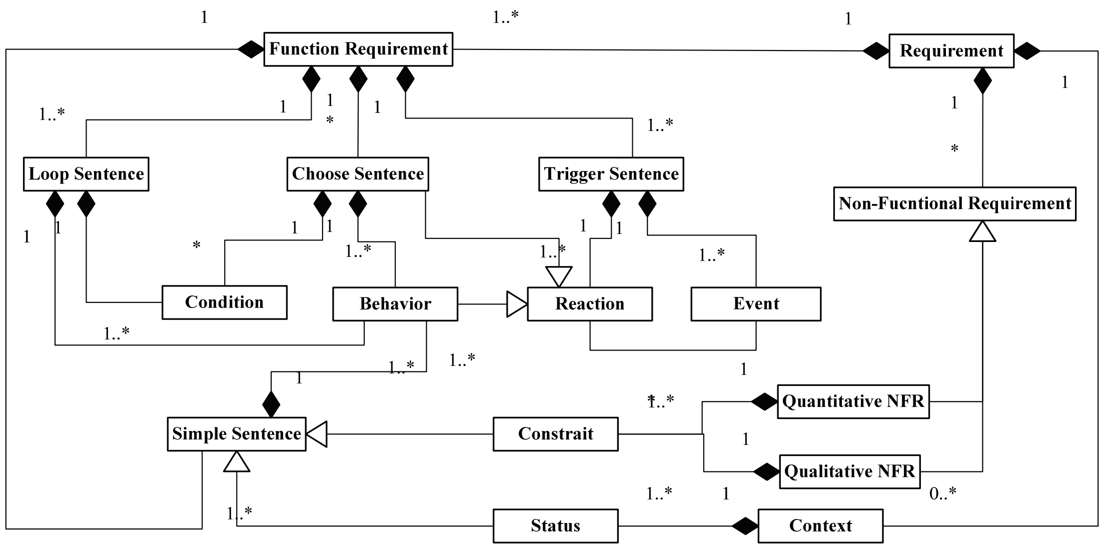


Figure 1. Requirements description metamodel

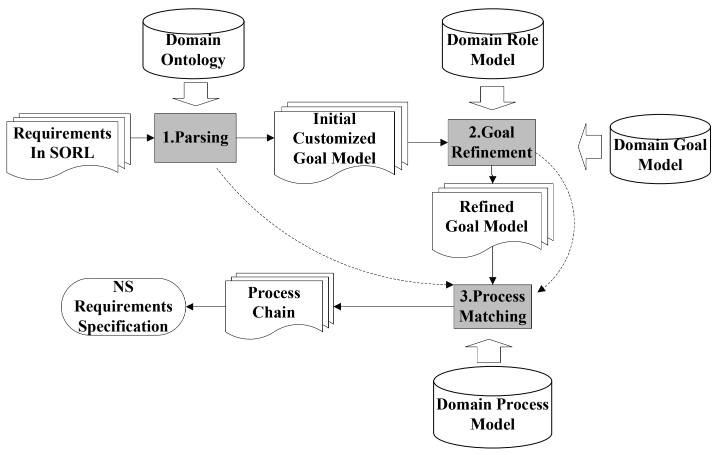


Figure 2. Overview of the NS requirements specification generation process

The domain model is provided by domain experts under the guidance of these four metamodels. Some domain requirements assets are stored as domain knowledge in each layer of RGPS, which corresponds to the respective requirements of different users group. Requirements of the user are elicited, analyzed, and finally matched to the common domain requirements specifications described with Web services.

3. Overview of the Approach

Figure 2 illustrates the overview of the requirements specification generation process. The three shadowed parts are key steps to convert requirements into NS requirements specifications.

- The first step is to parse users' requirements described in SORL to initial customized goal models by a finite automaton. SORL and domain ontology share a common vocabulary to ensure that the requirements can be parsed to goal model in a consistent manner. All the functional goals and non-functional goals are matched against domain goal model. All the unmatched goals will be marked, and stored into the specification repository for further analysis by domain engineers to check whether common requirements assets are enough.
- Secondly, each goal in the first step will be linked to a role. The goals can be decomposed into more concrete sub-goals according to the hierarchical structure in domain goal model. While the sub-goals are all operational goals, the refinement process is ended. The goal refinement process is shown in Figure 3. An operational goal means a goal which can be matched to a simple process or a composite process consistently. In this step, the goals depended by existing goals from users' requirements will also be added to improve users' demands. For example, if user needs the goal "book airline ticket by credit card". To achieve this goal, the goal "validate credit card" must be carried out previously.

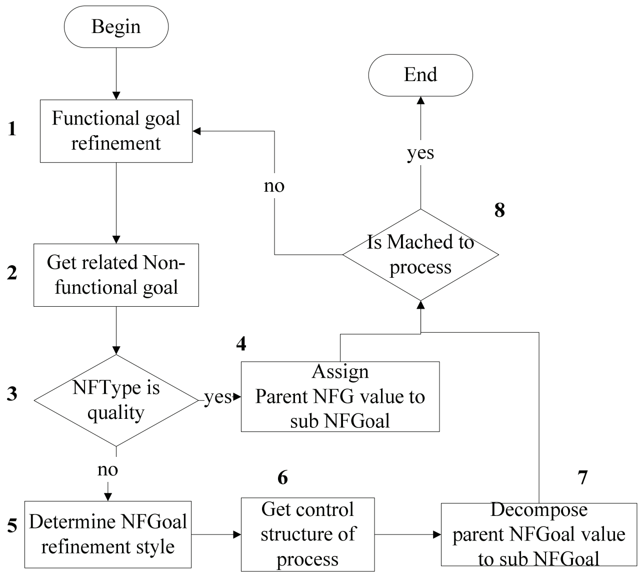

Begin
1
Functional goal refinement
End
yes
no
8
Is Mached to process
2
Get related Non-functional goal
4
3
NFType is quality
yes
Assign Parent NFG value to sub NFGoal
no
6
7
5
Determine NFGoal refinement style
Get control structure of process
Decompose parent NFGoal value to sub NFGoal

Figure 3. Goal refinement process

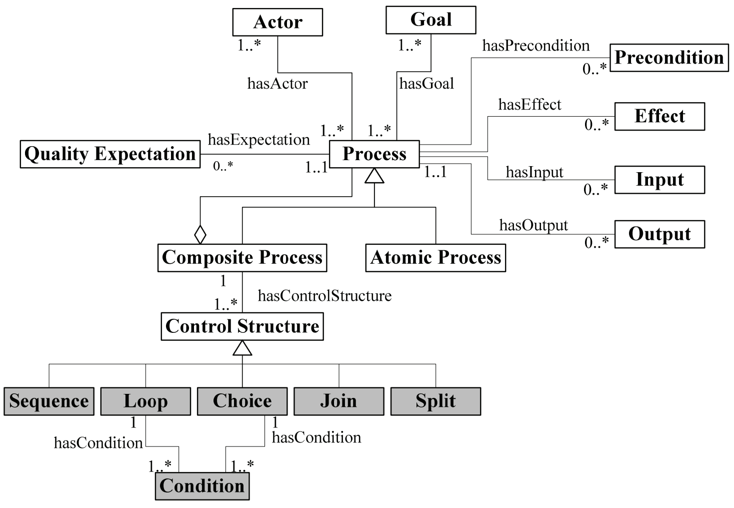


Figure 4. Process metamodel in RPGS

- In the third step, a set of processes are gained based on the refined goal model composing operational goals. Since process model has its own control structure (e.g. sequence, choice, join, etc), a set of extraction rules is needed to extract control structures from above two steps and composite these processes into process chains. With the control structures, the discontinuous processes can be integrated as process chains and saved as requirements specification finally.

In this section, the overview of our approach is illustrated, in next section, the process control structure extraction rules will be discussed in detail.

4. Extraction Rules for Process Control Structure

This section discusses some rules of extracting process control structures. Figure 4 illustrates the process metamodel in RPGS frame. The shadowed parts are primitive control structures defined in RPGS, i.e. Sequence, Loop, Choice, Join, and Split. In order to obtain control structures in process chains, each control structure should be mapped in metamodel layer.

In our approach, process control structures are generated from two sources: one is sentences based on SORL. The other is Depend relationships between related goals of the specific process.

4.1 Extraction Rules Based on SORL Sentence

As Section 2 describes, two kinds of compound sentence in SORL can be used for process control structure extraction, i.e. Loop sentence and Choose sentence.

Loop Sentence

The loop control structure can be extracted from loop sentence. Loop sentence describes system cyclic behavior under a certain condition. As Figure 5 shows, loop sentence is represented as "loop <function requirement pattern> until <condition pattern>". To extract loop control construct from the corresponding loop sentences, we define extraction rules as follows:

Def.1 In a Loop sentence, *FP* is the Function Requirement Pattern and *CP* is the Condition Pattern. So the Loop sentence "loop *FP* until *CP* is true" denoted as *LoopSentence* (*FP, CP*).

Def.2 In a process chain, *P* is a process and *C* is a condition. "Loop a Process *P* until condition *C* is true" will be denoted as *LoopControlStructure* (*P, C*).

Def.3 If *FP* is a Function Pattern, the process related to the corresponding goal of this Function Pattern is denoted by *P*. If *CP* is a Condition Pattern, the related Condition in process chain is denoted by *C*.

Extraction Rule.1:

LoopSentence (*FP*,*CP*) $\Rightarrow$ *LoopControlStructure* (*P*,*C*)

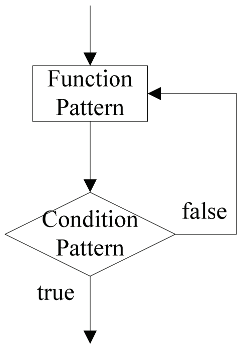


Figure 5. Loop sentence

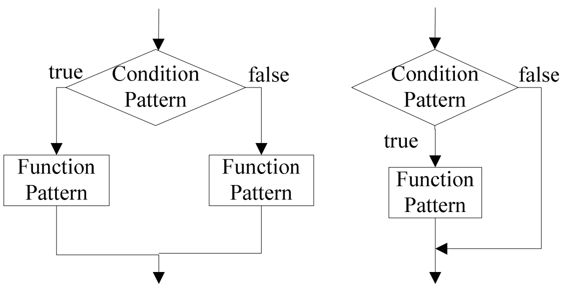


Figure 6. Choose sentence

Choose Sentence

The extraction of Choice control structure is similar to the one of Loop control structure. In SORL, choose sentence describes a selection of different system behaviors. Choose sentence is represented as "if <Condition Pattern>, then <Function Pattern>, else <Function Pattern>", as Figure 6 depicts. The extraction rules of choice control structure are the followings:

Def.4 FP_A and FP_B are two alternative Function Requirement Patterns in a Choose Sentence, and *CP* is the Condition Pattern. The Choose sentence "if *CP*, then FP_A, else FP_B" is denoted by *ChooseSentence* (*CP*, FP_A, FP_B). When *FP* is a Function Requirement Pattern in a Choose Sentence, the Choose sentence "if CP, then FP" is denoted by *ChooseSentence* (*CP*, *FP*).

Def.5 *C* is a condition in process chain, and P_A and P_B are two processes. The Choice control structure in process chain is denoted by *ChoiceControlSturcture* (*C*,P_A,P_B), which means "if C is true, then execute PA; if not, execute PB". When P is a process, if C is true, then executes P, else skip P, this is denoted by ChoiceControlSturcture (C,P)

Extraction Rule.2:

ChooseSentence (*CP*, FP_A, FP_B) $\Rightarrow$ *ChoiceControlStructure* (*C*, P_A, P_B)

ChooseSentence (*CP*,*FP*) $\Rightarrow$ *ChoiceControlStructure*(*C*,*P*)

4.2 Extraction Rules based on Goal Model

In RGPS, Goal metamodel defines Depend relationships between goals. A Depend relationship can be either an Object Depend or a Conditional Depend. Object Depend used to indicate the Input/Output relationship between two goals, which is related to dataflow in system. Conditional Depend describes the Precondition/Postcondition between two goals, which is related to business rules.

Object Depend

In Object Depend relationship, realization of a goal depends on output data of the other goals. We assume that such dataflow in real time systems would never get failure. Correspondingly, the realization of related process also depends on the others. Therefore, we can extract process control structure from Object Depend relationship this section defines three extraction rules for Sequence, Split and Join control structure respectively.

Def.6 G_A and G_B are two goals in refined goal model. If G_A Object Depend on G_B, the relationship is denoted by *ObjectDepend* (G_A, G_B). $G_{Ai\,(i=1,2,\ldots,n)}$ and G_B are goals in refined goal model, $G_{Ai\,(i=1,2,\ldots,n)}$ Object Depend on G_B, it is denoted by $\underset{i=1,2\ldots n}{And}\ objectDepend(G_{Ai}, G_B)$. G_A and $G_{Bi(i=1,2,\ldots,n)}$ are goals in refined goal model, G_A Object Depend on $G_{Bi(i=1,2,\ldots,n)}$, it is denoted by $\underset{i=1,2\ldots n}{And}\ objectDepend(G_B, G_{Ai})$.

Def.7 P_A and P_B are two processes in process chain. If there is a sequence control structure between P_A and P_B and P_A executes after P_B, denoted by *Sequence* (P_A, P_B).

Extraction Rule.3:

$$ObjectDepend(G_A, G_B) \Rightarrow Sequence(P_B, P_A)$$

Def.8 $P_{Ai(i=1,2\ldots n)}$ and P_B are Process in process chain, if $P_{Ai(i=1,2\ldots n)}$ is split from P_B, the control structure is denoted by *Split* (P_B, P_{Ai}).

Extraction Rule.4:

$$\underset{i=1,2\ldots n}{And}\ ObjectDepend(G_{Ai}, G_B) \Rightarrow \underset{i=1,2\ldots n}{And}\ Split(P_B, P_{Ai})$$

Def.9 $P_{Ai(i=1,2\ldots n)}$ and P_B are Process in process chain, if $P_{Ai(i=1,2\ldots n)}$ is join to P_B, the control structure is denoted by *Join* (P_{Ai}, P_B).

Extraction Rule.5:

$$\underset{i=1,2\ldots n}{And}\ ObjectDepend(G_B, G_{Ai}) \Rightarrow \underset{i=1,2\ldots n}{And}\ Join(P_{Ai}, P_B)$$

Conditional Depend

Conditional Depend describes the business rules between goals. "Goal A is Conditional Depend on Goal B" means "if Goal B cannot be achieved, then Goal A cannot be achieved; if not, it should be skipped". For example, a traveler want to travel to Wuhan, he want to "book an airline ticket to Wuhan", and to "book a standard room in Wuhan". If the first goal can't be achieved, which means he can't arrive in Wuhan in time, the second goal will be canceled. From this point of view, we defined the following rules:

Def. 10 G_A and G_B are two goals in refined goal model, If G_A Conditional Depend G_B, denoted by *ConditionalDepend* (G_A, G_B).

Def. 11 P_A is a process, if P_A is successful execution, the condition C_A in P_A 's Effect would be True, else be False.

Extraction rule.6

$ConditionalDepend(G_A, G_B) \Rightarrow ChoiceControlStructure$ (C_B, P_A)

4.3 Case Study

In this section, a case in travel and transportation domain will be demonstrated. Firstly, we constructed the domain ontology and domain model in [5]. And the following ones are requirements examples.

A customer is planning a trip to Wuhan. His SORL-ased requirements are as followings:"Query travel expense in Wuhan", "Book an airline ticket to Wuhan by credit card", "if the price is no more than 500 RMB, then, book a room, the standard of the hotel is no less than 4-Star, else, book a standard room, the standard of the hotel is equal to 3-Star."

Figure 7 Refined goal model of the case.

- Based on extraction rule 2, we can find that there is a choice control structure in the corresponding processes of "book room".
- Based on extraction rule 3, the related process of "Validate Credit Card" must be executed before the related process of "Book Airline Ticket by Credit Card".
- Based on extraction rule 6, if "book airline ticket" cannot be achieved, "book room" should be skipped.

The generated process chain is shown in Figure 8.

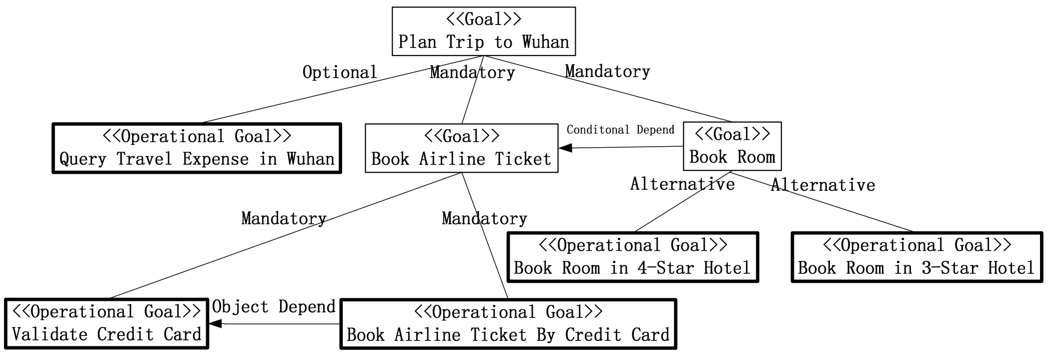


Figure 7. An example of refined goal model

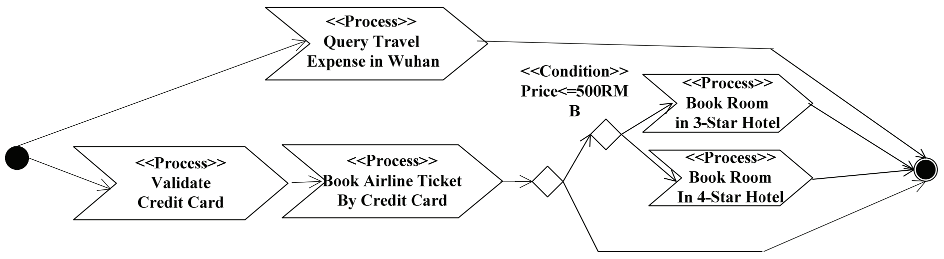


Figure 8. An example of process chain

A trial system based on this approach is build up preliminarily. With the graphical process tool, users can see their requirements directly, so that users can understand the work flow intuitively and modify the work flow with ease.

5. Related Works

Process-oriented technology has many potential advantages in software development, and many SOC technologies have adopted process in their methods to improve the service composition. For example, OWL-S [10] models services as processes to give services a more detailed perspective. In WS-BPEL [11], processes use Web Service interfaces to export and import functionality.

In [12], W. M. P. van der Aalst proposes an algorithm based on petri net to extract process model from "workflow log", which contains information about the workflow processes in real-time systems. Moreover, many systems (e.g. ERP, CRM) provide a set of pre-defined process models. In our work, we use RGPS to manage not only process model but also related goal, role and service model. A solution for process model generation from user's requirements is given.

In [13] and [14], authors established relationships between process model and high-level requirements models (e.g. KAOS and i*), presented methods to generate, validate and improve process model. In [15], authors present a framework and associated techniques to synthesis service composition from temporal business rules through process models. Compared with these approaches, our approach is more suitable for end-users and caters for the "user-centric" trend.

6. Conclusions and Future Work

In this paper, we proposed an approach to generation of process-oriented requirements specifications. It establishes mappings from SORL metamodel and Goal metamodel in RGPS to the control structures defined in Process metamodel. So it can be used to generate an NS requirements specification including goal models and process chains through these control structures.

In the future, the control structure extraction rules will be improved, the influences from goal refinement types (e.g. mandatory, optional) to the control structure will be supplemented. Moreover, the corresponding validation and verification methods will be provided later, followed by the relevant tools and platforms.

7. Acknowledgements

This research project was supported by the National Basic Research Program of China (973) under Grant No. 2006CB708302 and 2007CB310801, the National High Technology Research and Development Program of China (863) under Grant No.2006AA04Z156, the National Natural Science Foundation of China under Grant No. 60803083, 60873009, 60803025 and 60703018, the Provincial Natural Science Foundation of Hubei Province under Grant No.2006ABA228,and the Research Fund for the Doctoral Program of Higher Education of China under Grant No. 60803025.

REFERENCES

[1] M. N. Huhns and M. P. Singh, "Service-oriented computing: Key concepts and principles," IEEE Internet Computting, pp. 2–8, 2005.

[2] K. He, R. Peng, J. Liu, F. He, *et al.*, “Design methodology of Networked Software Evolution Growth based on Software Patterns,” Journal of System Science and Complexity, Vol. 19, pp. 157–181, 2006.

[3] K. He, P. Liang, R. Peng, *et al.*, “Requirement emergence computation of networked software,” Frontiers of Computer Science in China, 1(3), pp. 322–328, 2007.

[4] W. Liu, K. He, J. Wang, *et al.*, “Heavyweight semantic inducement for requirement elicitation and analysis,” In Proceedings of the 3rd International Conference on Semantics, Knowledge and Grid (SKG 2007), Xi’an, China, pp. 206–211, 2007.

[5] J. Wang, K. He, P. Gong, C. Wang, R. Peng, and B. Li, “RGPS: A unified requirements meta-modeling frame for networked software,” in Proceedings of Third International Workshop on Advances and Applications of Problem Frames (IWAAPF’08) at 30th International Conference on Software Engineering (ICSE’08), Leipzig, Germany, pp. 29–35, 2008.

[6] L. Wei, “Research on services-oriented software requirements elicitation and analysis,” Ph. D thesis, Wuhan University, 2008.

[7] P. Massonet and A. van Lamsweerde, “Analogical reuse of requirements frameworks,” Third IEEE International Symposium on Requirements Engineering, pp. 26, 1997.

[8] W. Lam, “A case-study of requirements reuse through product families,” Springer Netherlands, pp. 253–277, 1998.

[9] Y. Q. Lee and W. Y. Zhao, “Domain requirements elicitation and analysis–An ontology-based approach,” In: Proceedings of Workshop on Computational Science in Software Engineering (CSSE), pp. 805–813, 2006.

[10] M. Smith, C. Welty, and D. McGuinness, “OWL web ontology language guide,” W3C Candidate Recommendation, 2004. Available: http://www.w3.org/TR/owl-guide/.

[11] OASIS: Web services business process execution language version 2.0, 2006. Available: http://docs.oasis-open.org/ wsbpel/2.0/wsbpel-specification-draft.html.

[12] W. van der Aalst, T. Weijters, and L. Maruster, “Workflow mining: Discovering process models from event logs,” IEEE Transactions on Knowledge and Data Engineering, Volume 16, Issue 9, 2004.

[13] G. Koliadis and A. Ghose, “Relating business process models to goal-oriented requirements models in KAOS,” PKAW 2006, pp. 25–39, 2006.

[14] A. Ghose and G. Koliadis, “Actor eco-systems: From high-level agent models to executable processes via semantic annotations,” COMPSAC 2007, pp. 177–184, 2007.

[15] J. Yu, Y. B. Han, J. Han *et al.*, “Synthesizing service composition models on the basis of temporal business rules,” Journal of computer science and technology 23(6) pp. 885–894, 2008.

J. Software Engineering & Applications, **2009, 2:20-24**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

Data Hiding Method with Quality Control for Binary Images

Ki-Hyun Jung[1], **Kee-Young Yoo**[2]

[1]School of Computer Information, Yeungjin College, 218 Bokhyun-Dong, Buk-Gu, Daegu 702−721, Republic of Korea, [2]Department of Computer Engineering, Kyungpook National University, 1370 Sankyuk-Dong, Buk-Gu, Daegu 702−701, Republic of Korea.
Email: kingjung@paran.com, yook@knu.ac.kr

Received December 16th, 2008; revised January 21st, 2009; accepted February 24th, 2009.

ABSTRACT

Secret data hiding in binary images is more difficult than other formats since binary images require only one bit representation to indicate black and white. This study proposes a new method for data hiding in binary images using optimized bit position to replace a secret bit. This method manipulates blocks, which are sub-divided. The parity bit for a specified block decides whether to change or not, to embed a secret bit. By finding the best position to insert a secret bit for each divided block, the image quality of the resulting stego-image can be improved, while maintaining low computational complexity. The experimental results show that the proposed method has an improvement with respect to a previous work.

Keywords: *Data Hiding, Quality Control, Binary Images*

1. Introduction

Data hiding involves concealing information in a host signal, such as text, image, audio, or video. Binary images are two-color images, with a 0 or 1 value for each pixel, in which each pixel requires only one bit representation, to indicate black and white. The difficulty lies in the fact that changing pixel values in a binary image can cause irregularities that are visually very noticeable. Hiding data in binary images is therefore more challenging than hiding it in other formats [1].

There are two primary methods of data hiding in these images: sub-block modification and single pixel manipulation. The first modifies the sub-block, which is divided into a group of pixels. Matsui and Tanaka embedded secret data in 'dithered' images by manipulating the dithering patterns; they also embedded in fax images, by manipulating the run lengths [2]. Low *et al.* changed line spacing and character spacing to embed secret data in textual images, for bulk electronic publications [3,4]. These methods are used for some special types of binary images. The second approach modifies single pixel from black to white or vice versa: some special single pixels in the image are changed to embed the secret data. Koch and Zhao proposed a data hiding method by forcing the ratio of black and white pixels in a block to be larger or smaller than one [5]. However, there is a difficulty with this. Only a limited number of bits can be embedded, since the enforcing method has trouble dealing with blocks that have a significantly low or high percentage of black pixels. Wu *et al*. embedded bits in image blocks, selected by calculating a characteristic value and finding a pattern [6]. Lie *et al*. partitioned the binary image into blocks of 2x2 pixels and embedded a bit 0 or 1 in the block. This method can hide one bit per block by modifying 0.5 pixels on average [7]. Wu and Liu manipulated flappable pixels to enforce a specific block-based relationship in order to embed a significant amount of data without causing noticeable visual effects [8]. Venkatesan et al. proposed using the parity of blocks. The cover image is partitioned into small blocks, in which one bit information is stored. Unfortunately, if all of the pixel values belong to 0 or 1, a secret bit cannot be hidden [9].

This paper proposes a new data hiding method for binary images, using optimized bit position and parity bit check with adopted block parity. This method manipulates sub-divided blocks. The parity bit for a specified block decides whether to change or not, to embed a secret bit. By finding the best position to insert a secret bit for each divided block, the image quality of the stego-image can be maintained with relatively low computational complexity.

This paper is organized as follows. In Section 2, data hiding scheme using block parity proposed by Venkatesan *et al.* is reviewed. In Section 3, our proposed data hiding method is described in more detail. In Section 4, our experimental results are presented and discussed. Our conclusions are presented in Section 5.

2. Related Work

Venkatesan *et al*. proposed data hiding method for binary images to maintain the quality of cover image. To change a bit close to the position which has the same value, neighbor matrix was adopted. For 3x3 sub-block B_m, the resulted sub-block B''_m is more difficult to be detected than B'_m. To find the location, Venkatesan *et al*. defined a neighbor matrix.

$$B_m = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad B'_m = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & \mathbf{1} \end{bmatrix} \quad B''_m = \begin{bmatrix} 1 & 1 & 0 \\ 1 & \mathbf{1} & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Figure 1. Sub-block matrix of cover image

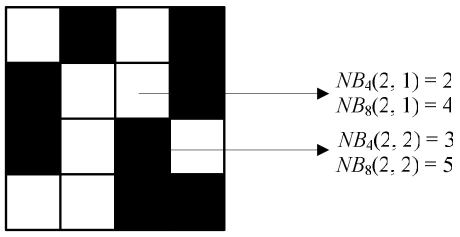


Figure 2. Difference of the neighboring pixels

But there is a room to improve the capacity with less distortion to the human visual system. This paper proposes a new data hiding method for binary images using optimized bit position and parity bit check.

3. Proposed Method

In this section, we consider the position of data embedding, and how the secret data is embedded and extracted. The binary image is made up of black and white. There is only one bit representation for each pixel, say 0 or 1.

Let C be the cover image of WxH pixels and S be the n-bit secret data. For $p(i,j)$ pixel value belonging to C image, a new pixel value is defined as $p'(i,j)$.

3.1 Embedding Scheme

The following steps are executed to embed secret data.

Step 1: For a given cover image, calculate the matrix which stores the position of the isolated pixel. Given a specified pixel value $p(i,j)$, the value of the neighboring four pixels for the row and column direction is compared, and the difference value is calculated for each 0 or 1. Next, the difference of the neighboring eight pixels which are surrounding the specified pixel is compared and calculated. These are defined as $NB_4(i,j)$ and $NB_8(i,j)$ separately, where $\Gamma(\cdot)$ is an indicator function which is taking value from {0,1}.

$$NB_4(i,j) = \sum_{k=-1}^{1} \Gamma(\{p(i+k,j) \neq p(i,j)\}) + \sum_{l=-1}^{1} \Gamma(\{p(i,j+l) \neq p(i,j)\})$$
$$NB_8(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} \Gamma(\{p(i+k,j+l) \neq p(i,j)\}) \tag{1}$$

For example, when a cover image is given as Figure 2, the difference of the neighboring pixels for $p(2,1)$ are calculated as $NB_4(2,1) = 2$ and $NB_8(2,1) = 4$ respectively.

Step 2: The cover image is partitioned into MxN-size blocks. For a specified sub-block B_m, calculate the sum of the block which is given by

$$S(B_m) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} p(k,l) \tag{2}$$

Step 3: If $S(B_m)$ is not equal to 0 or MxN, embed 1-bit of secret data into B_m to the following three cases:

Case 1: If $S(B_m)$ mod 2 is equal to 0 and the embedding bit is 1, then change the isolated pixel $p(i,j)$ where $NB_4(i,j)$ value is larger than any other pixel for $p(i,j) = 0$ pixel. When there exists another pixel which the value of the neighboring four pixels is equal, select a pixel that $NB_8(i,j)$ value is larger.

Case 2: If $S(B_m)$ mod 2 is equal to 1 and the embedding bit is 0, then change the isolated pixel $p(i,j)$ where $NB_4(i,j)$ value is larger than any other pixel for $p(i,j) = 1$ pixel. When there exists another pixel, for which the value of the neighboring four pixels is equal, select a pixel that $NB_8(i,j)$ value is larger than.

Case 3: If it does not belong to Case 1 and Case 2, any other pixel values for the block remain unchanged.

Step 4: For the embedded block B'_m which is a new block for B_m after embedding, calculate the sum of the block. When the value belongs to 0 or MxN, discard this block for embedding a secret bit.

For example, $NB_4(i,j)$ and $NB_8(i,j)$ values are calculated for the 4x4 sub-block shown in Figure 3. As a result, $NB_4(0,3) = 3$ and $NB_8(i,j) = 7$ are largest value for a sub-block, so pixel $p(0,3)$ is selected to embed a secret bit.

3.2 Extracting Scheme

The following steps are executed to recover the secret data. It can be directly extracted from the stego-image only.

Step 1: For a given stego-image, partition into MxN-size blocks. For each block B'_m, calculate the sum of the block which is given by

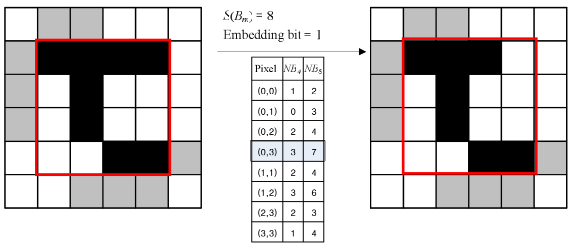
$$S(B'_m) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} p'(k,l) \tag{3}$$

Step 2: If $S(B'_m)$ is not equal to 0 or MxN, extract 1-bit from B'_m to the following two cases:

Case 1: If $S(B'_m)$ mod 2 is equal to 0, the extracted bit is 0.

Case 2: If $S(B'_m)$ mod 2 is equal to 1, the extracted bit is 1.

Step 3: For all the embedded block B'_m, stack an extracted bit.



Pixel	NB_4	NB_8
(0,0)	1	2
(0,1)	0	3
(0,2)	2	4
(0,3)	3	7
(1,1)	2	4
(1,2)	3	6
(2,3)	2	3
(3,3)	1	4

Figure 3. Example of data embedding process

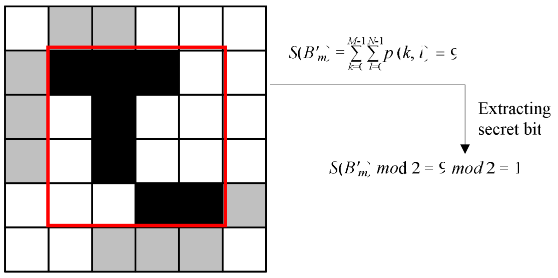

$S(B'_m) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} p(k,l) = 9$
Extracting secret bit
$S(B'_m)$ mod 2 = 9 mod 2 = 1

Figure 4. Example of data extracting process

For a sub-block B'_m, the sum of block $S(B'_m)$ is equal to 9 and finally the value of $S(B'_m)$ *mod* 2 = 9 *mod* 2 = 1. So the embedded bit is 1.

4. Experimental Results

In our experiments, four 512x512 binary images shown in Figure 5 were used as cover images. The secret data was generated by pseudo-random numbers. This study adopts the peak signal-to-noise ratio (PSNR) as extending 1-bit binary value to 8-bit, and calculates capacity for the amount of embedded data. Our experiments used an extended eight-bit value, in which white value is extended to 0xFF and black to 0x00.

Figure 6 shows the stego-images and embedded blocks for the Baboon image, where these compare with for various MxN block sizes. As the results, there are not any visual artifacts present.

We found that these distortions do happen, to the edge areas of the image. This means that such distortions will be less noticeable, because changes to edge areas of binary image are generally less conspicuous to human eyes.

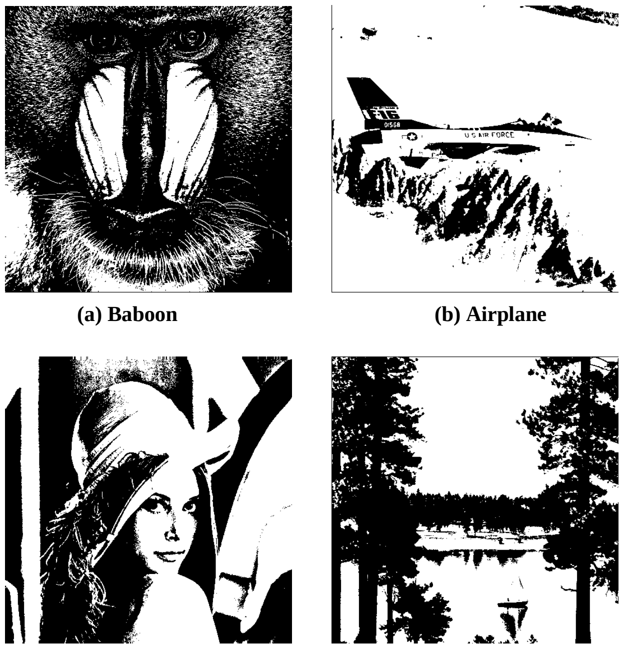


Figure 5. Four cover images

Figure 6(a) illustrates a Baboon image, size 512x512 and it takes 13,415 bits embedded blocks, which are divided into 2x2 sub-blocks. Figure 6(c) and 6(e) show the stego-images after embedding 11,960 bits and 8,301 bits respectively. Figure 6(b), 6(d) and 6(f) display the embedded pixels for 2x2, 3x3 and 4x4 sub-block respectively.

Table 1 shows the capacity of the proposed method on the different sub-block sizes. The table shows that the proposed method can hide more secret data when a cover image changes its pixel value rapidly. For example, the Baboon has a higher capacity than the other images, which contain many blocks so that all of the sub-block's values are scattered uniformly and randomly. As sub-block size is larger, the capacity is low and the PSNR value is high. This means that there is a trade-off between capacity and invisibility. Each sub-block can embed a secret bit, except a full black or white block.

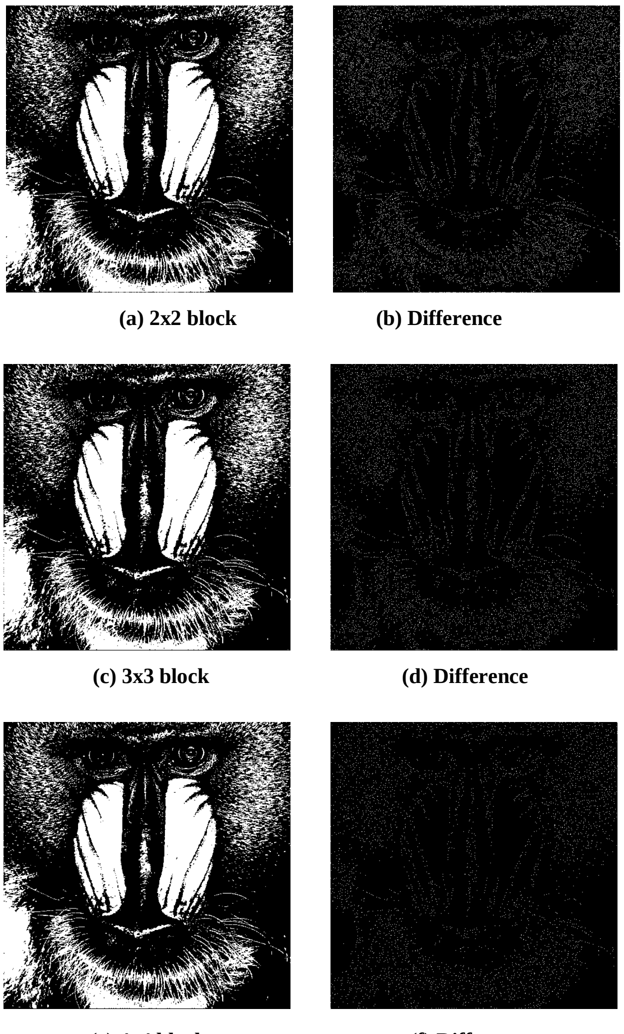


Figure 6. Embedding results: (a) (c) (e) stego-images; (b) (d) (f) difference images

Table 1 shows the comparison with other method. The proposed method can hide more secret data than other method although the PSNR value is similar for each image. In here, the PSNR value is adopted for comparison with other method.

Figure 7 shows the stego-images after embedding secret data for each 3x3 sub-blcok. In these experiments, the proposed method produces less distortion to the cover image, in spite of its higher capacity. For the stego- images shown in Figure 7(a), 7(b), 7(c) and 7(d), it can be embedded less than other two images since many sub-blocks in these images have a full 0 or 1 value, where discarded for embedding.

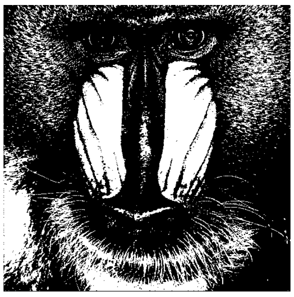
Figure 8 shows the detailed image for result. The stego-images are shown in Figure 8(b), 8(c), 8(e) and 8(f) for 170x170 cover images. As shown in Figure 8, the proposed method has less distortion than the previous method.

5. Conclusions

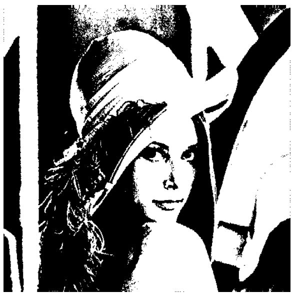
We have proposed a data hiding method using optimized bit position and parity bit check for binary images. No reference to the original cover image was required when extracting the embedded secret data from the stego-image.

Table 1. Comparison of embedding for 3x3 block size

Cover Image	Venkatesan et al.'s		The proposed method	
	Capacity (bits)	PSNR (dB)	Capacity (bits)	PSNR (dB)
Baboon	9,441	16.36	11,960	16.32
Airplane	3,293	21.25	3,587	21.50
Lena	3,657	20.45	4,433	20.51
Boat	3,714	20.56	4,429	20.45


(a) capacity=11,960bits

(b) capacity=3,587bits


(c) capacity=4,433bits

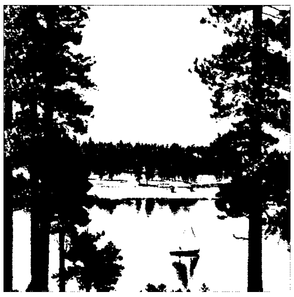
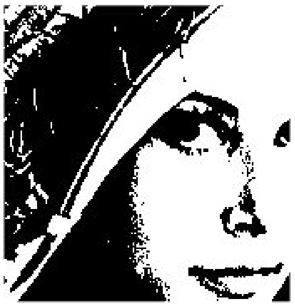
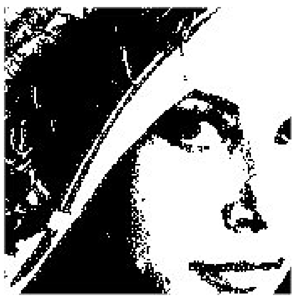
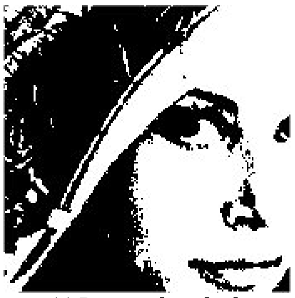
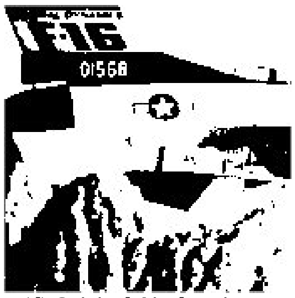

(d) capacity=4,429bits

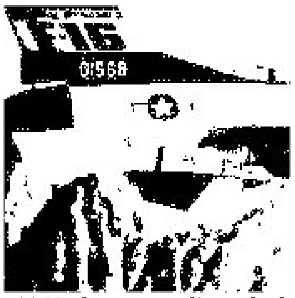
Figure 7. Stego-images of the proposed method


(a) Original Lena image


(b) Venkatesan et al's method


(c) Proposed method


(d) Original Airplane image


(e) Venkatesan et al's method

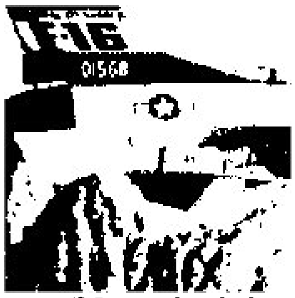

(f) Proposed method

Figure 8. Detailed comparison for stego-images.

This method manipulated blocks which were sub-divided into a small *M*x*N*-size block. The parity bit for a specified block decided whether to change or not, to embed a secret bit. By finding the best position to insert a secret bit for each divided block, the image quality of the stego-image could be improved, while retaining a low computational complexity. Our experimental results have shown that the proposed method provided a better way to hide more secret data compared with other method without making noticeable distortions.

REFERENCES

[1] H. Liang, W. Ran, and X. Nie, "A secure and high capacity scheme for binary images," Proceedings of the ICWAPR, pp. 224–229, 2007.

[2] K. Matsui and K. Tanaka, "Video-steganography: How to secretly embed a signature in a picture," Proceedings IMA Intellectual Property Project, pp. 187–206, 1994.

[3] S. H. Low, N. F. Maxemchuk, and A. M. Lapone, "Document identification for copyright protection using

centroid detection," IEEE Transactions on Communications, pp. 372−383, 1998.

[4] J. T. Brassil, S. H. Low, and N. F. Maxemchuk, "Copyright protection for the electronic distribution of text documents," Proceedings of IEEE, pp. 1181−1196, 1999.

[5] E. Koch and J. Zhao, "Embedding robust labels into images for copyright protection," Proceedings of the International Congress on Intellectual Property Rights for Specialized Information, Knowledge & New Technologies, 1995.

[6] M. Wu, E. Tang, and B. Liu, "Data hiding in digital binary images," IEEE International Conference on Multimedia & Expo, 2000.

[7] C. Liu, Y. Dai, and Z. Wang, "A novel information hiding method in binary images," Journal of Southeast University, 2003.

[8] M. Wu and B. Liu, "Data hiding in binary image for authentication and annotation," IEEE Transations on Multimedia, pp. 528−538, 2004.

[9] M. Venkatesan, P. Meenakshidevi, K. Duraiswamy, and K. Thiagarajah, "A new data hiding scheme with quality control for binary images using block parity," 3rd Inter. Symposium on Information Assurance and Security, pp. 468−471, 2007.

[10] H. Yang and A. C. Kot, "Pattern-based data hiding for binary image authentication by connectivity-preserving," IEEE Transactions on Multimedia, Vol. 9, No. 3, pp. 475−486, 2007.

[11] J. Chen and T. S. Chen, "A new data hiding method in binary images," Proceedings of 4th International Symposium on Multimedia Software Engineering, pp. 88−93, 2003.

[12] H. B. Zhang and L. Man, "Data hiding in binary line drawing images," Wavelet Analysis and Pattern Recognition, ICWAPR, pp. 134−140, 2008.

[13] Y. J. Chang and J. C. Lin, "Data hiding using VQ index file," Intelligence and Security Informatics, ISI 2008, pp. 230−232, 2008.

[14] H. Gou and M. Wu, "Improving embedding payload in binary images with super-pixels," Image Processing, ICIP 2007, pp. 277−280, 2007.

[15] S. Huang and J. K. Wu, "Optical watermarking for printed document authentication," Information Forensics and Security, Vol. 2, pp. 164−173, 2007.

[16] K. H. Jung, K. S. Ha, and K. Y. Yoo, "Data hiding in binary images by pixel-value weighting," Convergence and Hybrid Information Technology, ICHIT 2008. pp. 262−265, 2008.

[17] A. M. Attar, O. Taheri, S. Sadri, and A. F. Rasoul, "Data hiding in halftone images using error diffusion halftoning with adaptive thresholding," Electrical and Computer Engineering, CCECE 2006, pp. 2029−2032, 2006.

J. Software Engineering & Applications, **2009, 2:25-33**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

Formal Semantics of OWL-S with Rewrite Logic[#]

Ning Huang[*], Xiao Juan Wang[*], Camilo Rocha[+]

[*]Beihang University, Beijing, China, [+]University of Illinois at Champaign Urbana, USA
Email: hn@buaa.edu.cn; wxjbuaa@hotmail.com

Received December 29th, 2008; revised January 27th, 2009; accepted February 27th, 2009.

ABSTRACT

SOA is built upon and evolving from older concepts of distributed computing and modular programming, OWL-S plays a key role in describing behaviors of web services, which are the essential of the SOA software. Although OWL-S has given semantics to concepts by ontology technology, it gives no semantics to control-flow and data-flow. This paper presents a formal semantics framework for OWL-S sub-set, including its abstraction, syntax, static and dynamic semantics by rewrite logic. Details of a consistent transformation from OWL-S SOS of control-flow to corresponding rules and equations, and dataflow semantics including "Precondition", "Result" and "Binding" etc. are explained. This paper provides a possibility for formal verification and reliability evaluation of software based on SOA.

Keywords: *SOA*, *Web Services*, *OWL-S*, *Formal Semantics*, *Rewrite Logic*, *Consistent Transformation*, *Reliability Evaluation*

1. Introduction

Service Oriented Architecture (SOA) adopts the Web services standards and technologies and is rapidly becoming a standard approach for enterprise information systems. We believe that there will be a heavy demand of reliability evaluation for the SOA software in the early development phase.

Web services are the essential of the SOA software, because SOA offers one such architecture, it unifies business processes by structuring large applications as an ad hoc collection of smaller modules called "services", Services-orientation aims at a loose coupling of services with operating systems, programming languages and other technologies which underlie applications. There are many higher level standards such as BPEL [1], WSCI, BPML, DAML-S (the predecessor of OWL-S), etc. OWL-S (Web Ontology Language for services) is a well-established language for the description of Web services based on ontology, it has been recommended by Web-ontology Working Group at the World Wide Web Consortium [2].

In order to undergo more accurate reliability evaluation and prediction of software based on SOA in the early phase, we should give software architecture the description semantics for gaining more components information, according to this motivation, we plan to use OWL-S to describe the software architecture, and then use formal method to construct a well-defined mathematical model of the system described by OWL-S, once we have this formal model, we will compute the reliability of the software based on the formal model. Here, as a meta-language of rewrite logic, Maude has been chosen as a language to give OWL-S formal semantics in a framework in static and dynamic aspects. With this framework in hand, an OWL-S specification can be transformed into an easy-understand formal one only concerning syntax, and this transformation makes a critical contribution for the formal verification and reliability evaluation of OWL-S model using the Maude language.

The rest of this paper is organized as follows. We will give an overview of related works in Section 2, the background of our method will be presented in Section 3 by an overview of OWL-S, rewrite logic and Maude, section 4 presents the abstraction of the model to introduce what we should abstract from the model. How to give OWL-S model the formal semantics using Maude in static and dynamic aspects will be presented respectively in Section 5 and Section 6. Finally, we will give a conclusion in Section 7 and an acknowledgement in Section 8.

2. Related Works

Most of previous related works focus on model checking instead of providing a precise formal semantics for the specification. For example, [3] converts OWL-S Process Model using a C-Like code specification language and then uses BLAST to validate it by the test cases automatically generated in the model checking process. [4] proposes a Petri Net-based operational semantics, which models the control flow of DAML-S (the former of OWL-S) Process Model, but lack of dataflow. Based on the work of [4,5] extends it to translate OWL-S Process Model using Promela, a SPIN specification language, and uses SPIN to do model checking. [6] presents a formal denotational model of OWL-S using the Object-Z(OZ) specification language, but it focuses on the formal model

[#]Supported by the 11th Defense Five-Year plan Foundation, and Aviation Fund of China

for the syntax and static semantics of OWL-S, and does not discuss the dynamic semantics.

These researches give good examples of formal specification of OWL-S and applications, but there are some problems:

(1) Some semantics is undefined: Because OWL-S is not a traditional language, so some properties can't be expressed directly. Some papers didn't claim this problem explicitly, but some do so, for example, in [5], "Precondition" and "Effect" can't be expressed in Promela and so ignored. In this paper, these properties are declared as important ones for processes and defined clearly.

(2) Transforming consistency: The structural operational semantics (SOS) of OWL-S hasn't been mapped to the specification language directly in related researches. This gives less consistency assurance for the transformation. But for rewrite logic, the mapping is rather clear. On the other hand, the later transformation for OWL-S specification only concerns syntax.

(3) Lack of analysis of dataflow: Among the above researches, only [5] explicitly stated the dataflow, it simplifies "Input" and "Output" as "Integer" and connects them to "channel". On the other hand, rewrite logic used in this paper is much more appropriate and efficient for properties deal not only with causality of events but also with data types or recursive constructs.

[6] and [7] use an algebra specification language Z. But the former focuses on the analysis of ontology, including some properties verification and reasoning without analysis of control flow and dataflow. The latter gives a good explanation for static semantics of OWL-S but without dynamic semantics.

3. Background

3.1 Introduction of OWL-S

OWL-S is an OWL-based (Recommendation produced by the Web-Ontology Working Group at the World Wide Web Consortium) Web service ontology, which supplies Web service providers with a core set of constructs for describing the properties and capabilities of their Web services in unambiguous, computer interpretable form. OWL-S markup of Web services will facilitate the automation of Web service tasks, including automated Web service discovery, execution, composition and interoperation.

It is the first well-researched Web Services Ontology, which has numerous users from industry and academe, and is still undergoing. Details of the latest version of OWL-S submission document can be referred to [7].

3.2 Rewrite Logic and Maude

Rewriting logic is a computational logic that can be efficiently implemented and that has good properties as a general and flexible logical and semantic framework, in which a wide range of logics and models of computation can be faithfully represented [8].

Definition 1: A rewrite theory R is a triple R = (Σ, E, R), with:

- (Σ,E) a membership equational theory, and
- R a set of labeled rewrite rules of the form: "$l : t \rightarrow t' \Leftarrow$ cond", with "l" as a label, t, t′ $\in T_{\Sigma}(X)_k$ for some kind k, and "cond" is a condition (involving the same variables X).

In general, a rule in rewrite logic is like:

$$l : t \rightarrow t' \Leftarrow \left(\bigwedge_i u_i = u_i'\right) \wedge \left(\bigwedge_j v_j : s_j\right) \wedge \left(\bigwedge_k w_k \rightarrow w_k'\right)$$

Maude is a formal programming language based on the mathematical theory of rewriting logic. With Maude system's support, this kind of language specifications can be executed and model can be checked, what is more, its mathematic semantics can be obtained. A program in Maude is just a rewrite logic theory, and Maude offers a comprehensive toolkit for the analysis of specifications, such as LTL model checker, Inductive Theorem Prover (ITP), Maude Termination Tool, Church Rosser Checker, Coherence Checker, etc.

In Maude, object-oriented systems are specified by object-oriented modules, defined by the keyword "*omod ... endom*", in which classes and subclasses are declared. A class declaration has the form *class* C|a_1: S_1, ..., a_n: S_n ,where *C* is the name of the class, the a_i are attribute identifiers, and the S_i are the sorts of the corresponding attributes. An object of a class *C* is defined as:< *O : C | a1 : v1, ... ,* a_n *:* v_n >, where *O* is the object's name, and the v_i are the corresponding values of object's attributes, for i=1 . . . n. Objects can interact in a number of different ways such as messages passing, messages between objects can be defined by the keyword "msg". Details of Maude can be referred to [9].

The main reasons why we choose rewrite logic to give the OWL-S specification the formal semantics are listed as follows, and more detail advantages can be referred to [10]:

- ♦ Consistency in transforming: Rewrite logic is a flexible and expressive one that unifies algebraic denotational semantics and structural operational semantics (SOS) in a novel way, which can be seamlessly transformed from OWL-S SOS into rewrite rules/equations [8], and suitable for describing data types and their relationships. On the other hand, the latter transforming from an OWL-S specification model into an algebraic one only concerns syntax.
- ♦ Suitable for many logic formula expressions in OWL-S: In [11], it is argued that rewriting logic is suitable both as a logical framework in which many other logics can be represented, and as a semantic framework.
- ♦ Efficiency implementation: Maude is a high performance rewriting logic implementations [12]. It is demonstrated that the performance of the Maude model checker "is comparable to that of current explicit-state model checkers" such as SPIN [11].

- ♦ Analyzing tools: It has been well established now that a rewrite logic specification can be benefited for comprehensive tool-supported formal verification [13].

4. Abstraction of the Model

There are three methods to transform an OWL-S specification into a rewrite logic one:

(1) Translate every lines in OWL-S specification into corresponding rewrite logic ones directly. The translating is the same as giving semantics to OWL-S specification.

(2) Give OWL-S (the language) rewrite logic semantics for every parts of its syntax. Then the OWL-S specification itself is a rewrite logic one with semantics. None transformation is needed.

(3) Abstract the main parts of OWL-S (the language), define syntax in rewrite logic for the sub-set and give semantics for the syntax. And then translate the OWL-S specification into a rewrite logic one by abstracting it into the syntax in rewrite logic.

Method (1) is direct, but difficult to ensure the consistency. Method (2) is a complete one. For example, a service is modeled by a process in OWL-S, some tags such as < *process: CompositeProcess rdf:ID="CP"*>, <*process: hasInput rdf:resource="#inputownright1"/*> are used to give a detailed perspective within a composite web service. All tags should have semantics (here the tags are "*process: CompositeProcess rdf: ID*" and "*process: hasInput rdf: resource*").

In this paper, method (3) is accepted. One reason to do so is that abstraction can reduce the complexity of analyzing a model; another reason is that we can go to the most difficult and challenge problems quickly.

In the following section, we will explain what has been abstracted from OWL-S, including control flow and data flow. This abstraction becomes a sub-set of OWL-S.

1) Parameters and Expressions: Parameters are the basis of representing expressions, conditions, formulas and the state of an execution. In OWL-S, parameters are distinguished as *"ProcessVar", "Variables" and "ResultVar"*, etc. They can even be identified as variables in SWRL. Our abstraction in this paper doesn't distinguish these, but refer them all as parameters.

Expressions can be treated as literals in OWL-S, either string literals or XML literals. The later case is used for languages whose standard encoding is in XML, such as SWRL or RDF. In this paper, expressions are separated into Arithmetic and Boolean expressions.

2) Precondition: If a process's precondition is false, the consequences of performing or initiating the process are undefined. Otherwise, the result described in OWL-S for the process will affect its "world".

3) Input: Inputs specify the information that the process requires for its execution. It is not contradictive with the definition of messages between web services, because a message can bundle as many inputs as required, and the bundling is specified by the grounding of the process model.

4) Result and Output: The performance of a process may result in changes of the state of the world (effects), and the acquisition of information by the client agent performing it (returned to it as outputs). In OWL-S, the term "Result" is used to refer to a coupled output and effect. Having declared a result, a process model can then describe it in terms of four properties, in which, the "*inCondition*" property specifies the condition under which this result occurs, the "*withOutput*" and "*hasEffect*" properties then state what ensures when the condition is true. The "*hasResultVar*" property declares variables that are bound in the "*inCondition*".

Precondition and Result are represented as logical formulas in OWL-S, but when they are abstracted, Boolean expression and assignment are used separately in this paper.

5) Process: A Web service is regarded as a process. There are three different processes: Atomic process corresponds to the actions that a service can perform by engaging it in a single interaction; composite process corresponds to actions that require multi-step protocols and/or multiple services actions; finally, simple process provides an abstraction mechanism to provide multiple views of the same process. We focus on atomic process and composite process here.

6) Control structure: Composite processes are decomposable into other (non-composite or composite) processes; their decomposition can be specified by using eight control structures provided for web services, including *Sequence, Split, Split-Join, Choice, Any-Order, If-Then-Else, Repeat-Until, and Repeat-While.*

7) Dataflow and Variables binding: When defining processes using OWL-S, there are many conditions where the input to one process component is obtained as one of the outputs of a preceding step. This is one kind of data flow from one step of a process to another.

A Binding represents a flow of data to a variable, and it has two properties: "*toVar*", the name of the variable, and "*valueSpecifier*", a description of the value to receive. There are four different kinds of valueSpecifier for Bindings: *valueSource, valueType, valueData,* and *valueFunction*. The widely used one "*valueSource*" is addressed in this paper.

The information listed above gives an overview of how web services are bound together with control structures and dataflows.

5. Syntax and Static Semantics in Maude

According to the method (3) described above, we now need to define how to express the information abstracted in Section 3 in rewrite logic, namely, syntax of the sub-set in Maude. Because of space limited, we only explain parts of it:

1) Parameters and Expressions: To express them, several rewrite logic modules have been defined. They are *NAME*, *EXP* and *BEXP*.

To specify process variables we define a module named "*NAME*", in which "*op_._: Oid Varname-> Name* "is defined to be the form "process.var" as a variable name, while *"Oid"* is the name of a process, which has been regarded as an object identification. And a *"NameList"* is used to be a list of variables.

The value of a variable is stored in a "Location" which is indicated by an integer. When we bind a location with a variable name, the variable get the value stored in that location.

Arithmetic expressions (sort name is *"Exp"*) and Boolean expressions (sort name is *"BExp"*) are defined separately in module *EXP* and *BEXP*, which gives a description of how to use variable names to describe expressions.

2) IOPR (Input/Output/Precondition/Result), Data flow and variable bindings:

"Input" and *"Output"* of a process are defined as *"NameLists"* which are attributions of a process.

In OWL-S, *"Precondition"* and *"Effects"* are represented as logical formulas by other languages such as SWRL. Here we first simplify Precondition as "BExp" to be an attribution of a process class.

"Result" is more complicated. After separate *"Output"* as an attribution of a process, *"Result"* combines a list of *"Effect"*, while every Effect is simplified as a conditional assignment here. The definition in Maude is " *op_<-_if_: Name Exp BExp -> Effect.*"

As discussed above, there are four types of binding *"valueSpecifier"*. Here we defined binding as "*op fromto* : *Name Name -> Binding* " to specify *"valueSource"* in module *WSTYPE*. With this definition, dataflow in a composite web service is created.

3) Processes and Control Structures: Atomic and composite web services are defined as two classes with different attributions. In order to distinguish definitions of *"ControlConstructList"* and *"ControlConstructBag"* for control structure, *"OList"* is defined to represent the object list which should be executed in order, and *"OBag"* to represent there is no order for the objects.

It seems very hard to express that a web service set can be executed in any order. But benefited with Maude operator attribution "comm", we can get this with definition "*op_#_: OBag OBag -> OBag [ctor assoc comm id: noo]*". *"comm"* attribution means that this "op" is with commutative property, which makes the objects in this "bag" ignore the order unlike it is in *"OList"*.

After defining two sorts as follows:

subsort Qid < Oid < Block < BlockList.
subsort Qid < Oid < Block < BlockBag.

We define a nested control structure. For example, *"sequence"* as "*op sequence: BlockList->Block [ctor]*" and "*split" as "op split : BlockBag -> Block [ctor]* ". This separates *"Block"* into three cases:

(1) Only a process.

(2) A group of processes within one control structure (we refer it as a control block).

(3) A group of processes and control blocks within one control structure.

Obviously, the (3) is a nested control structure. If the group is order sensitive, it is a *"BlockList"*, otherwise, it is a *"BlockBag"*.

Syntax of atomic web service: A class *"Atomws"* is defined in Definition 2. When an instance of atomic web service is created, it should be declared as an object of class *"Atomws"*.

Definition 2: *class Atomws | initialized : Bool, father : Oid, input : NameList, output : NameList, precondition : BExp, result : Effect .*

Syntax of composite web service: And a class *"Compositews"* is defined in Definition 3. We have explained "IOPR", "Result", "Precondition" and "Binding" above. Other attributions are: "initialized" to represent whether this instance object (composite web service) of the class has been initialized with actual values of its "IOPR", "Result", "Precondition", "Binding" and control structures. *"father"* denotes which composite web service (instance) it belongs to. *"struc"* is the control structure with *"BlockList"* and *"BlockBag"* as its subsort. Other attributions are defined to be used when the composite one is executed, especially for the nested control structures.

Definition 3: *class Compositews | initialized : Bool, input : NameList, output : NameList, precondition : BExp, result : Effect, binding : Binding, father : Oid, struc : CStruc, nest : BlockList, wait : OList, blockwait : NList, waitbag : OBag .*

When an instance of composite web service is created, it should be declared as an object of class *"Compositews"*. And prepare an initial equation for itself (how to define an initial equation is ignored here).

6. Dynamic Semantics in Maude

6.1 Auxiliary Modules

When "Precondition" of a process is true, it can be initialized and executed. It affects the "world" by various "Effect". So we need to define what the "world" will be for a web service. Here a module of *"SUPERSTATE"* is extended with *"CONFIGURATION"* which already defines as a "soup" of floating objects and messages in Maude.

A *"Superstate"* is the "world" of a process which defined as "*op_|_: State Configuration -> Superstate*". "*State*" is a group of variables with corresponding locations, and locations with corresponding values. A message is defined as "*msg call: Oid Oid -> Msg*" for a composite web service to trigger its sub-process to execute. Another is defined as "*msg tellfinish: Oid Oid -> Msg*" to tell its father that it has finished execution.

In module *"SUPERSTATE"*, assignment, evaluation of an arithmetic expression and a Boolean expression are defined, which gives semantics to how these syntax can be executed to affect the "world" of a process.

An operator *"k"* is defined as "*op k: Configuration -> Configuration*" to indicate that one web service is ready to be executed. Two operators *"val"* and *"bval"* are defined to evaluate expression and Boolean expression values in a state.

A sort *"NList"* (natural number list) is also defined in *"NLIST"* module to give semantics of executions of a nested control structure, with the help of the four attributions: *nest, wait, blockwait, and waitbag*.

6.2 Dynamic Semantics

In this section, we first analyze how executions of web services can affect their "world", and then by giving out the SOS for control structure, explain the corresponding rewrite logic rules or equations.

1) Execution of a Service:

◎Execution of an Atomic One

As defined in OWL-S, atomic processes have no sub-processes and execute in a single step only if the service is concerned and its precondition is true. The execution gives result to its "world" by "Effect". The main parts for its execution semantics (Figure 1) have been chosen to be explained as below.

Equation (1) asks atomic web service "ws" do initialization if its precondition *"Cd"* is true and hasn't been initialized before. Initialization is designed as an equation in a module of an instance of "Atomws". It prepares an initial state for this web service.

Equation (2) explains that when an atomic web service *"ws"* gets a message from its father *"F"*, it is the same meaning that it will be executed after initialized.

(1) ceq < ws : Atomws | initialized : init-state, father : F, precondition : Cd > call(F, ws)
= (initial(< ws : Atomws | initialized : true, father : F >) call(F, ws)) if (not init-state) and bval(Cd, st1) .
(2) eq st1 | (conf < ws : Atomws | initialized : true , father : F> call(F, ws)) = st1 |(conf k(< ws : Atomws | father : F >)) .
(3) eq st1 | (conf k(< ws : Atomws | father : F, result : (x1 <- exp1 if cond) Eff >))
= st1 | (conf k(< ws : Atomws | father : F, result : (x1 <- exp1 if bval(cond, st1)) Eff >)) .
(4) rl [execute] :
(mem(x1, location1) loc(location1, y1) st1) | (conf k(< ws : Atomws | father : F , result : (x1 <- y) Eff >))
=> (mem(x1, location1) loc(location1, y) st1) | (conf k(< ws : Atomws | father : F , result : Eff >)) .
(5) rl [finish] : st1 |(conf k(< ws : Atomws | father : F, initialized : init-state , result : noEff >))
=> st1 | (conf < ws : Atomws | father: F , initialized: false. result: noEff > tellfinish(F. ws)).

Figure 1. Semantics of execution atomic web service

Equation (3) explains that how to execute a condition inside an *"Effect"*. Of course there are rules that explain how to evaluate expression inside an *"Effect"* (ignored here).

The forth rule (4) simulates state changes by one *"Effect"*. And rule (5) ensure that only after all the *"Effect"* of this web service has been executed it tells its father it has finished, and prepares a same instance waiting for its *"Precondition"* to be true to be initialized to execute again.

◎Execution of a composite process

Composite web service changes its "world" by executing its sub-processes according to its control structures.

Different from atomic web service, before a composite one is going to be executed, it should prepare "binding" information. A rule below is used to explain how to do that. After that, "sourcedata" should be defined to affect the "world" by other rules.

rl st1 | (conf call(F, ws) < ws : Compositews | initialized : true, father : F, binding : fromto(v1 , v2) BD >)
=> (sourcedata(v1, v2, st1) st1) | (conf < ws : Compositews | father : F, binding : BD > call(F, ws)) .

After that, composite web service will be executed when its precondition is true like the atomic one. The difference is that the sub-processes grouped in control structures should be executed according to semantics of control. How to execute these structures will be explained below.

2) Sequence: The SOS of "sequence" and the corresponding rewrite logic rule are showed in Figure 2. "BLK" is a "Block" and "BL" is a *"BlockList"*. Obviously, attribution *"nest"* here is used to separate the *"BlockList"*, leaves the first one in *"struc"* to be executed first.

The question is how to ensure the first control block be executed firstly? Especially when it is a nested control structure-because when the most inner to be executed, the decomposing difference (order sensitive of *BlockList* and opposite *BlockBag*) should be recorded.

As discussed above in Section 4, a *"Block"* has three cases. But all of them should ensure that this *"Block"* be executed before *"BL"* is going to be executed for *"sequence"*. To ensure this, two attributions *"wait"* and *"blockwait"* are defined. *"wait"* is a *"OList (object list)"* to ensure that only after the list of objects are all finished that this service "ws" can be executed. *"blockwait"* is defined as *"NList (nature number list)"*. When an order

$$\frac{\langle BLK, \sigma\rangle \to \sigma'' \; \langle BL, \sigma''\rangle \to \sigma'}{\langle sequence\ (BLK; BL)\ \sigma\rangle \to \sigma'}$$

rl k(< ws : Compositews | father : F, struc : sequence(BLK ; BL), nest : BL1, wait : nilo, blockwait : BW >)
=><ws:Compositews|father:F, struc:BLK, nest : sequence(BL) ; BL1, wait: nilo, blockwait : (1 BW) > call(F, ws) .

Figure 2. Semantics of execution sequence

sensitive control block (here is sequence) is separated, it definitely asks the *"Block"* left in *"struc"* (here is *BLK*) should be finished before other "Block" left in "nest" are going to be executed. So we add natural number "1" into the *"NList"* (here is *BW*). Otherwise, "0" is added for no order sensitive one, such as for control structure "Split". And "2" will be added when the outer control structure asks order but this one doesn't.

After separating a control structure, there are three cases waited to be explained of how to execute *BLK*.

Case 1: *BLK* is a process.

In this case, if it is a composite one, it can be separated recursively. If it is an atomic one (here is *"A"*), different rules should be matched according to "blockwait" (here is *BW*). The value of head of *"BW"* is "1" means *"A"* should be completely finished before *"ws"* continues, showed as Figure 3 rule (1). So *"A"* is put into "wait", and "1" is added to *"BW"* of "ws" to indicate that this an order sensitive block. And then two messages are released to trigger *"A"* and *"ws"*. Of course, there should be a corresponding rule when *"A"* completes its execution (Figure 3 rule (2)).

If *head(BW)==0 and sum(BW) > 0* (rule (3)), then "2" is added to *"BW"* and *"A"* is added to "waitbag" (here is OO), this indicates that *"A"* need not to be executed firstly in this level of the control structure, but it need to be so in outer control structure. The corresponding rule to release the blocking is showed as rule (4).

Similarly if *head (BW) == 0 and sum(BW) == 0*, "0" is added to *"BW"* to indicate there is no need for *"A"* to be executed firstly.

```
(1)  crl : k( < ws : Compositews | father : F, struc : A ,
        wait :nilo, blockwait : BW > )
           => < ws : Compositews | father : F, struc : empty,
        wait : A ,
        blockwait: ( 1 BW) >    call(ws, A) call(F, ws)
        if head(BW) == 1 .
(2)  crl k( < ws : Compositews | father : F, struc : CS1,
        nest :BL, wait : A ~ L, blockwait : BW >)      tellfinish(ws, A)
       => (call(F, ws)     < ws : Compositews | father : F,
     struc : CS1,
        nest : BL , wait : L, blockwait : BW > )    if head(BW)
     == 1 .
(3)  crl :k( < ws : Compositews | father : F, struc : A ,
        wait : nilo, blockwait : BW, waitbag : OO > )
        => call(ws, A) call(F, ws) < ws : Compositews | father :
        F, struc : empty, wait : nilo, blockwait : ( 2 BW), waitbag : A # OO >
       if head(BW) == 0 and sum(BW) > 0 .
(4)  crl k( < ws : Compositews | father : F, struc : CS1,
        nest :BL, waitbag : A # OO, blockwait : BW >)      tellfinish(ws, A)
        => < ws : Compositews | father : F, struc : CS1, nest :
        BL , waitbag : OO, blockwait : BW > call(F, ws)
           if head(BW) == 2 or sum(BW) > 0 .
```

Figure 3. Semantics of execution nested control Block

$$\frac{\langle bexp1, \sigma \rangle \rightarrow false}{\langle repeat\text{-}while\ bexp1\ CS1, \sigma \rangle \rightarrow \sigma}$$

$$\frac{\langle bexp1, \sigma \rangle \rightarrow true \qquad \langle CS1, \sigma \rangle \rightarrow \sigma'' \ \langle repeat\text{-}while\ bexp1\ CS1, \sigma'' \rangle \rightarrow \sigma'}{\langle repeat\text{-}while\ bexp1\ CS1, \sigma \rangle \rightarrow \sigma'}$$

```
crl   k( < ws : Compositews | father : F, struc : repeat CS1
while bexp1, nest : BL, wait : nilo > )      =>
if bexp1
          then < ws : Compositews | father : F, struc : CS1, nest :
(repeat CS1 while bexp1) ; BL, wait : nilo >       call(F, ws)
          else < ws : Compositews | father : F, struc : empty,
nest : BL, wait : nilo > call(F, ws)
        fi.
```

Figure 4. Semantics of execution Repeat-while

Case 2 and Case 3: are similar with case1, but with recursive definition. Because of space limited, we will not discuss them here

3) Repeat-while: "Repeat-While" tests the condition, exits if it is false and does the operation if the condition is true, then loops. Its SOS and corresponding rewrite rule are showed in Figure 4.

Actually, for the control structure itself, it is not complex. The rule in Figure 4 just gives semantics of how this structure can be executed. The difficult is how "Block" can be executed inside it.

For example, when there is simple composite web service which contents only one atomic web service "A" within its repeat-while, say *(k(< ws : Compositews | father: F, struc: repeat 'A while bexp1, nest: BL, wait: nilo >))*, how about the execution of *"A"*?

As discussed for Definition 2 and 3, before this composite *"ws"* enters its execution *"k"* state, it should prepare an atomic instance *'A*. If the precondition of *'A* is true, it can be initialized and go into *"k"* execution state. This may affect the "world" of *"ws"* according to the group rules and equations in Figure 1. After 'A has finished its execution, rule (5) in Figure 1 prepares another instance *'A* waiting for its *"precondition"* again. If *"bexp1"* decides to execute *'A* again, execution can be continue, and the *"Result"* may turn *"bexp1"* to be true by affecting the "world" of *"ws"*.

4) Repeat-until: "Repeat-until" does the operation, tests for the condition, exits if the condition is true, and otherwise loops. Its SOS and corresponding rewrite rule are showed in Figure 5.

$$\frac{\langle CS1, \sigma \rangle \rightarrow \sigma' \quad \langle bexp1, \ \sigma' \rangle \rightarrow true}{\langle repeat\text{-}until\ bexp1\ CS1, \sigma \rangle \rightarrow \sigma'}$$

$$\frac{\langle bexp1, \ \sigma'' \rangle \rightarrow false \qquad \langle CS1, \sigma \rangle \rightarrow \sigma'' \ \langle repeat\text{-}until\ bexp1\ CS1, \sigma'' \rangle \rightarrow \sigma'}{\langle repeat\text{-}until\ bexp1\ CS1, \sigma \rangle \rightarrow \sigma'}$$

```
eq    k( < ws : Compositews | father : F, struc : repeat CS1 until
bexp1, nest : BL, wait : nilo > )    =
k(< ws : Compositews | father : F, struc : CS1, nest : (repeat
CS1 while not bexp1) ; BL, wait : nilo >           call(F, ws)).
```

Figure 5. Semantics of execution Repeat-until

Actually, Repeat-While may never act, whereas Repeat-Until always acts at least once. Other executions are the same.

5) Split: The components of a "Split" process are a bag of process components to be executed concurrently. Split completes as soon as all of its component processes have been scheduled for execution.

The rule below creates *"Block"* without order (because of split definition). At last these *"Block"*s produce atomic web services and messages into the "world" of "ws". Benefitted with objects concurrent execution in Maude, all web services that meet its precondition can be executed concurrently.

rl k(< ws : Compositews | father : F, struc : split(BLK @ BB), nest : BL, wait : nilo, blockwait : BW >)
=> < ws : Compositews | father : F, struc : BLK, nest : split(BB) ; BL , wait : nilo, blockwait : (0 BW) > call(F, ws).

6) Split-join: Here the process consists of concurrent execution of a bunch of process components with barrier synchronization. That is, "Split-Join" completes when all of its components processes have completed. To do this, a special object named "split-join" is defined, and then control structure "split-join(BB)" is equal to "sequence (split(BB) ; 'split-join)".

7) Choice: "Choice" calls for the execution of a single control construct from a given bag of control constructs. Any of the given control constructs may be chosen for execution. As discussed above, any *"Block"* inside the control bag may be chosen to match *BLK@BB*, because of commutative property. This gives a choice to the control bag. And then "0" is added to *"BW"* means there is no need waiting for this *"BLK"*.

rl k(< ws : Compositews | father : F, struc : choice(BLK @ BB), nest : BL, wait : nilo, blockwait : BW >)
=> < ws : Compositews | father : F, struc : BLK, nest : BL, wait: nilo, blockwait : (0 BW) > call(F, ws).

8) Anyorder: "Anyorder" allows the process components (specified as a bag) to be executed in some unspecified order but not concurrently. Execution and completion of all components is required. The execution of processes in an Any-Order construct cannot overlap, i.e. atomic processes cannot be executed concurrently and composite processes cannot be interleaved. All components must be executed. As with Split+Join, completion of all components is required.

rl k(< ws : Compositews | father : F, struc : anyorder(BLK @ BB), nest : BL, wait : nilo, waitbag: OO, blockwait : BW >)
=><ws:Compositews | father:F, struc: BLK, nest : anyorder(BB) ; BL, wait: nilo, waitbag: OO, blockwait : (1 BW) > call(F, ws).

9) If-then-else: The "If-Then-Else" class is a control construct that has a property *ifCondition*, then and else

$$\frac{<bexp1, \sigma> \to true \quad <CS1, \sigma> \to \sigma'}{< if\ bexp1\ then\ CS1\ else\ CS2, \sigma> \to \sigma'}$$

rl k(< ws : Compositews | father : F, struc : if bexp1 then CS1 else CS2, nest : BL, wait : nilo >) =>
if bexp1
then < ws : Compositews | father : F, struc : CS1, nest : BL, wait : nilo > call(F, ws)
else < ws : Compositews | father : F, struc : CS2, nest : BL, wait : nilo > call(F, ws)
fi.

Figure 6. Execution of if-then-else

holding different aspects of the If-Then-Else. Its semantics is intended to be "Test If-condition; if True do Then, if False do Else". Its SOS and corresponding rewrite rule are showed in Figure 6.

As discussed above, the rewrite logic rules are obviously consistent with the definition of SOS benefited from the great expressing capability of rewrite logic.

7. Case Study

Through the modules discussed above, we get a "semantics-OWL-S.maude" rewrite logic theory for semantics of the sub-set OWL-S. With this theory in hand, a software requirement or design in OWL-S can be abstracted into a rewrite logic theory with the syntax described above by extending this frame. Different from other translating methods directly mapping an OWL-S model into another specification language, this way avoids explaining the semantics in an actual model, translating work only concerns syntax mapping while semantics have been given in "semantics-OWL-S. maude".

For verifying the rewrite logic theory, we give an example to translate the process model to a Maude program, and undergo simple verifications [14] on it.

This example presented by OWL-S in Figure 7 is a web service based on Amazon E-Commerce Web Services. The process is to search books on Amazon by inputted keyword and create a cart with selected items, it is composed of four atomic processes through a sequence control construct.

Through the rewrite theory discussed above, we get a complete Maude program. Here we only display the main parts of the Maude program. Figure 8 is the initializing equation for the third atomic process "cartCreateRequest Process". In this module, we should build attributes to express process's IOPRs first. Two inputs and one output are translated name by name. Precondition and Effect in Result are translated to SWRL expression.

The main process of this example is a composite process, and the translated Maude code of initializing equation of it has been shown in Figure 9.

Load the Maude program, and then execute the process by the command "*rew execute-aws*", we can also search executing path using command "*search execute-aws =>! S: State|C: Configuration tell finish (', 'mainProcess*)." If input a right number less than or equal to the length of

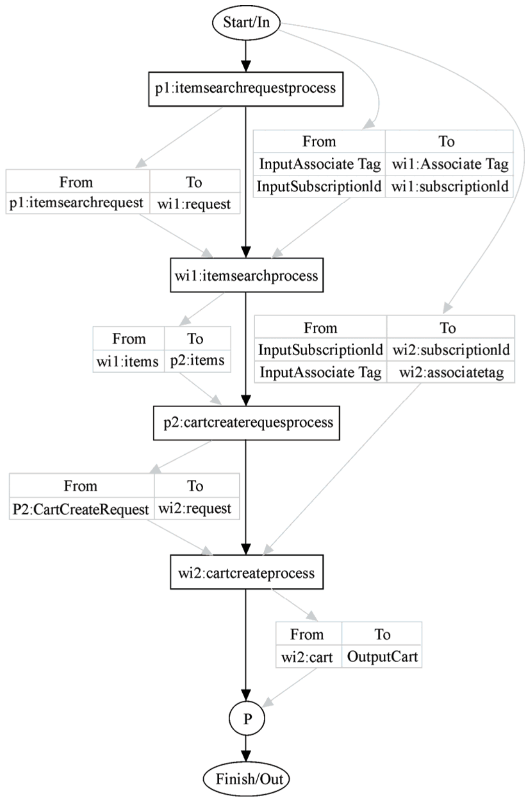

Start/In
p1:itemsearchrequestprocess
From | To
InputAssociate Tag | wi1:Associate Tag
InputSubscriptionId | wi1:subscriptionId
From | To
p1:itemsearchrequest | wi1:request
wi1:itemsearchprocess
From | To
wi1:items | p2:items
From | To
InputSubscriptionId | wi2:subscriptionId
InputAssociate Tag | wi2:associatetag
p2:cartcreaterequesprocess
From | To
P2:CartCreateRequest | wi2:request
wi2:cartcreateprocess
From | To
wi2:cart | OutputCart
P
Finish/Out

Figure 7. Structure of the web service process

```
vars ws F : Oid .
var state : State .
var attrs : AttributeSet .
var conf : Configuration .
var cartCreateRequestProcess : cartCreateRequestProcess' .

eq state | conf initial(<ws : cartCreateRequestProcess |
father : F,attrs>)call(F,ws)
=state | conf<ws : cartCreateRequestProcess |
initialized : true,father : F,input : ws.'index || ws.'items,
output : ws.'cartCreateRequest,
precondition : #not(swrlb:length(ws.'items) #< (ws.'index)),
result:noEff>call(F,ws) .
```

Figure 8. Initial equation of atomic process

'items' for 'index', the input of third atomic process "cart CreateRequestProcess", Maude will display the result in Figure 10. In other cases the result is like Figure 11.

We have done more works concerning this framework, because of space limitation, details are ignored:

(1) Test framework: although the directly mapping from OWL-S SOS to rewrite logic gives the consistency, some web services have been constructed to test the eight control structures and nested ones, including the execution of atomic web services. The results are the same as expected.

(2) Model checking and analysis: several cases are constructed including "philosopher dining" which not only concern control flow but also get a deadlock because of data sharing in the dataflow; and "online shopping" which concerns an error in dataflow. These errors can be found by the Maude analysis tools.

```
eq initial( < WS : MainProcess | father : F, attrs> )
    = < WS : MainProcess | initialized : true, father : F, input : ws .
    'InputAssociateTag || (ws . 'InputAssociateTag || (ws . 'InputSub-
    scriptionId,output : ws . 'OutputCart, precondition : true, result :
    noEff, binding : fromto( 'Perform_Search . 'SubscriptionId, ws .
                                                   'InputSubscriptionId)
 fromto('Perform_Search . 'AssociateTag, ws .'InputAssociateTag)
 fromto('Perform_Search . 'Request, ws . 'ItemSearchRequest)
 fromto('Perform_Search . 'Request, 'Perform_SearchReq . 'Item-
   SearchRequest)
 fromto('Perform_CreateReq . 'items, 'Perform_Search . 'Items)
 fromto('Perform_Create . 'SubscriptionId, ws . 'InputSubscrip-
   tionId)
 fromto('Perform_Create . 'AssociateTag, ws . 'InputAssociateTag)
 fromto('Perform_Create . 'Request, 'Perform_CreateReq . 'Cart-
   CreateRequest) ,
struc : sequence('Perform_SearchReq ; 'Perform_Search; 'Per-
   form_CreateReq; 'Perform_Create),
nest : null, wait : nilo, blockwait : niln, waitbag : noo >
 < 'Perform_SearchReq : itemSearchRequestProcess | initialized :
   false, father : ws,    precondition : true >
 < 'Perform_Search : ItemSearchProcess | initialized : false, fa-
   ther : ws, precondition : ture>
 < 'Perform_CreateReq : ItemSearchProcess | initialized : false,
   father : ws,    precondition : swrlb:length( 'Perform_CreateReq .
   'items) #>= ('Perform_CreateReq .'index) >
 < 'Perform_Create : CartCreateProcess | initialized : false, fa-
   ther : ws, precondition :    true>.
```

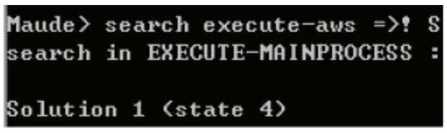
Figure 9. Initializing equation of composite process

```
omod EXEC-MAINPROCESS is
 pr MAINPROCESS .

 op MainProcess : -> MainProcess' .
 op execute-aws : -> Superstate .

 eq exec-MainProcess = loc(1, nilv) loc(2, 'SubscriptionId')
loc(3, 'AssociateTag') loc(4, nilv) loc(5, Keywords-1) loc(6,
SearchIndex-1) loc(7, nilv) loc(8, nilv) loc(9, nilv) loc(10,
index-1) loc(11, nilv) loc(12, nilv) loc(13, index-1)  loc(14,
nilv) loc(15, nilv) loc(16, index-1) loc(17, nilv) loc(18,nilv)
mem('mainProcess       .       'InputSubscriptionId,       2)
mem('MainProcess . 'InputAssociateTag, 3)
mem('Perform_SearchReq     .     'ItemSearchRequest,      4)
mem('Perform_Req          .          'Keywords,          5)
mem('Perform_SearchReq       .       'SearchIndex,        6)
mem('Perform_Search . 'Items, 7)
mem('Perform_Search          .'OperationRequest,          8)
mem('Perform_Search . 'Shared, 13)
mem('Perform_Search . 'Validate, 14) mem('Perform_Search .
   'XMLEscaping, 15) mem('Perform_CreateReq . 'CartCre-
      ateRequest, 9) mem('Perform_CreateReq . 'index, 10)
    mem('Perform_Create . 'Cart, 11) mem('Perform_Create .
  'OperationRequest, 12) mem('Perform_Create . 'Shared, 16)
 mem('Perform_Create . 'Validate, 17) mem('Perform_Create .
  'XMLEscaping, 18) | < 'MainProcess : MainProcess | initial-
          ized : false, father : ', precondition : true >
   call(' , 'MainProcess).
endom
```

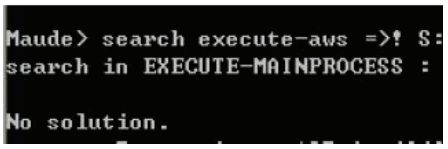
Figure 10. Executing environment module



```
Maude> search execute-aws =>! S
search in EXECUTE-MAINPROCESS :

Solution 1 (state 4)
```

Figure 11. Process can finish



```
Maude> search execute-aws =>! S:
search in EXECUTE-MAINPROCESS :

No solution.
```

Figure 12. Process can not finish

8. Conclusions

This paper gives a formal semantics for OWL-S sub-set by rewrite logic, including abstraction, syntax, static and dynamic semantics. Compared with related researches, the contribution of this paper gives a translation consistency and benefited with formal specification, dataflow can be analyzed deeply, which makes formal verification, and reliability evaluation of software based on SOA possible.

The undergoing future works include: “Precondition” and “Effect” in SWRL format; WSDL and grounding information; and a more complex application analysis.

9. Acknowledgement

This work has been greatly helped by Prof. Meseguer. Thanks to Michael Katelman, Feng Chen and Joe Hendrix in Formal Systems Lab of UIUC, and all the developers of the shared software.

REFERENCES

[1] X. Fu, T. Bultan, and J. W. Su, “Analysis of interacting bpel web services,” in Proceedings of the 13th International Conference on World Wide Web, New York, NY,USA, pp. 621–630, May 2004.

[2] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. R. Payne, E. Sirin, N. Srinivasan, and K. Sycara, “OWL-S: Semantic markup for web services,” Technical Report UNSPECIFIED, Member Submission, W3C, http://www.w3.org/Submission/ OWL-S/, 2004.

[3] H. Huang, W. T. Tsai, R. Paul, and Y. N. Chen, “Automated model checking and testing for composite web services,” in Proceedings Eighth IEEE International Sympo sium on Object-Oriented Real-Time Distributed Com puting, Washington, DC, USA, pp. 300–307, May 2005.

[4] S. Narayanan and S. A. Mcllraith, “Simulation, verification and automated composition of web services,” in Proceedings 11th International Conference on World Wide Web, Honolulu, Hawaii, USA, pp. 77–88, May 2002.

[5] A. Ankolekar, M. Paolucci, and K. Sycara, “Spinning the OWL-S process model-toward the verification of the OWL-S process models,” in Proceedings International Semantic Web Conference 2004 Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications, Hiroshima, Japan, 2004.

[6] H. H. Wang, A. Saleh, T. Payne, and N. Gibbins, “Formal specification of OWL-S with Object-Z: The static aspect,” in Proceedings IEEE/WIC/ACM International Conference on Web Intelligence, Washington, DC, USA, pp. 431–434, November 2007.

[7] J. S. Dong, C. H. Lee, Y. F. Li, and H. Wang, “Verifying DAML+OIL and beyond in Z/EVES,” in Proceedings the 26th International Conference on Software Engineering, Washington, DC, USA, pp. 201–210, May 2004.

[8] T. F. Serbanuta, F. Rosu, and J. Meseguer, “A rewriting logic approach to operational semantics (Extended Abstract),” Electronic Notes Theoretical Computer Science, Vol. 192, No. 1, pp. 125–141, October 2007.

[9] H. Huang and R. A. Mason, “Model checking technologies for web services,” in Proceedings the 4th IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance, Wanshington, DC, USA, pp. 217–224, April 2006.

[10] A. Verdejo and N. Marti-Oliet, “Executable structural operational semantics in maude,” Journal of Logic and Algebraic Programming, Vol. 67, No. 1–No. 2, pp. 226–293, April–May 2006.

[11] M. Birna van Riemsdijk, Frank S. de Boer, M. Dastani, and John-Jules Meyer, “Prototyping 3APL in the maude term rewriting language,” in Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, Hakodate, Hokkaido, Japan, pp. 1279–1281, May 2006.

[12] M. Clavel, F. Duran, S. Eker, P. Lincoln, N. M. Oliet, J. Meseguer, and C. Talcott, “All about maude-A high-performance logical framework,” Springer-Verlag New York, Inc., 2007.

[13] J. Meseguer and G. Rou, “The rewriting logic semantics project,” Theoretical Computer Science, Vol. 373, No. 3, pp. 213–237, April 2007.

[14] M. Clavel, F. Duran, etc., “Maude mannual,” Department of Computer Science University of Illinois at Urbana-Champaign, 2007, http://maude.cs.uiuc.edu.

J. Software Engineering & Applications, **2009, 2:34-39**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

Call for Implementation: A New Software Development Mode for Leveraging the Resources of Open Community

Weiping Li[1], Weijie Chu[1], Ying Liu[*]

[1]School of Software & Microelectronics, Peking University, Beijing 100871 China, *IBM China Research Lab, Beijing 100094, China
Email: wpli@ss.pku.edu.cn, chuwj@ss.pku.edu.cn, aliceliu@cn.ibm.com

Received December 31st, 2008; revised February 6th, 2009; accepted March 10th, 2009.

ABSTRACT

With the growth of the internet and open software, there are additional software developers available from the open community that can participate in the development of software application systems. Aiming to leverage these resources, a new development model, CFI (call for implementation), is proposed. The basic idea of CFI is to publish some part of a software project to the open community, whole or part, in certain phases of the software development lifecycle to call for implementation. This paper discusses the basic concept and method for a software development process in CFI mode. Two different modes of CFI with different granularities are analyzed. And one of the CFI modes, fine-granularity-CFI mode, is thoroughly discussed including the main methods and basic steps. To verify the ideas a pilot project, an online store system, is built up with the CFI development process. The online store system takes the traditional Model-View-Control architecture and some common technologies such as Struts, Hibernate, Spring are used. The result shows that this new kind of software development mode is feasible though there are many problems that are still requiring further study.

Keywords: *Call for Implementation*, *Software Development*, *Open Community*, *Integration*

1. Introduction

Open community, which is composed of students, programming fans, SOHO (small office/home office), etc., is a virtual development resource pool [1]. With the development of open source software, open community is now booming, which contributes many new ideas and solutions to some frameworks and common solutions.

Currently there are many open source software programs available in many web sites such as Apache, SourceForge, and many others. The Apache Software Foundation provides support for the Apache community of open-source software projects [2]. SourceForge.net provides free hosting to open source software development projects with a centralized resource for managing projects, issues, communications, and codes [3]. Besides the open source software, open community provides additional software development resources that, when properly compensate financially, can do the coding work for certain software components. For example, TOP CODER [4] provides a platform for the open community developers on which some software components are available for implementation. There are even small-size project provided on ScriptLance [5]. On SXSoft [6], a Chinese website, both small software components and small sized projects are available. And this new mode of development occurred only one or two years ago.

As we can see, there already exist some practices on using the resource of open community for coding. Though currently the components that done by the open community developers are small size and the main tasks are coding, it is really a new phenomenon which inspires us to leverage the rich resources of the open community. To follow this idea and practice, some companies are going beyond outsourcing small components to outsourcing the whole project. In this whole project outsourcing mode, we need some new methods and technologies to manage the whole project. We call this new mode is Call for Implementation (CFI). Fortunately there are some research efforts on outsourcing [7] and open source software development model [8]. And there are also some previous research efforts and practices on CFI at IBM China Research Lab, where some researchers finished a CFI project in the popular SOA architecture [1,9].

Sponsored by IBM China Research Lab, we, Peking University, have been working on a joint study CFI project, in which some methods for requirement analysis, designing and partitioning the project have been studied. This paper describes the basic ideas and methods for CFI and a practice that verifies the ideas in a pilot project.

The rest of this paper is organized as follows: Section 2 describes the basic ideas of CFI and the two different modes for CFI as well as some extra work which the project owner should take into account. Section 3 depicts in details the ideas, steps, and methods for one of the CFI model, i.e. fine-granularity-CFI mode. Section 4 puts forward a pilot project finished in the fine-granularity-CFI mode. Section 5 concludes the paper and presents the future work.

2. CFI Overview

2.1 The Concept of CFI

CFI is a new kind of development style that is different from traditional software development within an organization. Under the CFI mode, usually the owner of a certain software development project will publish some tasks on their website, or a specific platform, to call for implementation. The task here means a piece of work with a reasonable granularity such as a java class, a component, or a JSP page that can be implemented by a single developer. And usually a task is a package that is composed of different kinds of the artifacts. After finishing the tasks the developers will submit them back to the owner. The project owner then continues the testing and integration work. After all the necessary work done, the whole system is built up. The conceptual model of CFI is depicted in Figure 1.

Given this idea, we can tell the differences between CFI development and conventional development. In the normal software development lifecycle, there are the basic tasks such as requirement analysis, general design, detailed design, implementation, testing, and maintenance. Usually the process used in a certain software system development is one of the following models: Waterfall Model, Spiral model, RUP [10], etc. Whereas in the CFI model some extra work should be added to implement this new kind of development, namely, publish and submission. These two tasks can be added into certain phase of the process according to the policy of CFI. In the traditional process it is relatively easy to get feedbacks from different roles and the iterative process can be used. But in the CFI situation, it is hard to get any feedback and accordingly hard to take the iterative process.

There are two basic modes for CFI process, coarse-granularity-CFI and fine-granularity-CFI mode, which can be seen in Figure 2 and 3.

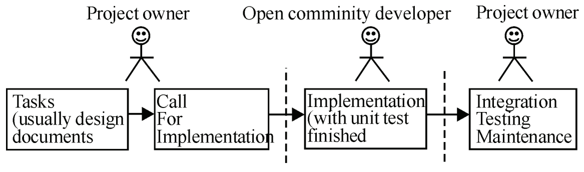

Project owner
Open comminity developer
Project owner
Tasks (usually design documents
Call For Implementatio
Implementation (with unit test finished
Integration Testing Maintenance

Figure 1. The conceptual model and process of a CFI project

Figure 2 shows the coarse-granularity-CFI mode, in which the publisher, i.e., the project owner, just publishes the requirement to the open community while the developer will perform further tasks such as design, coding, and testing. In the coarse granularity mode, usually the project owner should firstly design the structure of the database and the architecture of the software and determine the technologies used in the project. The developers should work under these constraints in order to achieve a smoother integration among the individual tasks submitted by the open community developers. For each task, the implementation includes the design, coding and unit testing in this mode.

Figure 3 shows the fine-granularity-CFI mode, in which the publisher will finish the designing work and publish the tasks which contain some part of the design artifacts. The implementation in this mode includes coding and partial unit testing.

2.2 The Main Challenges of CFI

As compared with the traditional software development process, there arise some challenges for the CFI development process that the project owner should balance among them. The main objective of CFI is to leverage the development resources in the open community. When taking advantage of this potential resource, the project owner has to face some extra effort for testing, integration, and potential quality decline. So the following issues should be taken into account in managing for CFI project.

• Information concealment: When publishing the tasks to the public, all the information that is not related to a certain task should be concealed. Only the needed information about the task itself will be published.

• Granularity: Fine granularity means the number of tasks will be increased accompanied by the difficulty of integration test, while the coding task itself is easy to implement. On the other hand, relative coarse granularity means single task will be hard to implement while decreasing the difficulty of integration and testing.

• Testing policy: Partitioning the whole project into parts creates another challenge: testing. And the main difficulty comes from the unit testing. Usually a class in a certain task may link with other class(es) in other task(s) and different tasks will be done by different people. So it is impossible for the classes in different tasks to run together until at the integration phase done by the project owner. The developer can write the unit test cases but it is hard to run them.

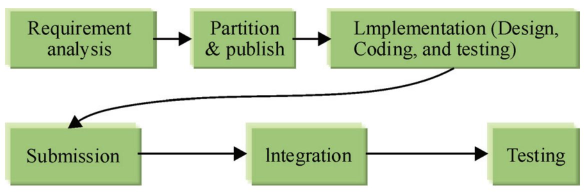


Figure 2. The coarse-granularity-CFI development process

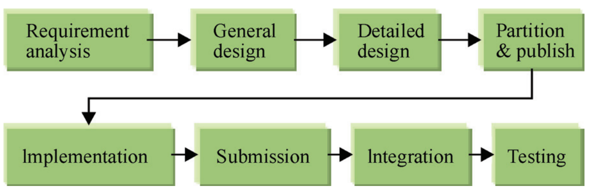


Figure 3. The fine-granularity-CFI mode development process

• Integration: The codes are from different developers and those developers are usually not available in the integration phase. So how to ensure the code quality is another big challenge, which the project owner and the core team have to face the challenge.

3. The Partition Methods for CFI Project

There are two CFI modes according to the granularity of tasks. Currently we are studying the fine-granularity-CFI mode and the corresponding partition method and rules for CFI development. Also some pilot projects have been done to verify the ideas of CFI. The study for the coarse granularity mode remains to be study.

For the fine-granularity-mode, the project owner should firstly finish the detailed design and then partition the artifacts into small pieces of tasks that will be implemented by different people. Suppose the designing work is as usual. To publish the tasks on the platform in this mode, we start from the design document, and focus mainly on the class diagram, which provides the foundation for further partitioning of the whole project. With the relationships among classes the big picture of the project can be obtained. To assure that all the development tasks can be included into the big picture and ready for further partition, the GUIs of the project, e.g. JSPs, should be included into the big picture, which is actually an extended class diagram.

The question now is how to partition the big picture, i.e. the extended class diagram, into tasks. For this purpose, some rules can be introduced to help the partition. Next we will use an example to describe the methods. Figure 4 shows a very simple big picture with the Java Script Pages (JSPs) included as classes. This class diagram follows the common Model-View-Control architecture. The nodes in the figure are classes or JSPs and the arcs indicate the relationship among the nodes. There are four kinds of nodes, namely, JSP, Action, Business Service, and Data Access Object (DAO). JSP stands for the GUI. Action stands for the Action Servlet in the Struts framework. Business Service represents the classes that fulfill the real business processes and functions. And DAO is the class for accessing the database. Each node gets a flag that indicates its type, e.g. JSP, ACTION, SERVICE, or DAO.

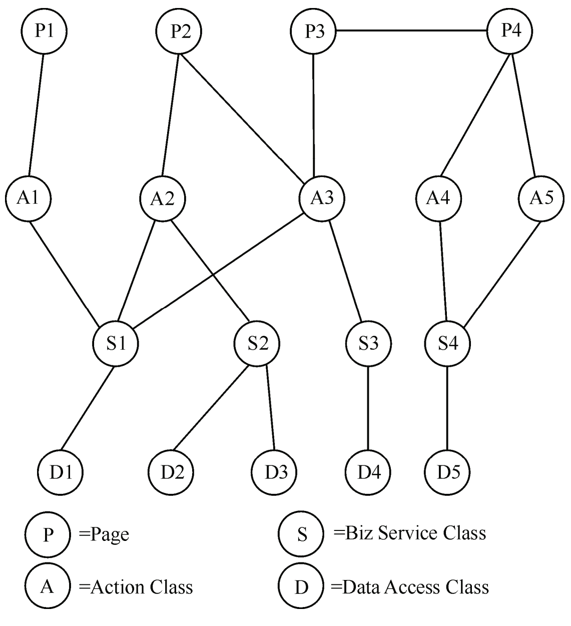


Figure 4. The extend class diagram for CFI partition

To split the whole picture into parts, namely tasks, some rules will be followed at the beginning of the partitioning process by the project manager or the architect. There exist some basic rules:

• A node (with certain type defined) can be placed into one task with the node(s) (with another type defined) linking to it.

• A node (with certain type defined), although having relations with others nodes, can be partitioned into one task.

• For the isolated nodes that have no arcs linked with any other nodes will be naturally in a task.

• A node can be placed into ONLY ONE task.

Besides these rules, when in actual situations, some other rules can be introduced when needed.

For the class diagram in figure 4, let's define the following rules:

1) The nodes with type JSP and nodes with type ACTION that are linked together will be grouped into one group.

2) The node with type SERVICE or DAO forms one single group.

Conforming to these rules, the class diagram can be grouped into thirteen groups, i.e. tasks. They are:

(P1, A1), (P2, A2, A3), (P3), (P4, A4, A5)
(S1), (S2), (S3), (S4)
(D1), (D2), (D3), (D4), (D5)

Because both A2 and A3 are linked with P2, they are in the same group. Although A3 is linked with P3, P3 itself forms a group because A3 is already in another group. Once all the groups are determined, they are ready to be published to the open community as tasks.

Before publishing these tasks to the open community to call for implementation, there is still some extra work that should be done. For example, usually the database design, some common interfaces and classes, the Cascading Style Sheets (CSS) of JSPs and other necessary building blocks need to be delivered to the open community developers.

4. A Pilot Project

To verify this new development mode, we started a pilot project using the CFI methods. The project is to develop an online store application system, using the fine-granularity-CFI mode. There are seven modules in this application system and we select on of the simple modules as the example to explain the CFI mode. Before this pilot project, a platform supporting CFI development was already developed by another team at Peking University under Professor Yu Lian [11]. The platform provides the necessary functions for the CFI development such as task publishing, payment, project management and so on. We use this platform to partition the tasks, to publish and receive the finished tasks, and to monitor the project.

4.1 The Online Store Project Overview

There are two teams involved in this project, one core team and one development team. The core team works on requirement analysis, detailed design, partitioning, publishing tasks, integration, and testing, while the development team works on coding and unit testing. There are eight persons in the core team: two teachers act as project managers, one senior graduate student as the architect and five junior graduate students as designers. A total of 45 students are in the development team that works on coding and unit testing. The 45 students here simulate the open community developers. Currently the integration is finished and next work is to test the system.

The time table is as follows:

- Oct. 15-Nov. 15, Requirement Collection,
- Nov. 19-Dec. 10, Detailed Design,
- Dec. 11-Dec. 12, Partition and Publishing,
- Dec. 13-Dec. 21, Coding and Unit Testing,
- Dec. 24-Dec. 29, Integration
- Feb. 18-.Mar. 30, Testing

There are seven modules in this online store application system, i.e. Customer Management, Shopping Cart, Sales, Procurement, Inventory Management, Consignment, and System Management. The system adopts the Model-View-Controller architecture and is developed in Java. The development toolkit is Rational Software Architect, Rational Application Developer 6.0 and the server is WebSphere Application Server 6.0. The main technologies are Struts, Hibernate, Spring, JUnit, etc.

4.2 An Example for Partition

Let's take the procurement module as an example to demonstrate the partitioning process in CFI. Procurement module is a simple module that is responsible for the procurement of the merchandise to be sold on the online store. This module follows a standard procurement process, including creating a purchase requisition, examining and approves a purchase requisition, creating a purchase order, examining and approving a purchase order, getting manifest, and warehousing. The class diagram is shown in Figure 5. There are thirteen JSPs, seven Actions, three business services and three DAO class in the extended class diagram.

Rules used

There are four rules used in this pilot project.

1) Every business service will be in a single group

2) Every DAO class will be in a single group

3) A JSP and its linked Actions will be grouped together.

4) If one action class has more than one linked JSPs then this action class will be grouped with one of the JSPs randomly.

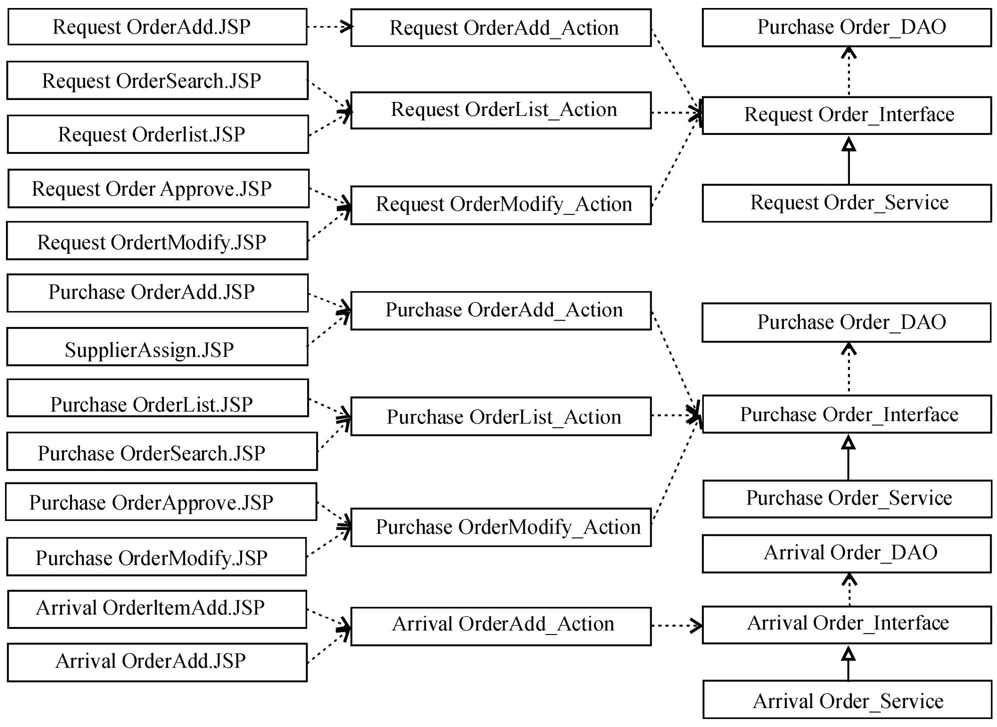


Figure 5. The extended class diagram for procurement module

The partition result
According to the rules the procurement module was divided into 16 groups, i.e. 16 tasks, see table 1. The three business services are grouped as single tasks according to rule No. 1, see task No. 14, 15, and 16. All the other JSPs and Actions are divided into the other 13 groups.

From Figure 5 we can see that ArrivalOrderAdd_ Action links with both ArrivalOrderAdd.JSP and ArrivalOrderItemAdd.JSP. According to rule No. 4, they are placed in group 12 and 13 respectively.

The DAOs in Figure 5 can be generated automatically by the Hibernate tool so they are not included in the tasks. That means they are finished by the core team. Also there still some other classes those were not included in the tasks and hence not published to the open community.

Before publishing the tasks on the platform, two extra efforts should be made. One is to package the right parts of design document with the tasks, which will help the developers to understand the design. Another is to build up the project framework and the some components, such as DAOs mentioned above, will be provided in the development toolkit, i.e. Rational Application Developer. The components in the project include the package and class name, the jar files that are imported into the project, some common interfaces and tools, and so on. This would provide all the developers the same working environment.

Additionally, the database design should be finished by the core team before the publishing or at least before the integration phase.

4.3 Analysis of the Practice

Compared with the traditional software development process, the CFI mode does have following different characteristics.

Integration
The integration is done by the core team. Before publishing the tasks the framework of the project is already built so the integration is relatively easy. Integration is simply put all the code on the right place, i.e. right package and right directories. The code may be finished by the open community developers or by the core team members.

Table 1. The partition result

1	Purchase 0rderAppmve JSP	9	Request 0rderModify JSP Request 0rder Modify_Action
2	Request 0rderModify JSP Request OmderModify_ Action	10	Request 0rdzer List JSP
3	Purchase 0rderList JSP	11	Purchase 0rderSearch JSP Purchase 0rderList_ Action
4	Request 0rderSearch JSP Request 0rderList_Action	12	Arriwal 0rderItmAdd JSP
5	SupplierAssign JSP	13	Arriva0rderAdd JSP Arrival0rderAdd_Action
6	Request0rderAdd JSP	14	Reques0rder_Service
7	Reques0rderAppnove JSP	15	Purchase0rder_Service
8	Purchase0rderAdd.JSP Purchase0rderAdd_Action	16	Arrival0rder_Service

The configuration file used in the project, namely Spring and Hibernate configuration files, shall not be open to all the developers for the purpose of information concealment. But usually there exist one configuration file for the whole project. In this project we cut the file into pieces according to different tasks' needs. This raised the extra work for cutting the file and reuniting the pieces into one integrated file.

Testing
The developers have finished the coding and, accordingly, have written the test cases for unit test using JUnit methods. But mostly unit test cases cannot run until the integration is finished because every single developer can only get a task, which may have to call other interfaces that will be implemented by other developers. Although they can use surrogate(s) to help finish the testing, it must be retested after integration. The core team will continue to complete the test wok. During this period, it will be lucky if we can find the developers to correct any errors, which we suppose won't do it because they have been already paid. Even if we are lucky, we can find the right persons, it is still hard to correct some errors which have relationships with other modules.

Code quality
Usually the code style is hard to control. The project owner may define the code style for the project. But it is hard to ensure all the open community developers abide by it well. Because the unit test can not be complete before the integration it is also difficult to ensure the quality of the code.

The code quality is deeply depended on the skill and experience of the programmer. This is a risk for the success of the project.

Surely the quality of the design document will impact the quality the code. And the partitioning work adds extra difficulty to the design documents.

5. Conclusions

This paper analyzes the basic ideas of Call for Implementation, which can help an enterprise to leverage the resources in the open community. The basic methods for CFI are introduced. Especially the partition method for dividing the class diagram into tasks is introduced. Based on this new idea and corresponding methods, a pilot project has been developed in the so called fine- granularity-CFI mode aiming to verify the methods.

From the pilot project we have found some interesting points that differentiate CFI from traditional software development. Some extra work is needed for the CFI development such as partitioning the design work, publishing the tasks, integrating the artifacts, and preparing the project framework.

1) In CFI mode, the coding work is finished by the open community programmers while the unit test will be

done by the core team. And there arise the potential risk of lowered quality and even more workload for correcting the errors.

2) Iterative development is regarded as a useful process. But in CFI mode it is hard to send any feedback to the developers. So it is hard to use this kind of process.

3) In CFI mode every task is a group of classes that may have some natural relationships with other task(s). So the information of the project is hard to be concealed. The developers must know the interfaces that this task calls. If someone purposely collects all the tasks in some way then he will discover the functions, interfaces, and even the architecture of the project.

4) Based on the previous practice we find two useful principles for partitioning, which we believe can improve the quality of the code. One is that the nodes in the extended class diagram that fall into one use case will be divided into one task. In this way the whole business process of a certain function will be encapsulate into one task so it will be done by one single programmer. This eliminates the communication among different programmers and hence improves the quality of the code. The other is that the controller classes in terms of Model- View-Control, namely the main process, shall be assigned to the core team. This will help conceal the business process and also improve the quality of the code.

Still some issues are not covered in this pilot project such as the information concealment and quality assurance, which need further study and practice. Or you can find some valuable information and analysis in paper [9].

6. Acknowledgement

This research is funded by the IBM China Research Lab and the China 863 project (No. 2007AA04Z150).

REFERENCES

[1] Y. Liu, C. H. Feng, W. Zhao, H. Su, and H. H. Liu, "A case study on community-enabled SOA application development," IEEE International Conference on Service-Oriented Computing and Application, Newport Beach, California, June 19–20, 2007.

[2] http://www.apache.org/.

[3] http://sourceforge.net/.

[4] http://www.topcoder.com.

[5] http://www.scriptlance.com/.

[6] http://www.sxsoft.com/.

[7] S. L. Huff, "Outsourcing of information services," Business Quarterly, pp. 62–65, 1991.

[8] T. Oreilly, "Lessons from open-source software development," Communications of the ACM, Vol. 42, pp. 32–37, 1999.

[9] X. Zhou, Y. Liu, *et al.*, "Case study: CFI-enabled application development leveraging community resource, 2008 international conference on Service Science, April 17–18, Beijing.

[10] P. Kruchten, "Rational unified process-an introduction," Addison-Wesley, 1999.

[11] A private discussion with Professor Yu Lian from Peking University.

J. Software Engineering & Applications, **2009, 2:40-43**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

Three-Tier Knowledge Management System Based on .NET

Mingxing Cai[1], Jintao Zheng[2], Ping Shi[1], Xiaohui Li[1]

[1]Facility Division, China University of Geosciences, Wuhan, China, [2]School of Software Engineering of Huazhong University of Sciences and Technology, Wuhan, China
Email: zheng.j.t@163.com

Received January 19th, 2009; revised February 20th, 2009; accepted March 24th, 2009.

ABSTRACT

Three-tier knowledge management system based on .NET architecture is designed according to requirement specification, characteristics of and relationship between enterprise electronic archives and knowledge management. This system using three-tier design based on factory pattern has good encapsulation and portability, with clearer and more concise structure. It degrades the costs of system development and maintenance and upgrades system's high reusability and development efficiency.

Keywords: *Knowledge Management*, *.NET*, *Three-Tier Architecture*, *Design Based on Factory Pattern*

1. Introduction

With the advent of the era of knowledge economy and increasing competition among enterprises, more and more enterprises are aware of the importance of the knowledge management to their development [1]. How to refine business knowledge in complex external and internal information, to effectively manage the acquisition, production and proliferation process of all kinds of enterprise knowledge, and to enhance knowledge-based capabilities, is the key to business survival. Broadly speaking, knowledge management is a positive approach to manage and optimize the enterprise's knowledge resources [2]. The goal of knowledge management is to pass on the most appropriate knowledge to the most appropriate person at the most appropriate time. It can give managers great support to make the best decisions [3]. At present, domestic researches on the enterprise knowledge management focus more on the theoretics than the practice [4]. Therefore, it is of great practical significance to design a set of easy-to-use and easy-to-promote knowledge management system, making use of the existing and somewhat general information technology, for promoting knowledge management in its application and development for the enterprise.

This paper researches and analyzes knowledge management in-depth theoretically and practically, using Microsoft .NET technology platform, to design and develop knowledge management system solutions suitable for enterprise applications, in the hope of promoting the widespread and in-depth use of knowledge management in enterprise applications.

2. Design for Three-Tier Knowledge Management System Based on .NET

The architecture design for this system is shown in Figure 1.

2.1 System Architecture Design

This system uses .NET platform based B/S three-tier architecture. B/S structure can lower the configuration requirements for the client. Three-tier architecture is a new method of programming, divided into the presentation tier, the business logic tier and the data access tier [5]. An advantage of three-tier architecture is its separation of data and format, with which programs can be easily expanded and modified to improve the coding reusability, once the business requirements are changed.

The presentation tier, using ASP.NET dynamic Web pages/sites techniques, provides users with the interoperability of the Web access interface, improving the precompiled operating mechanism, enhancing the security and guaranteeing good performance for the system.

The business logic tier contains almost all the processing functions of system business logic, which is a bridge between the data access tier and the presentation tier. Thus, the robustness, flexibility, reusability, scalability and maintainability of the software system, to a large extent, depend on the design of business logic tier [5]. In this paper, the business logic tier based on .NET platform encapsulates components such as the approval flow, business entities, statistical analysis, etc.

The data access tier offers centralized visits to the database to ensure good encapsulation and maintainability [6]. The business logic tier interacts with the database through the data access tier to avoid directly relying on the database. The data access tier, synthesizing the factory

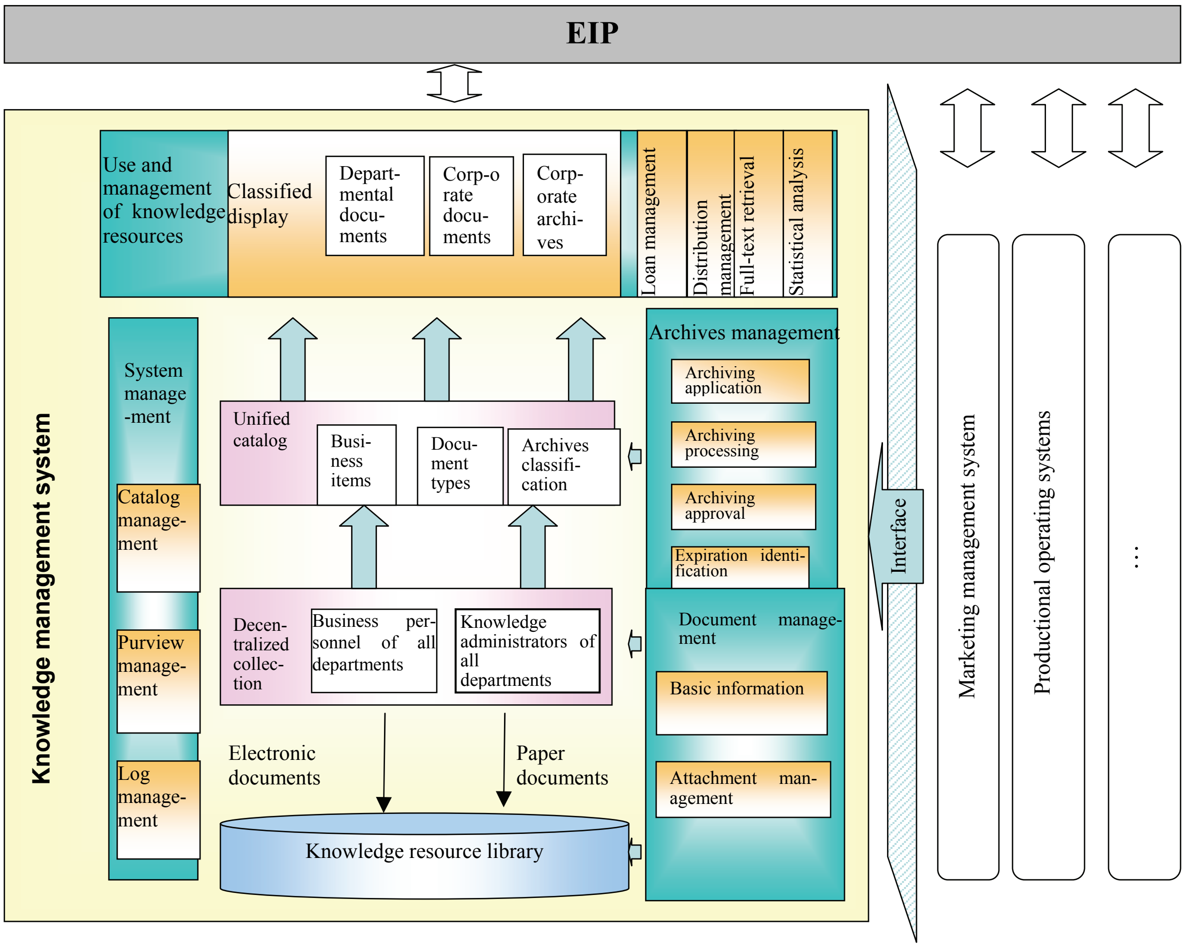


Figure 1. Architecture of knowledge management system

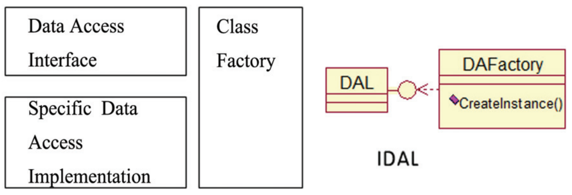


Figure 2. Data access tier based on factory pattern

design pattern and the reflex mechanism on .NET platform, makes itself more flexible, as shown in Figure 2. It really fulfills the characteristic that the data tier is unrelated to the upper tiers. That is, whatever data storage is using, MS SQL Server database, Oracle or XML documents, as long as the data is accessed through a certain interface, there is no need to revise other upper tiers and to recompile the whole system.

2.2 System Function Design

Overall flow of three-tier knowledge management system based on .NET is as shown in Figure 3. Four main functional modules are briefly illustrated as follows.

1) Document management: It mainly includes two functions of the basic information management and appended documents management, which requires business personnel of all departments to upload internally- generated documents into the knowledge management system in time.

2) Archives management: It includes four sub-modules of archiving management, expiration identification, departmental archives management and corporate archives management.

- Archiving management includes three functions of archiving application, archiving processing and archiving approval.
- Expiration identification includes three functions of expiration reminding, documents destruction and documents postponement.
- Departmental archives management module is to display, using views, the archiving application information to be processed by the departmental knowledge administrators and the archives information submitted to the corporate knowledge administrators for approval, including two schedule views of above-mentioned information. Departmental administrators have access to viewing.
- Corporate archives management is to display, mainly using views, the archives information required to be approved by the corporate knowledge administrators and the archives information already approved by them. Corporate administrators have access to viewing.

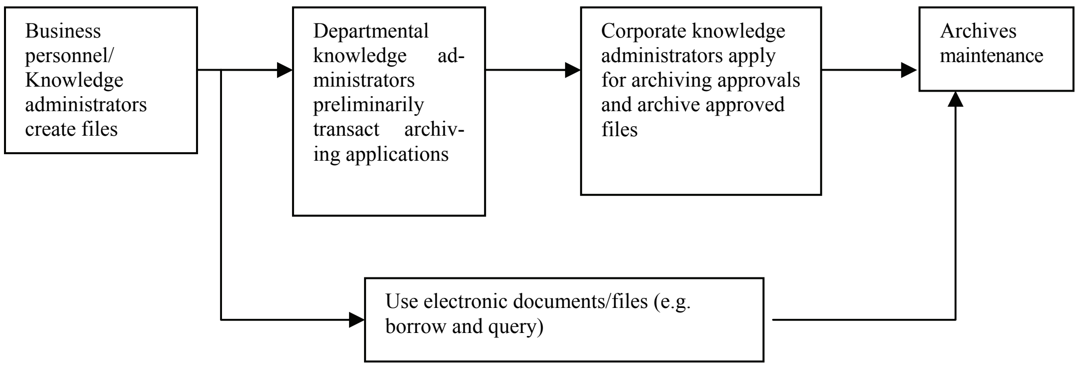

Business personnel/ Knowledge administrators create files
Departmental knowledge administrators preliminarily transact archiving applications
Corporate knowledge administrators apply for archiving approvals and archive approved files
Archives maintenance
Use electronic documents/files (e.g. borrow and query)

Figure 3. Overall flow of knowledge management system

3) Full-text retrieval: It mainly provides two functions of fuzzy retrieval and advanced retrieval. For both search modes, we have to consider the searchers' access to files. Files, to which searchers have no access, should not be included in the search results. Advanced retrieval function is to rapidly integrate related information scattered in various heterogeneous systems and then present it to users, according to the meanings of domain knowledge possessed by the information resources. The implementation of this search function is based on Conceptual Set C and Attribute Set A in the ontology =(N,F,C,A,R,R), accurate relationship between them and comprehensive setup of the ontology [7].

4) System management: It is mainly composed of five sub-modules such as classified catalog management, file storage settings, access management, log management, and full-text retrieval management.

3. System Implementation

As for three-tier architecture based system, there are three main assemblies-KMS. Web. dll for the presentation tier, KMS. BLL. dll for the business logic tier, and KMS. DAL. dll for the data access tier. And else there are three assisted assemblies-entity class KMS. Model. dll, public class KMS. Utility. dll, and class factory KMS. DAFactory. dll [8].

In order to implement real separation of data tier and other upper tiers, class factory design pattern and reflex mechanism design are integrated to fulfill the data access tier [8]. Using factory pattern + reflex mechanism + caching mechanism, dynamic creating different object interfaces to data tier is implemented.

1) Define an interface and a basic method used to access users' information [9].

```
namespace KMS.IDAL
{
        public interface IKMAdmin
            {
             /// add new data
            void Add(Maticsoft.Model.KMAdmin model);
            }
        }
```

IDAL interface project is only a statement of a series of methods, while specific implementation details will be accomplished by data access project.

2) Compile corresponding data access project for Sql database

```
namespace KMS.SQLServerDAL
{
    public class KMAdmin : IKMAdmin
    {
      public KMAdmin()
      { }
      /// add new data
      public void Add(KMS.Model.KMAdmin model)
      {
          // ellipsis
      }
    }
}
```

Data access project SQLServerDAL fulfills "add", "modify" and "delete" functions for records in connection with the Sql database.

3) Create a class factory "DALFactory" to identify which database users choose when they are building applications.

DALFactory read the database identification information from the system configuration file "web.config" to provide users with a unified interface, making users worry less about it.

When the Sql database is used, information identifiers in configuration files are as follows:

```
<add key = "DAL" value = "KMS.SQLServerDAL">
```

DALFactory project will select the corresponding database according to the database identification information.

```
private static readonly string AssemblyPath = ConfigurationManager.AppSettings["DAL"];
        /// <summary>
        /// create an object or obtain from cache memory
        /// </summary>
        public static object CreateObject(string AssemblyPath, string ClassNamespace)
        {
            object objType = DataCache.GetCache(ClassNamespace);// read from cache memory
            if (objType == null)
            {
                try
                {
                    objType = Assembly.Load(AssemblyPath).CreateInstance(ClassNamespace);// generate reflection
                    DataCache.SetCache(ClassNamespace, objType);// write cache memory
                }
                catch
                { }
             }
             return objType;
        }
        /// <summary>
        /// create data access interface of KMAdmin
        /// </summary>
        public static KMS.IDAL.IKMAdmin CreateKMAdmin()
        {
            string ClassNamespace = AssemblyPath + ". KMAdmin";
            object objType = CreateObject(AssemblyPath, ClassNamespace);
            return (Maticsoft.IDAL.IKMAdmin)objType;
}
```

The system encapsulates the processing of the lower tiers into the project and provides interfaces to the logic tier, so that it can be directly called.

4) Examples for calling interface of the logic tier.

```
namespace KMS.BLL
  {
   public class KMAdmin
     {
      private readonly IKMAdmin dal = DataAccess.CreateKMAdmin();
      public KMAdmin()
         { }
         /// add new data
         public void Add(KMS.Model.KMAdmin model)
         {
         dal.Add(model);
         }
     }
   }
```

In this way, the encapsulation and universality are fulfilled well.

4. Conclusions

Three-tier knowledge management system using .NET architecture design has good reusability, encapsulation and scalability. Compiled programs are of high security and efficiency. Separation of interface and programs makes the structure clear, which is easy to maintain. Through the design for every module in the knowledge management system, giving support to current archives management work is fulfilled. It combines archives management with knowledge management to promote internal knowledge sharing.

REFERENCES

[1] B. Gallupe, "Knowledge management systems: Surveying the land-scape [J]," International Journal of Management Reviews, 3(1), pp. 61−77, 2002.

[2] J. X. Zhou and R. R. Zhou, "Research on PLM system framework and key technologies," Transactions of Nanjing University of Aeronautics & Astronautics, Vol. 35, No. 5, 2003.

[3] R. Mack, Y. Ravin, and R. J. Byrd, "Knowledge portals and the emerging digital knowledge work place," IBM Systems Journal, Vol. 40, No. 4, 2001.

[4] C. E. Wang, Y. P. He, and Q. L. Shu, "Product life cycle modeling and management," Science Press, Beijing, 2004.

[5] S. S. Wei and N. S. Xu, "Design and implementation for data monitoring client based on .NET components," Computer & Digital Engineering, No. 6, 2008.

[6] Q. Z. Wu, "Operations research and optimization method [M]," China Machine Press, Beijing, No. 8, pp. 218, 2003.

[7] X. L. Liu, H. R. Wang, and G. J. Yang, "Research and development of ontology-based knowledge management system," Journal of Hebei University, No. 5, 2008.

[8] S. Li and J. X. Wang, "Design and implementation of enterprise manager assessment & evaluation system based on .NET three-tier B/S architecture," Journal of Nanjing University, No. 3, 2007.

[9] K. Zeng, "Design and implementation of B2C mobile e-commerce website based on ASP. NET," Computer & Digital Engineering, No. 3, 2008.

J. Software Engineering & Applications, **2009, 2: 44-49**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

QoS-Aware Reference Station Placement for Regional Network RTK

Maolin Tang[1]

[1]Cooperative Research Centre for Spatial Information, Faculty of Science and Technology, Queensland University of Technology
2 George Street, Brisbane, Australia
Email: m.tang@qut.edu.au

Received December 24th, 2008; revised January 30th, 2009; accepted February 26th, 2009.

ABSTRACT

Network RTK (Real-Time Kinematic) is a technology that is based on GPS (Global Positioning System) or more generally on GNSS (Global Navigation Satellite System) measurements to achieve centimeter-level accuracy positioning in real-time. Reference station placement is an important problem in the design and deployment of network RTK systems as it directly affects the quality of the positioning service and the cost of the network RTK systems. This paper identifies a new reference station placement for network RTK, namely QoS-aware regional network RTK reference station placement problem, and proposes an algorithm for the new reference station placement problem. The algorithm can always produce a reference station placement solution that completely covers the region of network RTK.

Keywords*: *Reference Station, Placement, Regional Network RTK, QoS

1. Introduction

Network RTK (Real-Time Kinematic) is a technology that is based on GPS (Global Positioning System) or more generally on GNSS (Global Navigation Satellite System) measurements to achieve centimeter-level accuracy positioning in real-time. It involves a network of reference stations that are used to take and transmit their raw GPS or GNSS measurements [1,2,3,4].

The success of network RTK in recent years has resulted in the establishment of network RTK positioning services to anyone who is willing to pay for them. In order to provide network RTK services to an area, such as a state or a country, many reference stations must be set up and maintained. However, setting-up and maintaining reference stations is costly, typically tens of thousands of Australian dollars each with annual operational cost of approximately 10% of the reference station cost. Therefore, it is desirable to minimize the total number of reference stations without compromising the QoS of network RTK. However, the reference station placement problem has not drawn enough attention from the community, and little research has been done.

In our preliminary research, we have identified a significant reference station placement problem, namely location-oriented reference station placement problem, and proposed an effective and efficient algorithm for the reference station placement problem [5]. That reference station placement problem is described as: Given a set of potential locations where reference stations can be set up and the locations of the users, select reference station locations among the potential reference station locations such that the total number of reference stations is minimum subject to a constraint, that is, the maximal distance between any user to the closest reference station does not exceed a parameter D_{max}. This maximum distance constraint must be satisfied in order to guarantee the accuracy of the positioning services.

Recently, we identified and proposed an algorithm for an area-oriented reference station placement problem for network RTK [6]. Given a service area that a network RTK needs to cover, the area-oriented reference station placement problem strives to find a reference placement such that the service area is completely covered by the reference stations. The major deficiency of the algorithm is that it cannot provide a reference placement solution that can guarantee to provide QoS because the algorithm cannot guarantee to generate a reference station placement solution that completely covers the service area although in most cases it does. As a result, those users who are located in an area where is not covered by the reference stations may not receive quality positioning service.

In this paper we propose a new reference station placement problem, namely, QoS-aware regional network RTK reference station problem, and extend the area-oriented reference station placement problem to guarantee to generate a reference station placement solution that covers the whole service area. The proposed algorithm has been implemented and tested by simulation. Experimental results show that the algorithm is efficient and effective.

The remaining paper is organized as follows. Section 2 formulates the QoS-aware regional network RTK reference station placement problem. Section 3 discusses related work. Section 4 presents our approach to the new reference station placement problem in detail. The simulation results are presented in Section 5. Finally we conclude the proposed approach in Section 6.

2. Problem Formulation

Given an area and the maximal distance constraint from any user to its closest reference station, the QoS-aware regional network RTK reference station placement problem is to find locations on the area such that all the users at any location within the area can receive real-time centimeter accuracy positioning services.

In order to guarantee a user can get the real-time centimeter accuracy positioning services, it must be guaranteed that the maximum distance between the user to its closest reference station is less than or equals to the maximum distance constraint. Because the user can be distributed at any position on the area, the maximum distance constraint implies that the maximum distance between any position in the area and its closest reference station must not exceed the maximum distance constraint. Thus, the QoS-aware network RTK reference station placement is formulated as:

Given an area *A* and the maximal distance constraint D_{max}, find a set of reference stations $R = \{r_1, r_2, \ldots, r_n\}$ such that $|R|$ is minimum, subject to $\forall p = (x_p, y_p) \in A$, $\exists r_i = (x_i, y_i) \in R$, where, and $\sqrt{(x_p - x_r)^2 + (y_p - y_r)^2} \le D_{max}$. The constraint guarantees that the QoS, or the accuracy of positioning services, is satisfied.

3. Related Work

In our preliminary research on network RTK, we have identified a so-called location-oriented reference station placement problem. The location-oriented reference station placement problem is stated as below:

Given a set of reference station candidates $R = \{r_1, r_2, \ldots, r_n\}$, a set of users $U = \{u_1, u_2, \ldots, u_m\}$, and the maximal distance between any user and its closest reference station $D_{\max}$, the location-oriented reference stations placement problem is to find $R_0 \subseteq R$, such that $|R_0|$ is minimal, subject to $\forall u_i = (x_i, y_i) \in U$, $\exists r_j = (x_j, y_j) \in R$ and $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \le D_{\max}$, where $1 \le i \le m$ and $1 \le j \le n$.

The location-oriented reference stations placement problem can be represented in a so-called graph $G = (V, E)$, where $V = U \cup R$. An edge $(v_1, v_2) \in E$, if and only if $v_1 = (x_1, y_1) \in U$, $v_2 = (x_2, y_2) \in R$, and $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \le D_{max}$, where (x_1, y_1) and (x_2, y_2) represent the location of v_1 and the location of v_2, respectively. Therefore, the location-oriented reference station placement problem can be transformed into the following graph-theoretic problem:

Given a RTK graph $G = (V, E)$, where $V = U \cup R$, find $R_0 \subseteq R$, such as that $V = V_0$, where $V_0 = \{v | (r, v) \in E \ \& \ r \in R_0\}$.

A RTK graph can be represented by an $m \times n$ adjacency matrix $A = [a_{ij}]_{m \times n}$, where

$$a_{ij} = \begin{cases} 1, & if\,(u_i.r_j) \in E \\ 0, & otherwise \end{cases}$$

For example, for the RTK graph shown in Figure 1. The adjacent matrix

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The location-oriented reference stations placement problem is a constrained version of the problem of finding a minimum dominating set of an arbitrary graph, which is an NP-hard [7].

It is conjectured that this reference stations placement problem is also NP-hard. Thus, we proposed a heuristic algorithm in [5].

The heuristic algorithm starts with an initial reference station placement that uses all the reference station candidates. Then, the number of reference station candidates is gradually reduced by iteratively removing redundant reference stations. A reference station is redundant if all the users that can be covered by the reference station can also be covered by other reference stations. A user is said to be covered by a reference station if the distance between the user and the reference station does not exceed $D_{\max}$. A redundant reference station r_i can be identified in the matrix representation by checking if there is a second non-zero entry in the j^{th} column of the matrix. Algorithm III is the high-level description of the location-oriented reference station placement algorithm.

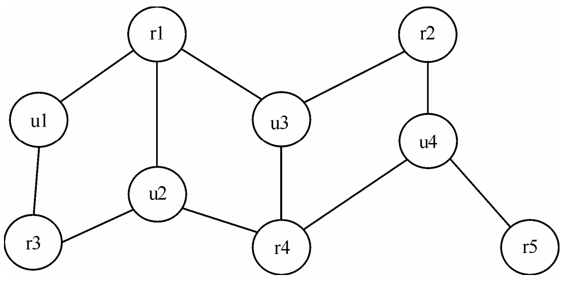


Figure 1. A RTK graph

```
Algorithm 1: Location-oriented placement algorithm
Require: Locations of reference station candidates, locations of users and the maximal distance allowed between a user and its reference station $D_{\max}$
Ensure: Return the location of reference stations
    generate its RTK graph $G = (V, E)$;
    for each reference station candidate $r \in R$ do
        if r is redundant then
            remove it from R;
            remove its associated edges from E;
        end if
    end for
    Output the location of the reference station candidates left in R.
```

It has been proved in [5] that the computational complexity of the algorithm is $O(n^3 * m)$, where *m* is the number of reference station candidates and *n* represents the number of users. Details about the algorithm analysis can be found in [5].

4. Approach

The QoS-aware network RTK reference station placement problem is a continuous optimization problem and it is difficult to be tackled directly. Thus, we transform the QoS-aware network RTK reference station placement problem into the location-oriented reference station placement problem discussed in the Related Work, and use the location-oriented reference station placement algorithm to tackle the QoS-aware network RTK reference station placement problem.

In the following, we will discuss how to transform the QoS-aware network RTK reference station placement problem into the location-oriented reference station placement problem. Then, we will present an algorithm for the QoS-aware network RTK reference station placement problem, and then we analyses the optimality and efficiency of the new algorithm.

4.1 Transformation

The QoS-aware network RTK reference station placement problem is a continuous optimization problem, where the locations of reference stations are on a continuous two-dimensional region. Thus, it is difficult to be handled directly as its search space is infinite. However, the continuous optimization problem can be transformed into the location-oriented reference station placement problem using the following procedure.

First of all, a two-dimensional grid graph is generated to completely cover the whole region of the network RTK *A*. This idea is illustrated in Figure 2. In the figure, the filled enclosed area represents the network RTK service area A.

Then, we select those grid nodes on or outside the boundary such that the area of the polygon formed by connecting those grid nodes is minimal, but can completely cover the network RTK service area *A*. Figure 3 shows such a polygon based on the grid graph shown in Figure 2.

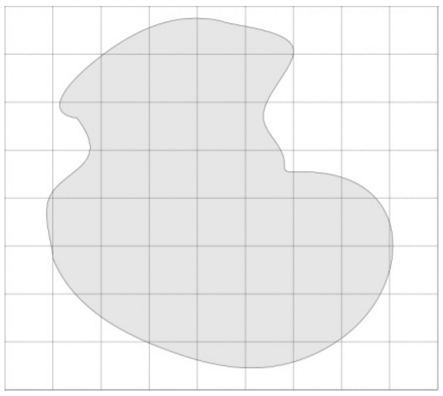


Figure 2. A grid graph that completelly covers the network RTK service area *A*

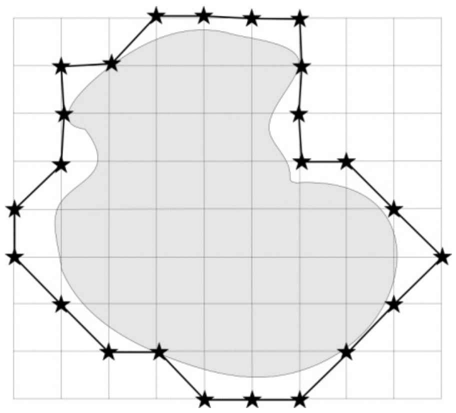


Figure 3. A minimal polygon that completelly covers the network RTK service area *A*

And then, we create a sub-graph of the grid graph by selecting all the grid nodes and arcs enclosed by the polygon. Figure 4 shows the sub-graph generated from the grid graph and the polygon shown in Figure 3.

In order to make use of the location-oriented reference station placement algorithm discussed in the Related Work section, we need to create dummy users and initial reference stations (reference station candidates). The selection of users and initial reference stations is very important because if they are not well chosen, then the location-oriented reference station placement algorithm may not produce a solution that can completely cover the whole service area of the regional network RTK. Thus, in the following, we will introduce rules for selecting dummy users and initial reference stations. We will prove that by selecting dummy users and initial reference stations it is guaranteed that the algorithm can always produce a solution that can completely cover the given service area of the regional network RTK.

Given a grid graph $G = (V, E)$, all the nodes whose degree is greater than one are selected as reference stations and on each edge we create three dummy users: one at each end of the edge and one at the middle of the edge. The rules for selecting initial reference stations and the rules for selecting dummy users are illustrated in Figure 5 and Figure 6, respectively. It will be proven in the paper that by selecting reference station candidates and user positions using the above rules, our algorithm can always produce a reference placement solution that can 100% covers the given service area *A*.

Using the rules, we can create a set of initial reference stations $R = \{r_1, r_2, \ldots, r_n\}$ and a set of users $U = \{u_1, u_2, \ldots, u_m\}$, and have transformed the QoS-aware regional network RTK reference station problem into the location-oriented reference station placement problem.

4.2 Algorithm Description

Once we have transformed the QoS-aware regional network RTK reference station placement problem into the location-oriented reference station placement problem, we can make use of the ideas of the location-oriented reference station placement algorithm presented in the Related Work section to tackle the QoS-aware regional network RTK reference station placement problem. Algorithm 2 gives a high-level description of the algorithm for the QoS-aware regional network RTK reference station placement problem.

Algorithm 2: QoS-aware regional network RTK placement station placement algorithm

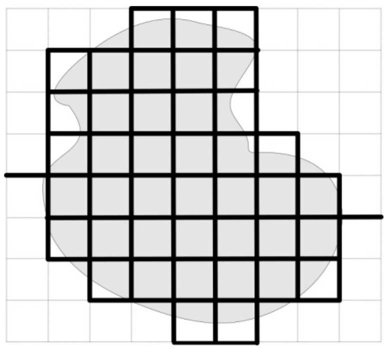


Figure 4. The subgraph enclosed by the minimal polygon

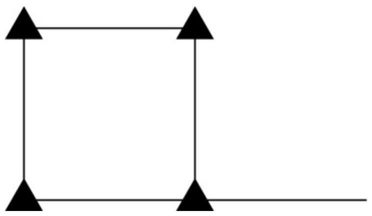


Figure 5. Initial reference stations selection rule

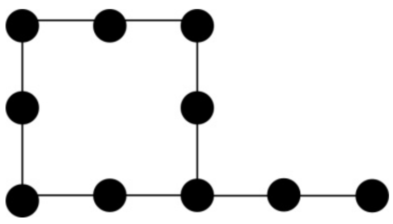


Figure 6. Dummy users selection rule

```
Require: Network RTK service area A and the maximal distance allowed between a user and its reference station $D_{max}$.
Ensure: Return the location of reference stations
    generate a grid graph of grid size g;
    generate the minimal polygon covering the network RTK service area A;
    generate the sub-graph enclosed by the minimal polygon;
    use the rules to create reference station candidate set R and user set U;
    generate its RTK graph $G = (U \cup R, E)$;
    for each reference station candidate $r \in R$ do
        if $r$ is redundant then
            remove it from R;
            remove its associated edges from E;
        end if
    end for
    Output the location of the reference station candidates left in R.
```

4.3 Algorithm Analysis

This subsection provides theoretic analysis of the algorithm for the QoS-aware regional network RTK reference station placement problem presented in the previous subsection.

***Lemma* 1:** Algorithm 2 always produces a reference placement station placement solution in which every generated user is covered by at least one of the reference stations.

***Proof*:** Algorithm 2 starts with generating a set of users and a set of reference stations, and then iteratively reduces redundant reference stations. It can be guaranteed by the rules for identifying users and initial reference stations that every generated user can be covered by at least reference station initially, and that during the iterative process of reducing redundant reference stations only redundant reference stations are removed, which means all the users that is covered by a removed redundant reference station can also be covered by another reference station. Thus, after removing all the redundant reference stations, every generated user is covered by at least one of the reference stations.

***Theorem* 1:** Given an area *A* and the maximum distance constraint $D_{\max}$. algorithm 2 always produces a reference station placement solution that completely covers *A*.

***Proof*:** The given area A is enclosed by a polygon. If we can prove that the polygon is completely covered by the reference station placement solution produced by Algorithm 2, then we have proved that the given area *A* is completely covered by the solution. In addition, it has been proved in Lemma 1 that Algorithm 2 can always produce a solution that covers all the dummy users. Thus, all we need to prove in the following is that when all the dummy users are covered, the polygon is 100% covered.

The area of the enclosing polygon is composed of two basic types of two-dimensional areas, squares and isosceles right triangles. For each of the squares as shown in Figure 7, it will be completely covered by the solution because in order to cover all the eight user position on the square as shown in Figure 6 at least two diagonal reference station candidates must be selected. When any two of the diagonal candidates on a square are selected, the whole area of the square will be covered. Note that the coverage of the reference station is always the length of the square. Thus, it can be concluded that all the square areas will be covered by the solution. Figure 7 illustrates the idea. In the figure, two diagonal candidates A and C are selected. The shadow circles display the coverage of the reference stations at candidate A and candidate C.

For each of the isosceles right triangles as shown in Figure 8, it will also be completely covered by the solution because according to the reference station candidate and user selection rules, the only possibility to cover all the three users on the line from A to B is to select the reference station candidate at position A, and when that reference candidate is selected, the isosceles right triangle ABC will be completely covered by the reference station. This is illustrated in Figure 8.

The quality of solutions obtained by the algorithm and the computation time of the algorithm are both dependent on the grid size g, which is an important parameter used by the algorithm. In the following section, we present our empirical study results on the impact of g on the optimality and performance of the algorithm.

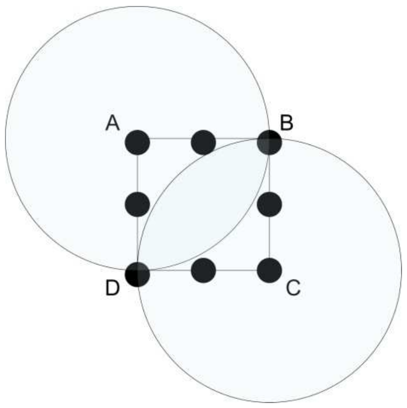


Figure 7. The square case

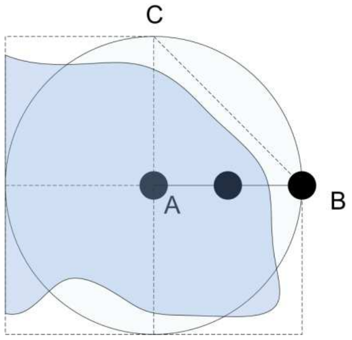


Figure 8. The isosceles right triangle case

5. Experiment

The proposed algorithm has been implemented in Microsoft Visual C#. In order to test the optimality and performance of the algorithm, we used the implemented algorithm to find a reference station placement solution for the network RTK area shown in Figure 2. The value of D_{max} . was fixed to 140km, but the values used for the gird size g varied from 10 to 140 with an increment of 10. All experiments were conducted on desktop computers with an Intel Core 2 Duo 6300 CPU (1.86GHz) and 2 GB of RAM. Table I records the experimental results.

Figure 9 shows the trend of the quality of the solutions obtained by the algorithm when the grid size increases. It can be seen from the figure that the grid size used by the algorithm directly affects the quality of the solutions obtained by the algorithm, and that the smaller the grid size the better solution the algorithm produces. For example, when the grid size is 20 the solution produced by the algorithm needs only 23 reference stations. However, when the grid size increases to 140 the solution generated by the algorithm needs 34 reference stations.

Figure 10 shows the trend of the computation time of the algorithm when the grid size increases. It can be seen from the figure that the computation time of the algorithm decreases dramatically when the grid size increase. For example, when the grid size is 20 the computation time is 47.81 seconds. When the grid size increases to 140 the computation time is only 0.02 seconds.

The experimental results reveal that the smaller the grid size is, the better the quality of the solution is. However, the computation time increases very quickly when the grid size decreases. Figure 11 displays the result for the area-oriented station placement problem obtained by the implemented algorithm when the grid size *g*=20. In the figure, the small filled squares represent reference stations and the circles are the coverage of the corresponding reference stations.

Table 1. Experimental results on the impact of g on the optimality and computation time of the algorithm

Grid Size (*g*)	Reference Station Number	Time (second)
20	23	47.81
40	26	2.32
60	27	0.45
80	28	0.15
100	31	0.06
120	33	0.03
140	34	0.02

6. Conclusions

This paper has identified a new network RTK reference station placement problem, namely QoS-aware regional network RTK reference station placement problem, and has proposed an effective algorithm for the problem.

The basic idea behind the algorithm is to transform the QoS-aware regional network RTK reference station placement problem, which is basically a continuous area-oriented reference station placement problem, into an existing discrete location-oriented reference station placement problem and make use of the existing algorithm for the location-oriented reference station placement problem to solve the challenging area-oriented reference station placement problem. The transformation process and the algorithm for the QoS-aware regional network RTK reference station placement problem have been discussed in detail in this paper.

The most challenging job in tackling the continuous area-oriented reference station placement problem is to guarantee the reference station placement solution can completely cover the region of the network RTK. It has been proved in this paper that our proposed algorithm can always produce a reference station placement solution that can 100% cover the service area of a regional network RTK.

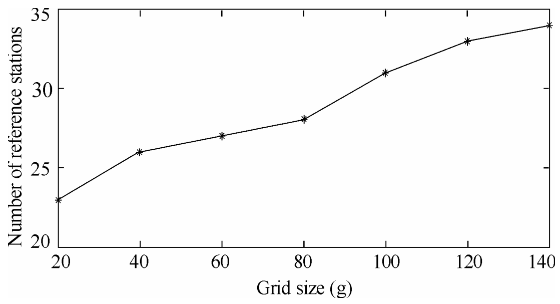


Figure 9. The impact of grid size on the optimality of the algorithm

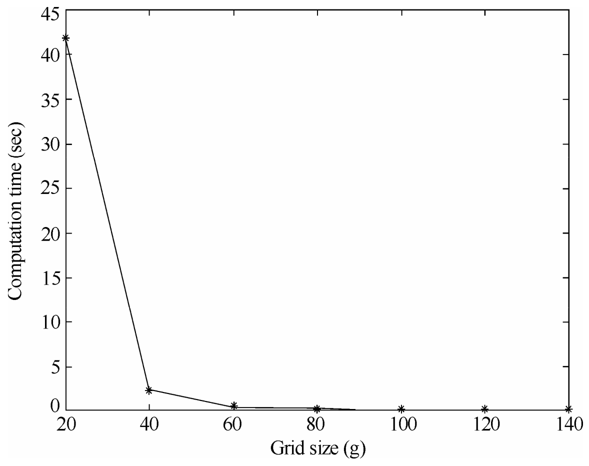


Figure 10. The impact of grid size on the computation time of the algorithm

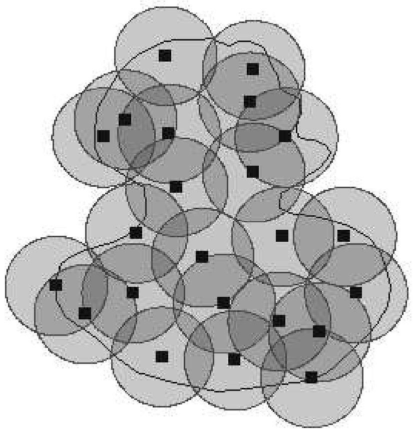


Figure 11. An area-oriented reference station placement solution

The grid size g is an important control parameter in the transformation as it affects the optimality and performance of the algorithm. Generally, the smaller g's is, the better the solution is. However, when g's value becomes small, the computation time increases dramatically. A potential solution to this problem is parallel implementations of the algorithm.

7. Acknowledgement

The work was carried out with financial supports from Cooperative Research Centre for Spatial Information (CRCSI) project 1.4: “Integrating electricity, telecommunications and government infrastructure to deliver precise positioning services in regional areas”, 2007−2010.

The proposed algorithm was implemented by Lifeng Ai of Queensland University of Technology.

REFERENCES

[1] L. Wanninger, “Introduction to Network RTK,” http://www. networkrtk. info/intro/introduction.html.

[2] C. Rizos, “Network RTK research and implementation-A geodetic perspective,” Journal of Global Positioning Systems, Vol. 1, No. 2, pp. 144−150, 2003.

[3] L. Wanninger, “GPS on the web: Virtual reference stations VRS,” GPS Solutions, Vol. 7, pp. 143−144, 2003.

[4] G. Fotopoulos and M. E. Cannon, “An overview of multi-reference station methods for Cm-Level positioning,” GPS Solutions, Vol. 4, No. 3, pp. 1−10, 2001.

[5] M. Tang, Y. Feng, L. Burnett, R. Hayward, and K. Raymond, “A reference station placement scheme in deployment of network real-time kinematic positioning systems,” Proceedings International Global Navigation Satellite Systems Society Symposium, Sydney, pp. 123−132, December 2007.

[6] M. Tang and Y. Feng, “Area-oriented reference station placement for network RTK,” Proceedings International Conference on Computer Science and Software Engineering, Wuhan, December 2008.

[7] M. Garey and D. Johnson, “Computers and intractability: A guide to the theory of NP-Completeness,” W. H. Freeman, San Francisco, 1979.

J. Software Engineering & Applications, **2009, 2:50-54**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

Research on Survivability of Mobile Ad-hoc Network

Yuan Zhou[1], Chunhe Xia[1], Haiquan Wang[1], Jianzhong Qi[1]

[1]Key Laboratory of Beijing Network Technology, School of Computer Science and Engineering, Beihang University, Beijing, China.
Email: zhouyuan84825@163.com

Received November 26th, 2008; revised January 7th, 2009; accepted February 16th, 2009.

ABSTRACT

In this paper, we analyze the survivability of Mobile Ad Hoc Network systemically and give a detailed description of the survivability issues related to the MANET. We begin our work with analyzing the requirements of survivability of ad hoc network, and then we classify the impacts that affect survivability into three categories: dynamic topology, faults and attacks. The impacts of these factors are analyzed individually. A simulation environment for the MANET towards survivability is designed and implemented as well. Experiments that under the requirements and the impacts we declared are done based on this environment.

Keywords: *Ad hoc*, *Survivability*, *GTNeTS, Simulation*

1. Introduction

A mobile ad-hoc network (MANET) is a kind of self-organized wireless network with a collection of mobile hosts that is multi-hop instead of using any fixed infrastructure or centralized management [1,2,3]. There are a lot of potential applications of MANET in both civilian and military areas. For instance, it can be applied in disaster communications, used as the back-up network of traditional mobile communication networks and used to build tactical network, etc. With the characteristic of MANET like: dynamic topology, no infrastructure or trust institution and limited power resources, research on survivability of MANET is becoming an academic hotspot.

The concept of survivability was first presented by Barnes in 1993. After that, Knight, K.Sullivan, R. J. Ellison and many research institutes have done lots of effort on issues of survivability. To be brief, survivability is the capability of a system to provide essential services under attacks, failures or accidents. At present, many researches of MANET survivability put emphasis on designing new survivable routing protocols for specific problems [4,5]. However, there is few integrated and systematic analysis in the MANET survivability area [6]. On the other hand, research institutions usually use simulation approaches for experimental verification on MANET. However, the existing MANET simulators are mainly constructed for the purpose of experimenting performance or protocol of MANET [7]. Attributions related to MANET survivability are absent in the simulator structure. Additionally, threats like faults and attacks can not to be brought in to those simulators easily to verify MANET survivability issues.

For the above reasons, this paper presents a systematic analysis of MANET towards survivability. It describes how various impacts affect the survivability of MANET in details. Meanwhile, a simulation environment is designed and implemented for MANET towards survivability, which incorporates not only the attributions related to MANET survivability but also fault events and attack events against MANET. Users can draw many kinds of scenarios of MANET to test survivability issues under this simulation environment.

This work was supported by three projects: the National 863 Project -Research on high level description of network survivability model and its validation simulation platform under Grant No.2007AA01Z407, The Co-Funding Project of Beijing Municipal education Commission under Grant No.JD100060630 and National Foundation Research Project.

2. Analysis of Survivability of MANET

2.1 Definition of Survivability

Barnes presented the conception of "Survivability" for the first time in 1993, but until now, there is no all-acceptant definition of Survivability. The most influencing definition is presented by a research group of CMU/SEI, who define survivability as the capability of a system to fulfill its mission in a timely manner, in the presence of attacks, failures, or accidents [8]. Many other researchers define survivability from different perspective: In the area of software engineering, Deutsch defined surviva- bility as the degree to which essential functions are still available even though some part of the system is down [9]. Ellison *et al.* introduced the following definition: survivability is the ability of a network computing system to provide essential services in the presence of attacks and failures, and recover full services in a timely manner [10]. Researchers like Moitra [11], Jha [12], and Wilson [13] also defined the survivability similarly.

The definitions above well describe the meaning of survivability in natural language, but attributions towards survivability are not embodied, especially those of the MANET. Compared with the descriptive definition, a formal description can give a more rigorous definition of survivability of MANET.

2.2 Definition of Survivability of MANET

After analyzing the definitions listed above, we find that the essence of survivability is the ability of providing essential system-wide service. Papers [6,14,15] presented that the essential service a mobile ad hoc network must provide is the communication service. That is to say, in the context of MANET, the essential service is primarily pivotal to a fundamental requirement: establishing a connection between any two nodes in an ad hoc network at any instant. So the survivability issue depends on how well an ad hoc network demands the requirement. On the other hand, the threads mobile ad hoc network must face to are as follows:

1) Dynamic topology influence. Nodes in an MANET can move arbitrarily. Consequently, network topology may change rapidly and randomly.

2) Faults that may happen in nodes and links. For instance, a node can shutdown itself for certain reasons, and a link may be influenced by an obstacle.

3) Every layer of ad hoc network architecture may be under attack. The Wormhole, Blackhole attacks aim at the network layer, jamming and eavesdropping attacks aim at the physical layer, and SYN flooding attacks aims at the transport layer, to name a few.

With the above analysis we can define the survivability of a mobile ad hoc network as, the ability of establishing a communication service between any two nodes in the network at any instant under the impact of dynamic topology, faults and attacks.

2.3 Detailed Description of Survivability of MANET

Based on the definition of survivability of MANET above, the system of MANET towards survivability issues can be abstracted into three members: mobile ad hoc network, the impacts which can affect the survivability of the mobile ad hoc network, and the essential service that a mobile ad hoc network must provide. It can be described as follows. Then, we will analyze them in detail

$$SurvivalAdhoc=\{ADHOC,SERVICE,IMPACT\} \tag{1}$$

The mobile ad hoc network is composed of a number of nodes with certain attributions related to the survivability and directed links. We consider the node obtaining following attributions: IP address, location, radio range, state (transmit, receive, idle and down), power, protocol type and, mobility type. All these attributions are related to the survivability of MANET. It can also be described as follows:

$$ADHOC::=\{NODE,LINK\} \tag{2}$$

$$NODE::=\{node_1,node_2,...node_n\} \tag{3}$$

$$LINK::=\{<node_i,node_j>|node_i,node_j \in NODE\} \tag{4}$$

$$node=(ip,id,location,R,state,power, protocoltype,mobilitytype) \tag{5}$$

The essential service of a mobile ad hoc network is to establish a communication service between any two nodes in the network at any instant. The ad hoc network establishes a connection by two steps. First, connections between any two adjacent nodes are provided by the link layer and physical layer, and then such connections are extended by the network layer form one-hop to multi-hops. That is to say: for an ad hoc network with N nodes, there need to be a directional path ($Node_i$, $Node_j$… $Node_l$, Nod_m,) between any two nodes. And the path must meet the following requirements:

1) The distance between any two nodes in neighbors, $Node_i$ and $Node_{i+1}$, is less than the transmission range of $Node_i$.

2) Any node in this path must possess a routing entry that sets the target node as the destination node, and the next hop is the succeeding node.

The impacts affecting the ad hoc network survivability are dynamic topology, faults and attacks. The essence of all these three factors is to destroy the path between two nodes so that the network is disconnected. Thus, the factors are described as follows:

$$\begin{aligned} IMPACT ::= \{a \mid DynamicToplogy(a) \vee Fault(a) \\ \vee Attack(a), a \in Actions\} \end{aligned} \tag{6}$$

The factor of dynamic topology is caused by the actions such as mobility of nodes. It can result in the location changed of the node, and lead to the distance requirement of link layer connection dissatisfied. The influence of dynamic topology is that the new distance of two nods in neighbor may be larger than the transmission range of the first node, so that the path is destroyed. If there is no other path between the destination node and target node, they can't afford the communication service.

Faults are considered to include node faults and link faults [14]. Node faults are the actions that cause the state of the nodes changed to down. If the state of a node is changed to down, the transmission range will be changed to 0, which will make the path dissatisfy the requirement. Link faults are introduced by obstacles between nodes, or by signals fading effect which may influence the transmission range of nodes. It can be described as follows:

$$Fault(a) \to Node(a) \vee Link(a), a \in Actions \tag{7}$$

Attacks of MANET are usually classified by network layers. The attack aiming at the application layer is more or less the same as the wired network. Its main object is to disrupt the application service, and worm is an instance. Attacks to the transportation layer are the actions to disrupt the transportation layer protocols, like SYN flooding. The attacks to the network layer are the actions that destroy the routing and forwarding processes in ad hoc network [16]. The attacks to link layer are the actions which destroy the connection of adjacent nodes. The attacks to physical layer are actions to jam, intercept or to eavesdrop the wireless channel. The description of attacks to network survivability is:

$$Attack(a) \rightarrow PhysicalAttack(a) \vee LinkAttack(a)$$
$$\vee NetworkAttack(a) \vee TranspAttack(a) \vee ApplicationAttack(a) \quad (8)$$

3. Simulation Environment

Existing simulators are not well-equipped to serve our purpose. Hence, we design a survivability-based simulation environment of MANET to test how those factors influence the network.

3.1 Characteristics of Simulation Environment

While designing the simulation environment, we concern the following factors:

• Towards Survivability: The users can configure the parameters freely, such as node amount, mobility model, radio range, and power of node, which may affect MANET survivability. Fault events and attack events can be added to simulate scenarios towards survivability thoroughly.

• Expandability: Users can easily expand the simulation environment by adding new protocols of various layers, modules, attack events and fault events. For this reason, MANET towards survivability description language is designed and a language translator is implemented for it. Users can add new protocols, functions, faults and attacks through appending new keywords.

• Introduce Fault-event and Attack-event: Based on the definition and description of MANET survivability, faults and attacks are described, modeled and simulated to support MANET survivability verification.

• Data Acquisition Interface: One of the most important purposes of simulation is collecting data for further analysis from simulation process. Thus, data acquisition interface is incorporated in the simulation environment and users are allowed to configure what kind of result to be collected.

Based on the analysis above, simulation environment includes description language of MANET towards survivability and its interpreter. Then, we choose Georgia Tech Network Simulator (GTNetS) [17] as the underlying network simulation platform.

3.2 Simulation Environment Architecture

The system architecture is as shown in Figure 1:

•Graphic Configuration Interface: MANET properties, attack-events and fault-events information are configured by users through graphic user interface (GUI) and then be saved in the configuration file.

• Event interpretation: it analyzes and interpreters the configuration file, executes corresponding events by calling functions from event class library. The event class library consists of fault class library and attack class library, each of which is built up by specific classes. Currently, there are node fault classes, link fault classes in fault class library. And the attack class library contains energy-consuming attack class, signal interference attack class, black hole attack class, SYNflooding attack class and Worn attack class. All these classes are used by receiving parameters from the interpreter and calling support functions from GTNetS.

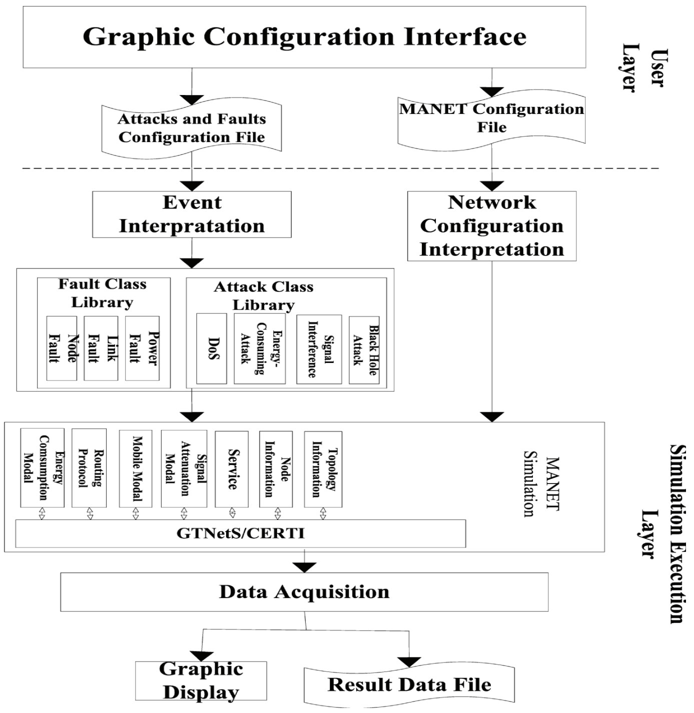


Figure 1. Simulation environment architecture

• Network configuration interpretation: it interprets the network configure file and makes function calls from GTNetS to construct simulation of MANET. All these network properties are supported: energy-consumption modal, AODV and DSR routing protocol, Random Waypoint, Random Walk and Random Direction mobility models, data transfer service, node information and topology information.

• Data Collection Module: it displays the simulation process graphically. After that, it parses the collected data that users care about, including total data transferred amount, average connectivity node-pairs and so on, then saves data into the result data file.

4. Experiments on MANET Survivability Analysis

4.1 Related Definition

For the simulation test of the definition of MANET survivability, the capability to provide connectivity service under various threats, terms that used in this paper are as follows:

Average Connectivity Efficiency [6] (*E*): It means the ratio of connected node pairs to all the node pairs within an N-node MANET in a certain period. In such MANET, each link between two nodes is directed, which means node *pair* (*i,j*) is different from node *pair* (*j,i*). The value of (*E*) reflects the capability of a MANET to provide connectivity service at a certain time. It reaches 100% when all the node pairs in the network can be connected.

$$Average\ Connectivity\ Efficiency(\%) = \frac{\sum_{i=1}^{T}(no.of\quad connected\quad node\quad pairs)*100}{T*Number\quad Of\quad Node-pairs} \tag{9}$$

4.2 Experiment of Influences of Dynamic Topology

We choose number of node, radio range of node and mobility speed of node as the variables in this experiment. We would like to figure out that how the dynamic topology influences the survivability of MANET.

We carry out the experiment in a 1000×1000 bounded squared topology, and Parameters used in the test of the influences on MANET survivability of dynamic topology are as follows:

• Simulation time:300
• Node amount:20-80
• Radio range of each node:50-250
• Mobile way of node: Random Walk, 10-20
• Routing protocol of each node :AODV
• Bandwidth of each link:1Mb

From the results we can easily find out that when the radio range increases, the connectivity efficiency increases. Howeverm, its effect to the connectivity efficiency is not obvious when the radio range invreases to a certain level.As a result, when choosing the radio range, we should consider its effort to the connectivity efficiency and its cost to the power consumption.

When the moving speed of node increases, the topology is more instable, which means the node will be easier to move out of other nodes' transmission range, so that the connectivity efficiency decreases.

A larger number of nodes mean a higher node density. So from the results we can see in a bounded area, large number of nodes will contribute to the connectivity efficiency.

4.3 Experiment of Influence of Faults and Attacks

There are node faults and link faults. When node faults happen, nodes can shut down themselves randomly. When link faults happen, it can be emerging obstacles that affect the links. This paper assumes that only one fault happens at one time, and ignores the mobility attribution of the nodes, so that the topology is stable when dealing with the influences of faults.

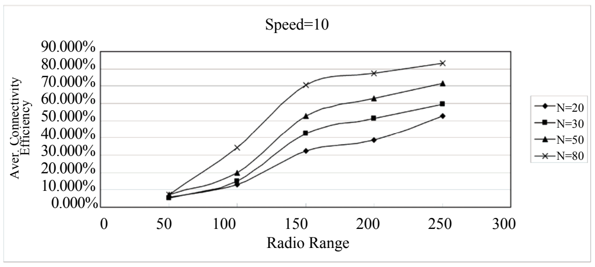


Figure 2. Average connectivity efficiency vs. transmission range for different number of node when speed =10

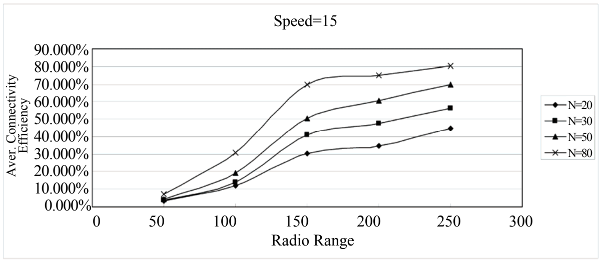


Figure 3. Average connectivity efficiency vs. transmission range for different number of node when speed =15

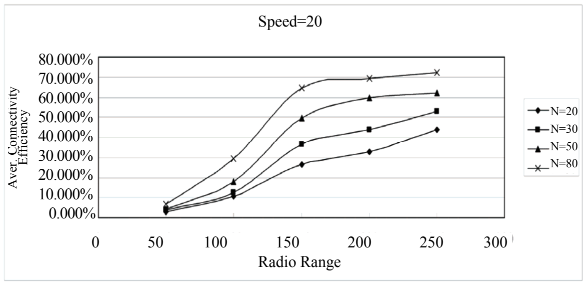


Figure 4. Average connectivity efficiency vs. transmission range for different number of node when speed =20

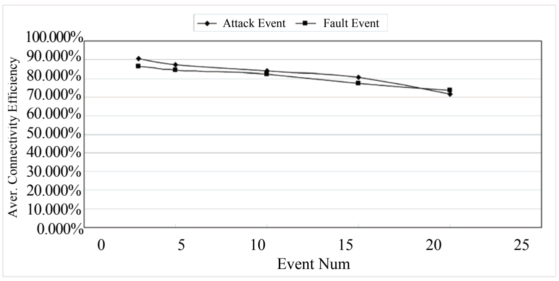


Figure 5. Average connectivity efficiency vs. different event number per minute

In this paper, attacks of network layer, data-link layer and physical layer are the three types of attacks that are included in the experiment. Energy consumption attacks, for network layer attack testing, and jamming attacks, for physical layer attack testing, are selected as examples to test the influences on MANET survivability under attacks. Node mobility is not considered either.

We make the simulation in a 1000*1000 bounded square area under different event frequency, which means we set different number of event that happen per minute. For every minute we choose these events randomly.

Parameters used in the experiment of the influences on MANET survivability of faults or attacks are as follows:

- Simulation time:300
- Average fault/attack amount per minute:3−20
- Node amount:100
- Radio range of each node:100
- Mobile way of each node: quite
- Routing protocol of each node: AODV
- Bandwidth of each link:1Mb

From the results we can find out that when event frequency of faults or attacks increases, more nodes can't provide stable services for routing and forwarding, and the connectivity efficiency decreases.

5. Conclusions and Future Work

In this paper, we systematically analyzed MANET survivability and the impacts that affect it. Based on this, we designed and implemented the simulation environment. The simulation environment represents characteristics of MANET towards survivability and events that impact survivability of MANET like fault events and attack events. We then tested the influences of the above- mentioned impacts under the simulation environment. Users can also configure new topologies, attacks and faults to test MANET survivability in this simulation environment using their own scenario. In our current research, the influences of each factor were tested independently. That means, future work shall be done to present cross-analysis of the influences of these impacts.

REFERENCES

[1] P. Papadimitratos and Z. Haas, "Handbook of ad hoc wireless networks," chapter Securing mobile ad hoc networks, CRC Press, 2002.

[2] F. Adelstein, S. K. S. Gupta, and G. G. Richard III, "Funda-mentals of mobile and pervasive computing," McGraw-Hill, 2005.

[3] D. Djenouri, L. Khelladi, and A. N. Badache, "A survey of security issues in mobile ad hoc and sensor networks," IEEE Communications surveys & tutorials, 7(4): pp. 2−28, 2005.

[4] M. G. Zapata, "Secure ad hoc on-demand distance vector routing," ACM SIGMOBILE Mobile Computing and Communications Review, 6(3): pp. 106−107, 2002.

[5] Y. Xue and K. Nahrstedt, "Providing fault-tolerant ad hoc routing service in adversarial environments," Wireless personal communications: an international journal, 29 (3−4): pp. 367−388, 2004.

[6] K. Paul, R. R. Choudhuri, and S. Bandyopadhyay, "Survivability analysis of ad hoc wireless network architecture," Proceedings of the IFIP-TC6/European Commission International Workshop on Mobile and Wireless Communication Networks, pp. 31−46, May 16−17, 2000.

[7] J. Short, R. Bagrodia, and L. Kleinrock, "Mobile wireless network system simulation," Wireless Network Journal, Vol. 1, No. 4, 1995.

[8] R. J. Ellison, D. A. Fisher, and R. C. Linger, "An approach to survivable systems," In: NATO IST Symposium on Protecting Information Systems in the 21st Century, 1999.

[9] M. S. Deutsch and R. R.Willis, "Software quality engineering: A total technical and management approach," IST Symposium on Protecting Information Systems in the 21st Century, 1999, Englewood Cliffs, NJ: Prentice- Hall, 1988.

[10] B. Ellison, D. Fisher, R. Linger, H. Lipson, T. Longstaff, and N. Mead, "Survivable network systems: An emerging discipline," Technical Report CMU/SEI-97-TR-013, Software Engineering Institute, Carnegie Mellon University, November 1997.

[11] D. Moitrasoumyo and L. K. Suresh, "A simulation model of managing survivability of network of emergent Systems," [J], TechniealRePort CMU/SEI-2000-TR-020, 2000.

[12] K. J. Sanjay, M. W. Jeannette, and C. L. Richard, "Survivability Analysis of Network Spesifications," In: Workshop on Depended ability Despite Malicious Fault, 2000 Intemational Conefreneeon Dependable Systemsand Networks (DSN2000); 2000, NewYork USA: IEEE ComPuterSoeiety: 2000, June 25−28, 2000.

[13] R. W. Makr, "The quantitative impact of survivable network architectural on service availbaility," [J], IEEE Communications Magazine, 36(5): pp. 71−77, 1998.

[14] D. Y. Chen, S. Garg, and K. S. Trivedi, "Network survivability performance evaluation: A quantitative approach with applications in wireless ad-hoc networks," Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems, Atlanta, Georgia, USA, pp. 28−28, September 2002.

[15] Dahlberg, T. S. Ramaswamy, and D. Tipper, "Issues in the survivability of wireless networks," Proceedings IEEE Mobile and Wireless Communication Networks Workshop, May 1997.

[16] B.Wu, J. M. Chen, and J. Wu, "A survey of attack and countermeasures in mobile ad hoc networks," Wireless Network Security, Springer US, pp. 103−135, 2007.

[17] G. F. Riley, "The georgia tech network simulator," In Proceedings of the ACM SIGCOMM, New York, pp. 5−12, August 2003.

J. Software Engineering & Applications, **2009, 2:55-59**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

A Semantic Vector Retrieval Model for Desktop Documents

Sheng Li [1]

[1]School of Information, Zhongnan University of Economic and Law, Wuhan, China
Email: kinglisheng@163.com

Received December 3rd, 2008; revised January 29th, 2009; accepted February 18th, 2009.

ABSTRACT

The paper provides a semantic vector retrieval model for desktop documents based on the ontology. Comparing with traditional vector space model, the semantic model using semantic and ontology technology to solve several problems that traditional model could not overcome such as the shortcomings of weight computing based on statistical method, the expression of semantic relations between different keywords, the description of document semantic vectors and the similarity calculating, etc. Finally, the experimental results show that the retrieval ability of our new model has significant improvement both on recall and precision.

Keywords: *Semantic Desktop*, *Information Retrieval*, *Ontology*, *Vector Retrieval Model*

1. Introduction

As an important branch of the semantic Web [1] technology, the semantic desktop indicates the development direction of desktop management technology in the future [2]. In order to implement semantic desktop retrieval, a certain information retrieval model is required, and it is an important research topic of information retrieval. At present, researchers provide a variety of information retrieval model from different angles such as probabilistic retrieval model, fuzzy retrieval model, and vector space retrieval model (VSM) [3]. According to them, the vector space model is the most effective one to express the structure of documents.

The main advantage of traditional vector space model is its simplicity, which could describe unstructured documents with the form of vectors, making it possible to use various mathematic methods to be dealt with. Therefore, we consider using ontology-based semantic information management methods to improve traditional vector space model, creating a semantic vector space model.

2. Traditional Vector Space Model

In the vector space model, the characteristic item t (also known as the index item) is the basic language unit appearing in document d, which could represent some character of the document. The weight of characteristic item is ω_{ik}, which reflects the ability of characteristic item t_k describing document d. The characteristic item frequency tf_{ik} and the inverse document frequency idf_k are used to calculate the value of ω_{ik} with the formula that $\omega_{ik} = tf_{ik} \times idf_k = tf_{ik} \times (\log_2(N/n_k)+1)$, Where tf_{ik} is the frequency of characteristic item t_k in document d_i, and N is the number of documents, n_k is the number of documents that involved the characteristic item t_k. From this formula, we can see that the value of ω_{ik} increases with tf_{ik} and decreases with n_k.

The distance between two document vectors is represented by similarity. The similarity between document d_i and d_j is defined as the cosine of the angle between two vectors:

$$Sim(d_i,d_j) = \cos\theta = \frac{\sum_{k=1}^{m}\omega_{ik}\times\omega_{jk}}{\sqrt{(\sum_{k=1}^{m}\omega_{ik}^2)(\sum_{k=1}^{m}\omega_{jk}^2)}} \tag{1}$$

During the procedure of query matching, the Boolean model could be used to realize the vector conversion of query condition QS.

$$q_j = \begin{cases} 1, & if\ \ t_j \in QS, \\ 0, & else. \end{cases}$$

The information retrieval algorithm based on the aforementioned basic knowledge is as follows:

1) Creating characteristic item database: Input the characteristic item of documents set, and creating characteristic item database;

2) Creating document information base: Input the content of documents into database, and creating the document information database;

3) Creating document vector database: For each record in document information base, computing its characteristic item weight by formula introduced before, and founding its corresponding document vector;

4) Document query: The user input query condition. Then, acquire eligible document vector by Boolean model, computing the similarity between the query condition and each document by Formula (1);

5) Output the ranking result: According to the similarities computed in step 4), output the query result.

3. The Features of New Model

Though the semantic vector space model draws on some thinking of traditional vector space model, it make some useful improvements based on the specific features of semantic information expression. The main features of semantic vector space model include:

1) The elements and dimension of semantic vector space are different from traditional one. In semantic vector space model, the document characteristic item sequence is not represented by the keywords as usual but the concepts extracted from documents. These concepts contain rich meaning in the ontology. At the same time, for each concept in the concept space, there is a corresponding list to describe. The list represents a vector in the property space. Therefore, each semantic vector in this model is composed of a 2D vector. So, the description capacity of semantic model is better than the traditional one.

2) The method for determining each item's weight is different between semantic vector space model and traditional one. In the semantic model, the weight of an item is related to not only the frequency of a keyword, but also the description of corresponding concept involved in the document. In addition, the TFIDF function in traditional model cannot accurately reflect the distribution of items in the documentation set. In semantic vector space model, the items in different position of a document will be set with different weights. For example, the items appearing in the title of one document will be heavier than the ones appearing in the abstract.

3) The two models use different algorithm to compute the similarity. In the semantic vector space model, the comparability and relativity between two concepts are fully taken into account. For example, in traditional vector space model, the words "People", "Person", and "Human" are totally different concepts, but these words could be conclude as one concept according to corresponding structures or relationships.

4) Besides the differences introduced above, the most important feature of semantic model is the using of ontology as a carrier of information. Comparing with traditional text retrieval methods, the new model involved the semantic information in the ontology.

4. Ontology Creating

Except for the differences introduced in last section, an important character of SVM is the usage of ontology as an information carrier.

The ontology could be seen as a specification of conceptualizations, it defines a group of concepts. Commonly, ontology could be divided into general ontology such as WordNet [4] and domain ontology that describe concepts in some special domain. In this paper, we only focus on ontologies in computer science domain.

4.1 The Relationships in the Ontology

In the ontology, concepts link themselves with other concepts through relationships. In the hierarchical structure graph of ontology, each edge represents a relationship. Three most common relationships are "Is-A", "Part-Of" and "Entity Relationship" [5].

1) Is-A Relationship: It describes the relationship of Generalization. For example, "Entity Extraction" Is-A "Information Extraction";

2) Part-Of Relationship: It describes the containing relationship between concepts. For example, the "CPU" is a Part-Of "Computer";

3) Entity Relationship: It describes the member relationship between a concept and its individual object. For example, "T. Berners-Lee" is an entity of concept "author".

4.2 The Structure in the Ontology

According to the basic principles of ontology and the ACM Topic Hierarchy [6], we create ontology to describe the terms about computer science, called "CmpOnto". Then, the ontology "SwetoDblp_2" is created through the extension of SwetoDblp [7] on the aspect of research field and keywords. The segment of ontology CmpOnto is as follows:

```
<owl:Class
rdf:about="http://www.acm.org/class/1998/acm#H.3">
  <rdfs:label>INFORMATION STORAGE AND
RETRIEVAL</rdfs:label>
  <rdfs:subClassOf
rdf:resource="http://www.acm.org/class/1998/acm#H"/>
</owl:Class>
      ...
<owl:Class
rdf:about="http://www.acm.org/class/1998/acm#H.3.3">
  <rdfs:label>Information Search and Retrieval</rdfs:label>
  <rdfs:subClassOf
rdf:resource="http://www.acm.org/class/1998/acm#H.3"/>
  <owl:disjointWith>
<owl:Class
rdf:ID="http://www.acm.org/class/1998/acm#H.3.1">
      <owl:Class
rdf:ID="http://www.acm.org/class/1998/acm#H.3.2">
    ...
  </owl:disjointWith>
</owl:Class>
```

The segment of ontology SwetoDblp_2 is as follows:

```
<owl:Class
rdf:about="http://lsdis.cs.uga.edu/projects/semdis/opus#Article"
>
```

```
  <rdfs:label>Article</rdfs:label>
  <rdfs:subClassOf
rdf:resource="http://lsdis.cs.uga.edu/projects/semdis/opus#Publ
ication"/>
  <rdfs:comment>An article from a journal or maga-
zine.</rdfs:comment>
  <owl:equivalentClass
rdf:resource="http://knowledgeweb.semanticweb.org/semanticp
ortal/OWL/Documentation_Ontology.owl#Article_in_Journal"
/>
<owl:equivalentClass
rdf:resource="http://sw-portal.deri.org/ontologies/swportal#Arti
cle" />
 <owl:equivalentClass
rdf:resource="http://purl.org/net/nknouf/ns/bibtex#Article" />
</owl:Class>
    ...
<owl:ObjectProperty
rdf:about="http://lsdis.cs.uga.edu/projects/semdis/opus#at_univ
ersity">
 <rdfs:comment>Indicates that a publication originates or is
related to a specific University.</rdfs:comment>
 <rdfs:label>at university</rdfs:label>
 <rdfs:range
rdf:resource="http://lsdis.cs.uga.edu/projects/semdis/opus#Univ
ersity"/>
 <rdfs:domain
rdf:resource="http://lsdis.cs.uga.edu/projects/semdis/opus#Publ
ication"/>
</owl:ObjectProperty>
```

5. Computing the Semantic Similarity

During the procedure of information retrieval based on semantic similarity, the concepts and properties in the vector are processed respectively. Considering the relativity between different conceptual entities and comparable properties, the method for measuring the concept similarity and the property similarity are introduced. Finally, the semantic similarity algorithm was provided.

5.1 The Concept Similarity

Ontology uses hierarchical tree structure to describe the logical relationship between concepts, which is the semantic basis for our retrieval algorithm. Since there is certain relativity between different concepts, we use concept similarity to describe and measure it in order to improve the precision of retrieval. Before computing the concept similarity, we give 3 definitions for different kinds of relationship between concepts as following:

Definition 1: The homology concepts. In the hierarchical tree structure of ontology, concept A and concept B are homology concepts if the node of concept A is the ancestor node of concept B. Call A is the nearest root concept of B, notes as *R(A,B)*; The distance between A and B is $d(A,B) = dep(B) - dep(A)$, where $dep(C)$ is the depth of node C in the hierarchical tree structure.

Definition 2: The non-homologous concepts. In the hierarchical tree structure of ontology, concept A and concept B are non-homology concepts if concept A is neither the ancestor node nor the descendant node of concept B; If R is the nearest ancestor node of both A and B, Call R is the nearest root concept of A and B, notes as *R* (*A*, *B*); The distance between A and B is $d(A,B) = d(A,R) + d(B,R)$

Definition 3: The semantic related concepts. Concept C is the semantic related concept of A and B, if and only if C satisfy the following conditions: If concept A and B are homology concepts, C exists in the sub-trees with root of A but not exists in the sub-trees with root of B; if concept A and B are non-homology concepts, C exists in the sub-trees with root of R, but not exits in the sub-trees with root of A or B.

Figure 1 shows details of the relationships described above. According to these definitions, the structure similarity between concept A and concept B is:

$$Sim(A,B)' = \begin{cases} (1-\dfrac{\alpha}{dep(R(A,B))+1})\times\dfrac{\beta}{d(A,B)}\times\dfrac{son(B)}{son(A)}, \\ \text{if } d(A,B)\neq 0, \text{and A,B are homology concepts;} \\ (1-\dfrac{\alpha}{dep(R(A,B))+1})\times\dfrac{\beta}{d(A,B)}\times\dfrac{son(A)+son(B)}{son(R)}, \\ \text{if } d(A,B)\neq 0, \text{and A,B are non-homology concepts;} \\ 1, \text{ if } d(A,B)=0. \end{cases} \tag{2}$$

where son(*C*) present the total number of nodes in subtree with the root of concept C. The parameter α and βis used to adjust the weight of *dep*(*R*(*A*,*B*)) and $d(A,B)$, whose range is (0, 1), and setting by filed experts.

According to formula given above, the concept similarity decreases with the distance between concepts. At the same time, for two concepts, the deeper the nearest root they have, the more common properties they should have, and the more similar they should be. Further more, the number of nodes in the sub-tree and semantic related concepts are also important factors during the computing of similarity.

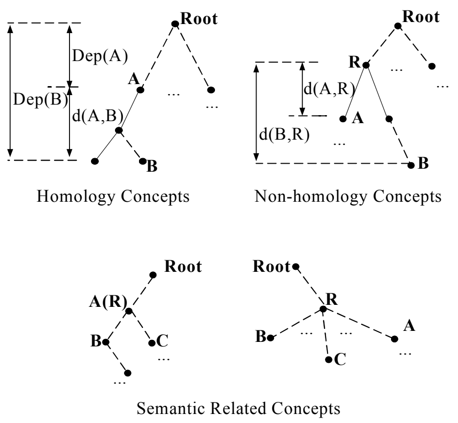


Figure 1. Three patterns of concepts

Finally, the formula defines that the similarity between the same concepts is 1, and the distance between them is 0.

5.2 The Property Similarity

Each concept in the ontology may have several different entities, the main difference among these entities rest with their property values. Further more, different concepts may have same properties. Therefore, not only the concept similarity but also the property similarity should be considered during the computing of similarity between two entities. For the property similarity measuring, we have definition as following:

Definition 4: Suppose I is the entity of concept C, the value of its property **P**$_i$ is p_i, i=1,2, . . . , n. Use I=C[P] to present this entity, where P is the property vector $(p_1,p_2,\dots,p_n)$.

Only the common properties need to process when computing the similarity between property vector $P=(p_1,p_2,\dots,p_m)$ and $Q=(q_1,q_2,...,q_n)$.

At first, transform the property vectors P and Q into common property vectors $P'=(p'_1,p'_2,....,p'_r)$ and $Q'=(q'_1,q'_2,...,q'_r)$. Then, according to the properties defined in the ontology and the similarity of property value, the property similarity of vector P and Q is given:

$$Sim_p(P,Q)=Sim_p(P',Q')=\sum_{i=1}^{r}\frac{\mu_i+\gamma_i}{2}.Sim_i(p'_i,q'_i) \quad (3)$$

where μ_i and γ_i are weights of property p'_i and q'_i respectively in their property vector, which are preset in the ontology; $Sim_i(p'_i,q'_i)$ is the similarity of property values, which is preset by field expert in the ontology. For example, the similarity between property value “Data mining” and “Information Retrieval” is 0.7, and that between “Data mining” and “Network” is 0.1. The range of $Sim_p(P,Q)$ is [0,1].

5.3 The Semantic Similarity

After computing the concept similarity of semantic vector and the property similarity of conceptual entity, we can get the final semantic similarity of semantic vector.

Suppose $V_1=(A_1[P_1],...,A_m[P_m])$ and V_2 = $(B_1[Q_1],...,B_n[Q_n])$ are two semantic vectors. The semantic similarity between V_1 and V_2 is:

$$Sim_V(V_1,V_2)=$$
$$\frac{1}{m}\sum_{i=1}^{m}\underset{j}{Max}\left(\omega\cdot Sim_C(A_{i,}B_j)+(1-\omega)\cdot Sim_P(P_{i,}Q_j)\right) \quad (4)$$

where ω is the weight of concept similarity, and its range is [0, 1].

Now, the main retrieval algorithm is as follows:

```
Begin
1) Initialize the documentation set, then load the user query vector $V_1$ and deciding its document clustering;
2) Load the semantic index file of documents, initializing the semantic vector $V_2$;
3) For each vector in the document clustering includes $V_1$. if current vector has never been processed then continue; else, process the next vector;
4) Compute all the concept similarity between concepts in $V_1$ and $V_2$;
5) Compute all the property similarity between concepts in $V_1$ and $V_2$;
6) Compute the semantic similarity between $V_1$ and $V_2$, insert $V_2$ into list S with descending order;
7) Output top n items in list S as retrieval results;
End.
```

6. Experiment and Analysis

In order to verify the effectiveness of our method, we design a prototype system and chose 100 abstracts downloading from DBLP as retrieval target document. In this prototype system, we use ontologies CmpOnto and SwetoDblp_2 introduced in Section 4.

In the experiment, the depth of ontology concept tree is 5, the range of $dep(R(A,B))$ in Formula (2) is [1,5], and the value of $d(A,B)$ is an integer from 1 to 10; both the value of weight α and β is 0.5; the μ_i and γ_i are parameters preset in the ontology, which could be gained by statistical method. The value of ω in Formula (4) will make influence on the retrieval results ranking. In order to choose proper ω, we implement primary experiment for analysis and choosing 0.8 as the optimal value of ω.

The first step of experiment is document pretreatment. Each document is described by a semantic eigenvector V_2 including 1 to 4 conceptual entities. We can find that the average precision of retrieval increase from 60% to 80% according to the increase of concepts in V_2. The corresponding results are shown in Table 1.

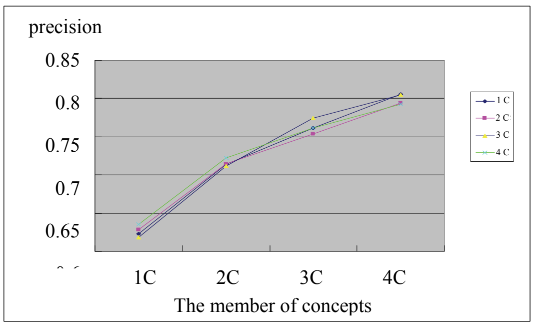


Figure 2. The influence of concepts in V_2 on the precision

Table 1. The influence of concepts in V_2 on the precision

V_1 \ V_2	1C	2C	3C	4C
1 C	0.619	0.711	0.759	0.802
2 C	0.625	0.716	0.752	0.796
3 C	0.630	0.711	0.771	0.803
4 C	0.633	0.724	0.763	0.797

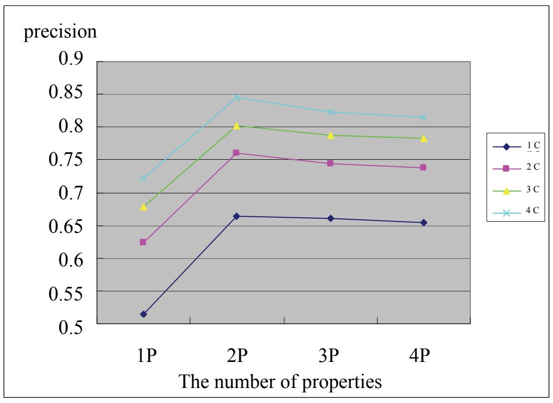


Figure 3. The influence of properties in V_1 on the precision

Table 2. The influence of properties in V_1 on the precision

V_2 \ V_1	1C	2C	3C	4C
1 P	0.513	0.621	0.675	0.721
2 P	0.662	0.764	0.804	0.859
3 P	0.649	0.741	0.785	0.821
4 P	0.637	0.739	0.781	0.811

Table 3. The comparing of different retrieval models

Documents \ Precision	Keywords Retrieval	Semantic Retrieval
5	74.2%	88.3%
10	66.5%	82.5%
15	58.4%	75.5%
20	50.3%	71.2%
25	43.9%	65.8%
30	37.7%	58.5%
35	34.3%	50.4%
40	27.4%	47.2%
45	21.6%	42.9%
50	19.4%	36.3%
Average	43.37%	61.86%

Figure 2 could reflect the relationship between the number of concepts in V_2 and the precision of query more directly.

Further more, statistical results show that the number of properties in the conceptual entity could also make influence on the precision. When the number of properties is 2, the effects go best. If a concept has too many properties, some proper target will be missed because of so many restrictive conditions. The corresponding results are shown in Table 2.

The Figure 3 is corresponding to Table 2.

In addition, we compare our new model with traditional VSM model based on keywords. The number of documents and the precision of retrieval are shown in Table 3. The average precision of semantic retrieval is 61.86%, but only 43.37% by traditional method in the same documentation set. According to the experimental data and analysis above, we know that the ontology could play a positive role in upgrading the precision of retrieval.

7. Conclusions

This paper provides a semantic retrieval model based on the ontology for desktop documents. Comparing with traditional vector space model, the new model using semantic and ontology technology to solve a series of problems that traditional model could not overcome. The experimental results prove the effectiveness of this new model.

In addition, the individual analyses for retrieval results tell us that there is little distinction in result ranking by different retrieval methods. The main reason for precision upgrading is that the semantic retrieval method could reduce the similarity of incorrect results, so that the correct result could be ranked in the front position. Therefore, how to re-rank and optimize the retrieval results is an important task, and it is our main item in the next stage.

REFERENCES

[1] B. Lee, Hendler, and Lassila, "The semantic web," Scientific American, Vol. 34, pp. 34–43, 2001.

[2] S. Decker and M. Frank, "The social semantic desktop," WWW 2004 Workshop Application Design, Development and Implementation Issues in the Semantic Web, 2004.

[3] I. R. Silva, J. N. Souza, and K. S. Santos, "Dependence among terms in vector space model," Database Engineering and Applications Symposium, pp. 97–102, 2004.

[4] G. A. Millet, "Wordnet: An electronic lexical database," Communications of the ACM, 38(11): pp. 39–41, 1995.

[5] G. Asian and D. McLeod, "Semantic heterogeneity resolution in federated database by metadata implan- tation and stepwise evolution," The VLDB Journal, the International Journal on Very Large Databases, Vol. 18, pp. 22–31, 1999.

[6] ACM Topic: http://www.acm.org/class/.

[7] B. Aleman-Meza, F. Hakimpour, I. B. Arpinar, and A. P. Sheth, "SwetoDblp ontology of Computer Science publications," Web Semantics: Science, Services and Agents on the World, pp. 151–155, 2007.

J. Software Engineering & Applications, **2009, 2:60-65**
Published Online April 2009 in SciRes (www.SciRP.org/journal/jsea)

FEL-H Robust Control Real-Time Scheduling

Bing Du[1], **Chun Ruan**[2]

[1]School of Electrical and Information Engineering, University of Sydney, Australia, [2]School of Computing and Mathematics, University of Western Sydney, Australia
Email: bing@ee.usyd.edu.au, c.ruan@uws.edu.au

Received January 26th, 2009; revised February 28th, 2009; accepted March 25th, 2009.

ABSTRACT

The existing scheduling algorithms cannot adequately support modern embedded real-time applications. An important challenge for future research is how to model and introduce control mechanisms to real-time systems to improve real-time performance, and to allow the system to adapt to changes in the environment, the workload, or to changes in the system architecture due to failures. In this paper, we pursue this goal by formulating and simulating new real-time scheduling models that enable us to easily analyse feedback scheduling with various constraints, overload and disturbance, and by designing a robust, adaptive scheduler that responds gracefully to overload with robust H∞ and feedback error learning control.

Keywords: *Robust*, *Real-Time Scheduling*, *Feedbacke*, *Simulation*, *Feedback*

1. Introduction

Feedback control is a powerful tool to make real-time scheduling robust towards external and internal disturbances and uncertainties. Feedback techniques were originally proposed in time sharing systems and have been successively applied to real-time and multimedia systems. Seto *et al*. [1] proposed integrating computer-control and real-time system design so that the performance of a task is a function of its sampling frequency, and identified an optimization problem to find a set of optimal task periods. Lu *et al*. [2] proposed a feedback scheduler based on earliest-deadline first scheduling (EDF), PID controller, and a more theoretically founded approach. Using a PID controller is often apposite in many industrial applications and in feedback control scheduling, but robustness is not guaranteed. Papers [3] integrated feedback control with model-based prediction to anticipate and correct future delay fluctuation. [4] proposed measuring, quantifying, adapting and bounding miss ratio and average system utilisation. These techniques addressed feedback priority-based scheduling literature to ensure that no deadlines are missed. Cervin *et al*. [5] combined feedback with feedforward to allow the scheduler to compensate for resource changing before any overload occurred. Their techniques are tailored to computing systems that may be modelled as sets of digital control loops.

However, no theoretical analysis has been provided about how to model a real-time computing system that includes the effects of the sampling rate, the jitter, actuation, plant uncertainty and nonlinearity, and how to design a robust real-time controller to optimise soft real-time system performance. The main obstacle that prevents control theories from being applied effectively in computing systems is how to construct a computing model that works in open and unpredictable environments. On the other hand, existing feedback scheduling approaches find it difficult to guarantee robust performance properties, as they only use a simplistic maximum constant model and “classical” proportional-integral-derivative PID to design a complex real-time scheduling system. In many cases, the PID controller design techniques are a satisfactory solution. It seems unnecessary to apply more powerful tools. However, when the scheduling system dynamics are complex and poorly modeled, or when the performance specifications are particularly stringent, no solution is forthcoming.

This paper aims at introducing advanced modern control theory to analyze and design real-time systems. The goal of our research is to investigate a real-time scheduling model that can be applied easily to different real-time systems, and proposes a new control scheduling algorithm that is more effective than PID feedback scheduling. We present a two-tank system that is used to simulate a dynamic real-time scheduling model and FEL-H (Feedback error learning and H∞ control) scheduling. The main contributions of this paper are as follows:

1) We use modern control theory as a theoretical foundation to analyze and design adaptive real-time scheduling. In contrast with most of the existing feedback scheduling algorithms that use an ad-hoc manner or PID algorithms, we employ H∞ and FEL control theory as a rigorous methodology to achieve faster response speed and more robust performance guarantees in unpredictable environments.

2) Traditional real-time scheduling theories depend on accurate a priori knowledge of the system workload pa-

rameters. Existing feedback scheduling algorithms cannot respond quickly to workload or model changes and guarantee robust system performance. Our scheduling algorithm, based on feedback error learning control, always tracks the scheduling system accurately and quickly. This feature is especially valuable for performance- critical systems such as on online trading, stock, e-business servers and defense applications. We also consider how to obtain an optimal sampling period and compensate for jitter that is usually not taken into account in existing feedback scheduling. Our H feedback control design methodology provides robust and analytical performance guarantees for open systems, despite workload uncertainties.

3) Unlike traditional physical electrical and mechanical control systems that have a finite set of ordinary differential equations as a design model, the dynamics of real-time computing systems are too complicated to capture the essential features of the scheduling systems. We propose new H∞-norm performance indexes that include the effects of inputs and disturbances on error and control signals. A two-tank system is analysed to simulate a dynamic scheduling model. This model enables us to model and analyse feedback scheduling with various constraints, overload and disturbance more exactly and easily. Furthermore, our new robust FEL-H feedback scheduling integrates H∞ robust optimal control theory, feedback error learning control theory and scheduling theories that do not require precise system model parameters.

4) The existing simulators are available only for a few schedulers and task models, and do not support new scheduling policies, such as H∞-norm feedback scheduling. Our feedback control scheduling simulator (FCSS) allows us to explore various approaches of feedback control real-time scheduling and constraints.

The rest of the paper is organised as follows. We present FEL-H scheduling architecture in Section 2. Section 3 gives how to model a FEL-H real-time scheduling system. In Section 4, we show the robust FEL-H feedback scheduling design. Experiment is discussed in Section 5. The paper is concluded, and suggestions for future work are presented in Section 6.

2. FEL-H Scheduling Architecture

To apply control theory methodology to scheduling, the controller should not only stabilize the nominal real-time kernel, but also meet performance specifications for all possible real-time kernels defined by the uncertainty. A typical FEL-H control scheduling architecture is composed of a feedforward controller Q, feedback controller K, task actuator, QoS actuator, EDF scheduler and CPU (as illustrated in Figure 1). The CPU, EDF scheduler, QoS actuator and task actuator can be treated as a basic scheduling model P. The objective of the control is to minimise the errors between the reference utilisation, deadline miss ratio, and system output utilisation U and deadline miss ratio M.

The objective of the control is to minimise the errors between the reference utilisation, deadline miss ratio, and system output utilisation and deadline miss ratio. An H ∞ controller attempts to keep the CPU utilization U at a high level, avoid overload, distribute the computing resources, and maintain the number of missed deadlines as low as possible. It computes the amount of CPU load that is added into or reduced from the system. An FEL controller will respond rapidly to nonlinear saturation, keep errors near 0 and pull U to return to a linear range. The task actuator controls the amount of workload into the system, and the QoS actuator adjusts the workload inside the system, so that the system accepts as many tasks as possible while minimising the deadline-miss ratio of all the submitted tasks.

The sensor monitors and measures the controlled variables and sends the M(k) and U(k) back to the controller.

The feedforward controller, which contains tunable parameters, controls the utilisation to respond quickly to workload changes. The H∞ feedback controller enables the scheduling system to guarantee robust system performance and to compensate for model error. The output of the H∞ controller regulates and trains the inverse scheduling dynamics model on-line. Therefore, the FEL-H scheduling controller captures, trains and controls at the same time. The EDF scheduler schedules the accepted tasks according to the EDF policy, which can achieve a deadline miss ratio of 0% if requested utilisation is less than 100%. The FEL-H controller is a control computation algorithm to be executed in every sampling period h. A set of tasks will share the CPU. These tasks are control tasks or scheduling tasks. They perform sampling, control computation and actuation. The deadline-miss ratio and CPU utilisation reference are Mr and Ur. The robust requirement is introduced into the design by imposing individual weighting functions on the uncertainties, external disturbances, and the performance specifications.

The Task actuator decides which workload will be allowed into the system and which will be denied. Whether the system should admit or reject requests depends on the task's request utilisation. If the requested utilisation of the incoming task results in a total CPU requirement of all tasks less than Ur = 90%, the task will be admitted, otherwise it will be rejected.

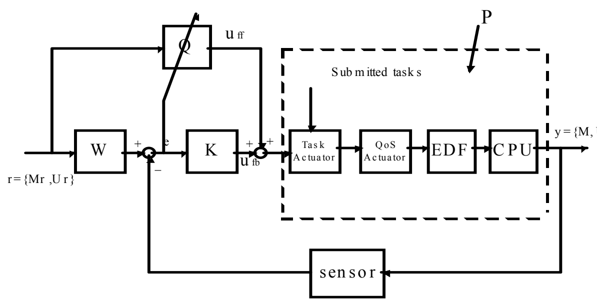


Figure 1. Architecture of FEL-H control scheduling

The task actuator may further adjust to admit more workload if the QoS actuator degrades the QoS level. It can also reject more workload if the QoS actuator upgrades the QoS level.

In the QoS control scheme, each task has different resource requirements for each discrete quality level. The system maintains a single value that represents the quality level of the overall system and is called the "QoS level". The QoS level determines how to allocate resources to each task. If the QoS level downgrades, the resource allocations of some tasks are decreased. Although many QoS control policies are proposed, they are not suited for real-time systems that need to keep the timing constraints. We use a table containing the resource requirements of all tasks [8]. Resource allocations for tasks and total resource utilisation can be obtained from the table. The QoS table allows the system designer to specify a QoS control policy. The QoS actuator changes the requested utilisation in the system by adjusting the service levels of accepted tasks. It will return the portion of tasks not accommodated to the task actuator.

The FEL-H architecture can use different real-time scheduling policies (such as EDF or Rate/Deadline Monotonic) as basic schedulers to schedule admitted tasks. There are significant different performance references for different basic scheduler policies when we design the FEL-H scheduling system.

3. Modeling a FEL-H Real-Time Scheduling System

To implement robust scheduling, our research [6,7,9] proposes a two-tank system model to emulate a scheduling system (Figure 2). The progress of a task request queue is similar to a fluid flow into a multi-tank system. They have the same dynamics due to their intrinsic queuing structure. The system output is the utilisation that is mapped to the liquid level h in tank 2 and input is represented by admitted task R that is mapped to the flow u into tank 1.

The tasks accepted by the CPU are simulated as liquid flowing into the CPU, and the tasks completed by the CPU are viewed as liquid flowing out of the CPU. The CPU can be viewed as a liquid tank that inputs liquid (accepting tasks) and outputs liquid (completing tasks). The tasks submitted to a real-time scheduling system are viewed as liquid that wants to flow into a real-time scheduling system. They may not be equal to the tasks accepted by the CPU. The task actuator decides whether to accept or reject submitted tasks. Therefore, submitted tasks cannot flow directly into the CPU tank. It is impossible to represent tasks accepted by the task actuator and tasks accepted by the CUP if we only use one tank. The tasks that are accepted by the task actuator are simulated as liquid flowing into the real-time scheduling system. The QoS actuator will decide whether this liquid can flow into the CPU tank or not. The part of the liquid that flows into the CPU tank represents the tasks accepted by the CPU. The other part, which does not flow into the CPU

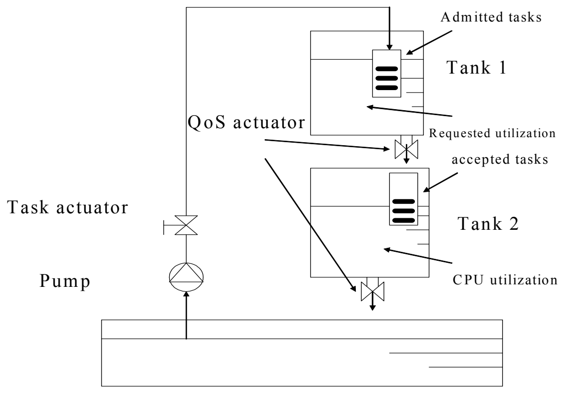


Figure 2. Two-tank system model of a scheduling system

tank, will stay in the real-time scheduling system. This part of the real-time scheduling system can be simulated as another tank. The heights of the liquid levels of the two tanks are coupled together and interact. They are a complex nonlinear, time-varying and multivariable system that is an approximate abstraction of the real-time scheduling system. The two-tank system model is sufficiently accurate to simulate and analyse a real-time scheduling system, as our experiments will show.

Level 2 and level 1 in tank2 and tank1 represent requested utilization and CPU utilization as shown in Figure 2. Our goal is to design a controller that regulates the CPU utilisation, keeping it at 90%. This is mapped to design a controller so that the level in tank 2 is regulated to the reference value by the task actuator and the QoS actuator. The transfer function of the scheduling system can be written as follows:

$$\frac{U(s)}{R(s)} = \frac{K^*}{(T_1 s+1)(T_2 s+1)} \tag{1}$$

where:

$$K^* = \frac{1}{\Phi}\sqrt{\frac{2U_0}{g}}$$

The time constants T_1 and T_2 are related to performance reference points and the QoS actuator that are mapped to the level in the tanks, Φ the outlet's cross-sectional area and the cross-sectional area of the tanks.

4. Design of FEL-H Real-Time Scheduling

We introduce the usual form and generic H∞ block diagram to represent the scheduling systems shown in Figure 3. The FEL-H controller should make the system stable, and satisfy the steady state and transient state performance specifications. The system output comprises the controlled variable miss ratio M(k) and utilisation U(k). The input signals to the scheduling system include the performance reference and disturbance input. The per-

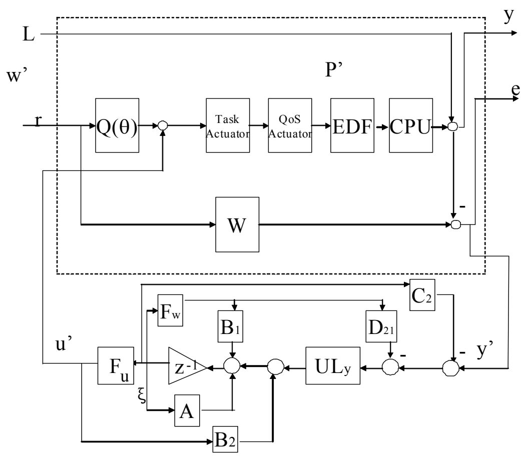


Figure 3. FEL−H Scheduling Controller

formance references Mr and Ur use a step signal. The internal overload that adds the total requested utilisation by the admitted tasks' CPU utilisation variation is modelled as a disturbance L. A step load is used as disturbance because it represents severe load variations. The stabilising H∞ controller minimises the transfer function matrix mapping to. This ensures scheduling-system performance-tracking and workload disturbance attenuation by keeping small.

The H∞ controller of scheduling systems can be described as the central controller (2) (3) (4) (as shown in Figure 3):

$$\xi(k+1)=A\xi(k)+B_1\eta(k)+B_2u(k)-UL_y(y(k)-C_2\xi(k)-D_{21}\eta(k)) \quad (2)$$

$$u(k)=F_u\,\xi(k) \quad (3)$$

$$\eta(k)=F_w\,\xi(k) \quad (4)$$

The FEL controller Q will further minimize as (5).

$$Q(z)=\frac{[4k_w+2T(f_2k_w+c_{w,2})+T^2(f_1k_w+c_{w,1})]z^2+2[T^2(f_1k_w+c_{w,1})-4k_w]z+[4k_w-2T[f_2k_w+c_{w,2}]+T^2(f_1k_w+c_{w,1})]}{[4+2T(f_2-d_{w,2})+T^2(f_1-d_{w,1})]z^2+2[T^2(f_1-d_{w,1})-4]z+[4-2T(f_2-d_{w,2})+T^2(f_1-d_{w,1})]} \quad (5)$$

5. Experiment

We developed a simulation environment to evaluate the performance of our FEL-H scheduling algorithms. The controllers can be designed by using MATLAB tools. Our robust FEL-H scheduling will still achieve performance specifications even with uncertain model parameters and unpredictable operating environments. This lets our FEL-H scheduling be directly applied to different systems and it does not need re-design for every system. We can compute the prefilter, feedback error learning controller and H∞ controller parameters. The sampling period T is 0.5 sec. The results of the FEL controller parameters are listed in Table 1.

The H∞ controller can be derived.

$$\xi(k+1)=\begin{bmatrix}3.45 & 4.71\\ 1.84 & 2.39\end{bmatrix}\xi(k)+\begin{bmatrix}2.34\\ 1.72\end{bmatrix}\eta(k) \quad (4)$$

$$u(k)=\begin{bmatrix}-2.31 & 0.59\end{bmatrix}\xi(k)-1.73\,\eta(k) \quad (5)$$

$$\eta(k)=y(k)-\begin{bmatrix}7.85 & 2.36\end{bmatrix}\xi(k) \quad (6)$$

The utilisation reference should be less than the nominal threshold of the EDF scheduling policy, so that the utilisation error will not remain at 0 when the system is overloaded. Otherwise, the system will stay in overload. The theoretical bound is 100% for EDF and the periodic task set. We set the utilisation reference Ur=90%. The miss ratio reference will change, since different ap- plications have different requirements and tolerances to missed deadlines. For example, the stock-trading transactions have more strict timing constraints than usual online trading. The miss ratio reference is chosen as Mr=2.5% in our experiment.

Table 1. FEL Controller Parameters

w_1	w_2	w_3	k_w	f_1	f_2	c_1	c_2	d_1	d_2
8	16	1	18.43	5.26	5.75	3.59	2.41	7.37	9.82

A set of tasks arrives suddenly with a total CPU utilisation of 150% at the highest QoS level. The workload increases from zero to 150% overload. All experiment algorithms use the same EDF scheduling policies and table-based QoS optimisation algorithm. Every simulation experiment is repeated 22 times to ensure the evaluation is accurate. We describe the experiment's results for EDF scheduling, PID feedback scheduling, H∞ control scheduling, FEL scheduling and FEL-H scheduling algorithms in response to a new arrival 150% overload, and compare the results. Then, we present the performance evaluation of these algorithms in response to the disturbance of the system's internal parameters.

In this section, we present the five different algorithms in response to a 150% overload (as shown in Figure 4).

EDF scheduling has a miss ratio that is too high and fails to provide performance guarantees. The PID controller cannot provide satisfying robust performance guarantees and because the overshoot of miss ratio and average miss ratio are too high, and the durations of the settling time and rise time of U(k) are too long, H∞ feedback can provide very robust performance for system uncertainty, but the response time is too slow. FEL scheduling has the fastest rise time, but the overshoot and average miss ratio are higher than for PID and H∞ scheduling. It cannot provide robust performance guarantees for model uncertainty. FEL-H scheduling can provide the desired robust performance guarantees that consist of fast rise time and settling time, low average miss ratio and high CPU utilisation, as it can obtain a fast response from the FEL controller and more robustness from the H∞ controller.

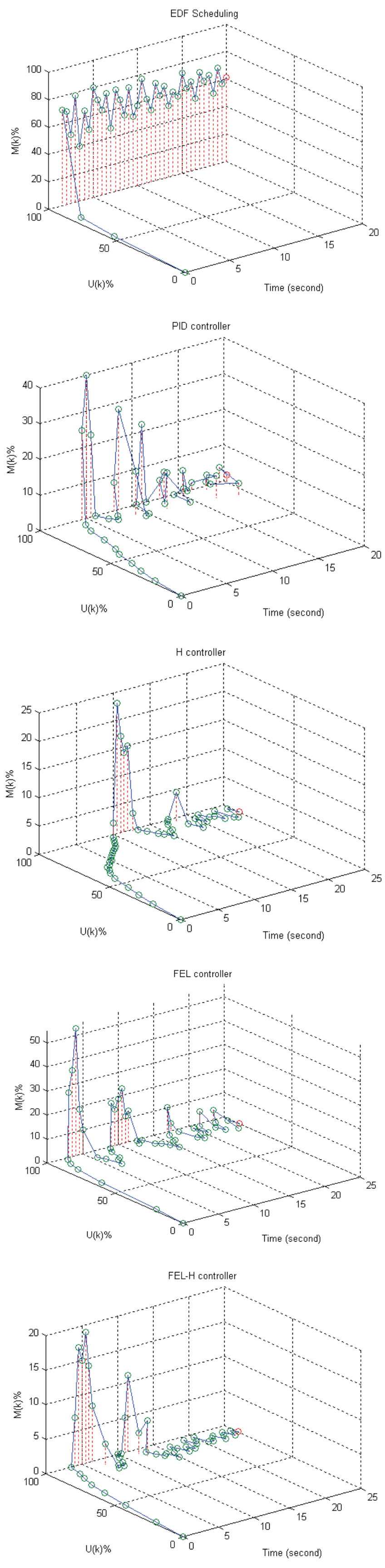


Figure 4. EDF, PID, H∞, FEL and FEL-H scheduling

Five different algorithms are in response to that the model parameters of the scheduling system are changed by 30%, while the new overload is 150%. (as shown in Figure 5). This case can be used to simulate the robustness of the feedback scheduling algorithms, and to demonstrate whether our scheduling algorithms can be applied to different real-time scheduling applications without needing to change the parameters of the controllers. Only the FEL-H scheduling can remain in the steady state while the utilisation U(k) remains close to 90%, and the miss ratio M(k) remains at 0 through running.

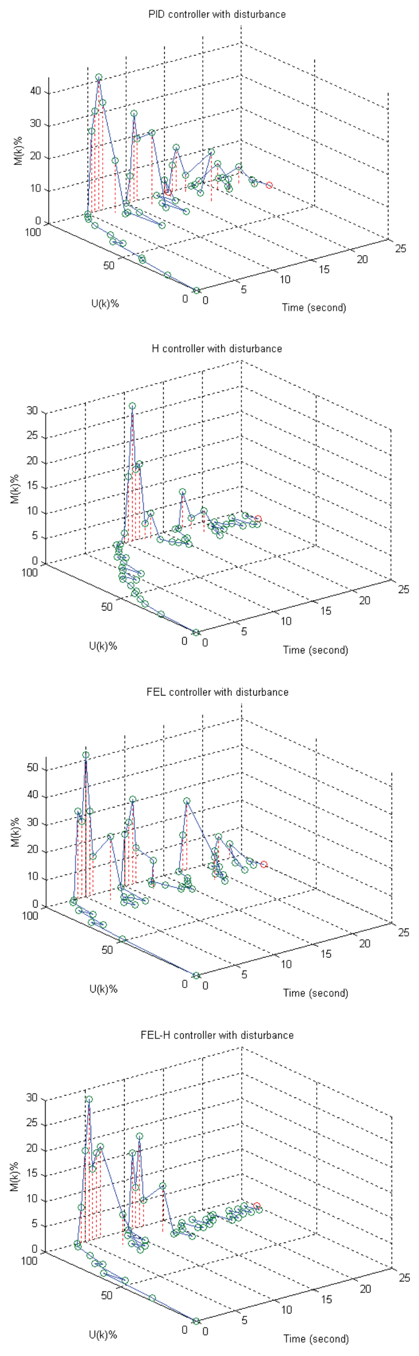


Figure 5. EDF, PID, H∞, FEL and FEL-H scheduling with disturbance

6. Conclusions

In this paper, we integrates H∞ control theory and feedback error learning control theory to model and deal with feedback real-time scheduling with uncertainty and nonlinearity. Our mechanism provides a systematic and theoretic platform for investigating how to deal with uncertainty and additive disturbances for real-time systems. The FEL-H scheduling algorithms can provide robust stability, low deadline miss ratio for real-time system uncertainty parameters and disturbance when the workload changes dramatically. Further work will improve the feedback scheduling scheme so that heterogeneous modelling and design of real-time and embedded system can be integrated. We will apply our FEL-H control scheduler to a realistic real-time system.

REFERENCES

[1] D. Seto, J. P. Lehoczky, L. Sha, and K. F. Shin, "On task schedulability in real-time control systems," In Proceedings 17th IEEE Real-Time Systems Symposium, Washington, DC, pp. 13–21, 1996.

[2] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley, "Performance specifications and metrics for adaptive real-time systems," In Proceedings 21st Systems Symposium, Orlando, Florida, pp. 13–23. 2000.

[3] D. Henriksson, Y. Lu, and T. Abdelzaher, "Improved prediction for Web server delay control," In Proceedings of the Euromicro Conference on Real-Time Systems, Catania, Sicily, pp. 61–68, Italy, 2004.

[4] T. Abdelzaher, V. Sharma, and C. Lu, "A utilization bound for aperiodic tasks and priority driven scheduling," IEEE Transactions on Computers 53(3): pp. 334–350, 2004.

[5] A. Cervin, J. Eker, B. Bernhardsson, and K. E. Årzén, "Feedback-feedforward scheduling of control tasks," Real-Time Systems Vol. 23, No. 1, pp. 113–120, 2002.

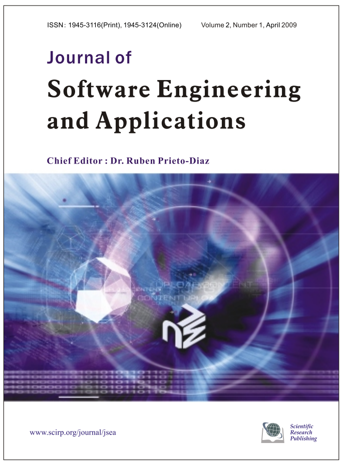
[6] B. Du, and D. Levy, "A robust control real-time scheduling design," special issue: Advanced Control and Real-Time Systems of DCEMP journal, Vol. 13, No. 3–No.4, pp. 335–340, 2005.

[7] B. Du and D. Levy, "H∞ robust scheduling design methodology in real-time systems," In the Proceedings of the 2003 International Conferenceon Embedded Systems and Applications, pp. 285–291, USA, 2003.

[8] S. Tomoyoshi and T. Kosuke, "Table-based QoS control for embedded real-time systems," ACM SIGPLAN Notices Vol. 34, No. 7, pp. 65–72, July 1999.

[9] B. Du and C. Ruan, "Robust feedback scheduling in distributed embedded real-time systems," In the Proceedings of the 6th IEEE/IFIP International Conference On Embedded and Ubiquitous Computing (EUC 2008), pp. 90–96, Shanghai, China, 2008.

Call For Papers


ISSN: 1945-3116(Print), 1945-3124(Online) Volume 2, Number 1, April 2009

Journal of Software Engineering and Applications

Chief Editor : Dr. Ruben Prieto-Diaz

www.scirp.org/journal/jsea

Scientific Research Publishing

Journal of Software Engineering and Applications (JSEA)

ISSN 1945-3116 (print) ISSN 1945-3124 (online)

www.scirp.org/journal/jsea

JSEA publishes four categories of original technical articles: papers, communications, reviews, and discussions. Papers are well-documented final reports of research projects. Communications are shorter and contain noteworthy items of technical interest or ideas required rapid publication. Reviews are synoptic papers on a subject of general interest, with ample literature references, and written for readers with widely varying background. Discussions on published reports, with author rebuttals, form the fourth category of JSEA publications.

Editor-in-Chief

Dr. Ruben Prieto-Diaz, James Madison University , USA

Subject Coverage

- Applications and Case Studies
- Artificial Intelligence Approaches to Software Engineering
- Automated Software Design and Synthesis
- Automated Software Specification
- Component-Based Software Engineering
- Computer-Supported Cooperative Work
- Software Design Methods
- Human-Computer Interaction
- Internet and Information Systems Development
- Knowledge Acquisition
- Multimedia and Hypermedia in Software Engineering
- Object-Oriented Technology
- Patterns and Frameworks
- Process and Workflow Management
- Programming Languages and Software Engineering
- Program Understanding Issues
- Reflection and Metadata Approaches
- Reliability and Fault Tolerance
- Requirements Engineering
- Reverse Engineering
- Security and Privacy
- Software Architecture
- Software Domain Modeling and Meta-Modeling
- Software Engineering Decision Support
- Software Maintenance and Evolution
- Software Process Modeling
- Software Reuse
- Software Testing
- System Applications and Experience
- Tutoring, Help and Documentation Systems

Notes for Prospective Authors

Submitted papers should not have been previously published nor be currently under consideration for publication elsewhere. All papers are refereed through a peer review process. For more details about the submissions, please access the website.

Website and E-Mail

Website: http://www.scirp.org/journal/jsea E-Mail: jsea@scirp.org

TABLE OF CONTENTS

Volume 2, Number 1 **April 2009**

Copyright©2009 SciRes

9771945311006 02