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

Journal of
Software Engineering and Applications

Chief Editor : Dr. Ruben Prieto-Diaz

www.scirp.org/journal/jsea

Scientific
Research

Journal Editorial Board

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

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

CONTENTS

Volume 2 Number 2 July 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., 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., USA.
E-mail: jsea@scirp.org

COPYRIGHT

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: 67-76**
doi:10.4236/jsea.2009.22011 Published Online July 2009 (www.SciRP.org/journal/jsea)

Interpretation of Information Processing Regulations

Sabah Al-Fedaghi

Computer Engineering Department, Kuwait University, Kuwait City, Kuwait.
Email: sabah@eng.kuniv.edu.kw

Received February 2nd, 2009; revised April 14th, 2009; accepted April 15th, 2009.

ABSTRACT

Laws and policies impose many information handling requirements on business practices. Compliance with such regulations requires identification of conflicting interpretations of regulatory conditions. Current software engineering methods extract software requirements by converting legal text into semiformal constraints and rules. In this paper we complement these methods with a state-based model that includes all possibilities of information flow. We show that such a model provides a foundation for the interpretation process.

Keywords: *Software Requirement, Laws, Regulation, Privacy, Personal Identifiable Information*

1. Introduction

Laws, regulations, and policies such as the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule and the Telecommunications Act of 1996 impose many requirements on business practices for handling information. In 2006, 161 billion gigabytes of digital information were created, captured, and replicated [1]. It is estimated "that today, 20% of the digital universe is subject to compliance rules and standards, and about 30% is potentially subject to security applications" [1]. In 2005, more than 20,000 regulations were passed related to creation, storage, access, maintenance, and retention of information [2].

Compliance with these laws, regulations, and policies requires identification of conflicting interpretations of regulatory requirements. The information system needs to be aligned with legal and regulatory requirements in order to be in compliance.

Statements of regulations in legal documents relevant to information processing contain a great deal of natural language ambiguity that makes it difficult to formalize requirements and constraints in software systems. The basic problem can be viewed as how to extract software requirements from regulations.

Researchers have introduced different methods for converting legal language into semiformal specifications; nevertheless, the approaches to interpreting legal text lack compatibility with the software-engineering style of problem solving. A need exists for an underlying information-processing model of the different information processing systems, similar to the classical communication model of sender, receiver, and message. Terms such as "collecting," "processing," "disclosing," and so forth are used loosely, without a pattern tying them together as actions based on information. We will demonstrate such aspects in an example after introducing our model.

So, what is the software-engineering style of problem solving that ought to be applied to interpretation of regulations to meet software requirements? It involves simply constructing an information flow model, and taking into account all possible types of actions utilized in processing information. While it is not practical to take into account every possible interpretation, we propose a state-based model that includes a limited number of possibilities for software responses to all categories of information handling.

2. Related Work

The software engineering field is rich with work related to software requirements for domain and systems descriptions. Methods have been proposed to extract requirements from policies and regulations using formal models [3], semantic parameterization [4], and ontology [5]. Several publications deal with the problem of extracting goals from natural language documents and Internet privacy policies [6,7]. Breaux and Antón [4,8] developed a method to trace the words in regulations to semantic primitives. Giorgini et al. [9] described a framework that enables modeling of actors and goals and their relationships. May et al. [3] introduced a methodology to extract formal models from regulations and applied it to the HIPAA Privacy Rule.

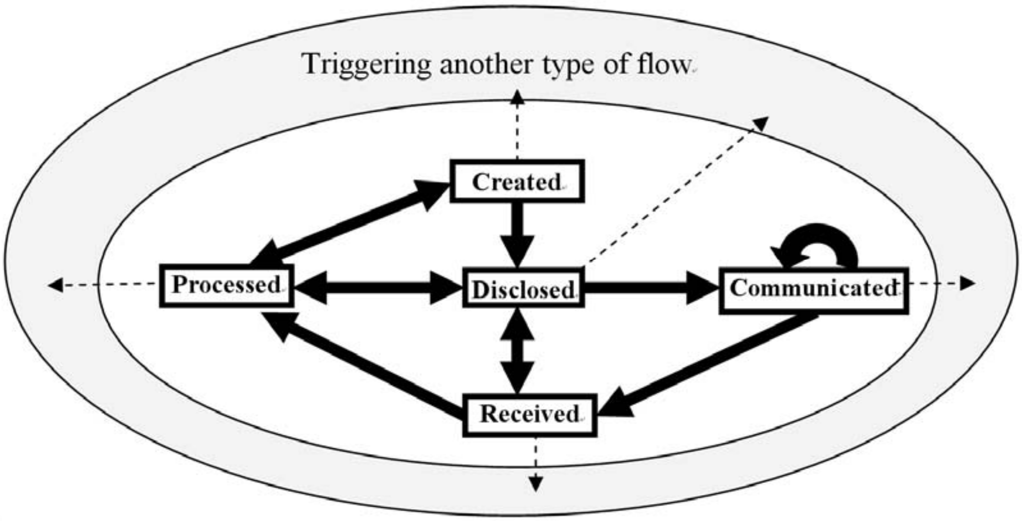


Figure 1. Information states in FM with the possibility of triggering another type of flow

We will concentrate on the recent methodology of Breaux and Antón [10] "to extract access rights and obligations directly from regulation texts . . . [and] present results from applying this methodology to the entire regulation text of the U.S. Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule." In our case, we will discuss a very limited portion of such a commendable venture.

Information processing models have evolved from the classic 1949 Shannon-Weaver communication theory of transference of electrical signals from sender to receiver. Nevertheless, the main aspect of Shannon and Weaver's model is communication, while information appears as a secondary component of the model. The "transmitted materials" in this communication can be information, data, meaningless signals, energy, or, with proper perspective, other entities.

We are interested more in information *properties* than in the communication act itself. Such an approach concentrates solely on information, while communication aspects (as important as they are) become secondary features of the flow of information. This paper introduces an information flow model that includes five stages: collection, processing, creation, disclosure, and communication. Such a model provides a better framework on which to base interpretation of information handling processes.

3. Model of Information Flow

The flow model (FM) has been proposed and used in several applications. In FM, the flow of flowThings indicates movement inside and between spheres. The sphere is the environment of the flow and includes five stages that may be sub-spheres, each with its own five-stage schema. The stages may be named differently; for example, in an information sphere, a stage may be called *communication*, while in action flow, the same stage is called *transferring*.

To illustrate the notion of flowThing, we will assume that the "thing that flows" is information. We use the term *information* to refer to information and misinformation, as in common usage where a reporting statement can be true or false. An information sphere denotes the information environment. The lifecycle of information is a sequence of states in its lifecycle, as follows: 1) *Received*, 2) *Processed* (in a way that changes its form, but not content), 3) *Released*, 4) *Communicated* (to another sphere), and 5) *Created* (i.e., a new piece of information). These five states of information form the main stages of the stream of flow, as illustrated in Figure 1. Each stage may include sub-stages, such as storage and usage.

The states shown in Figure 1 are exclusive in the sense that if information is in one state, then it is not in any of the other four states. Consider a piece of information, σ, possessed by a hospital; σ is thus in one of the following states:

1) σ has just been collected from some source (patient, friend, sent by some agency) and stored in the hospital record, waiting to be used. It is *collected* (row) information that has not yet been processed by the hospital.

2) σ has been processed in some way, converted to another form (e.g., digital), translated, compressed, etc. Also, it may be stored in the hospital information system as *processed* data waiting for some use.

3) σ has actually been created in the hospital as the result of doctor's diagnoses, lab tests, etc. Thus, σ is in the possession of the hospital as *created* data to be used.

4) σ is being released from the hospital infosphere. It is designated *disclosed* information ready for transfer. Analogous to a factory environment, σ represents materials ready to be shipped outside the factory. It may actu-

ally be stored for some period waiting to be transported; nevertheless, its designation as "for export" keeps it in such a state.

5) σ is in a transfer state, being transferred between two infospheres. It has left the disclosure state and will enter the collection state, where it will become collected information in the new infosphere.

It is not possible for processed information to directly become collected information in the same infosphere. Processed information can become collected information in another infosphere by first becoming disclosed information, and then transferred information, in order to arrive at the other environment.

We use this model, called the information flow model (IFM), to classify information in generic theoretical categories that can be applied in any infosphere, including laws and regulations. According to Kurt Lewin, "There is nothing quite as practical as a good theory."

Consider the following extract from the Safety Officer's Briefing Book of the United States Air Force Auxiliary [11]:

Water *vapor* is an invisible *gas*, similar to nitrogen and oxygen, the two gases which make up 98% of our atmosphere. We see clouds and fog when the *gaseous* water *vapor* is cooled sufficiently to allow a change of state from *gas* to *liquid*. We see snow, sleet, and hail when the *liquid* water is further cooled to a *solid* state. And when water changes directly from *gas* to *solid* it becomes frost, through a process called "sublimation." [Italics added.]

Imagine this to be a statement of regulations with no model of water circulation among its states of liquid (water), gas (vapor, fog), and solid (ice). We could probably manage to write safety regulations using terms such as "clouds" and "frost," as in the following rewrite of the previous quote:

Water is an invisible material, similar to nitrogen and oxygen, the two gases which make up 98% of our atmosphere. We see clouds and fog when the water is cooled sufficiently. We see snow, sleet, and hail when the water is further cooled. And when water changes, it becomes frost, through a process called "sublimation."

A great deal of conceptual clarity is lost without specification of the three natural states of water in a model of its circulation among those states. Similarly, information handling regulations not based on the information flow model are simply vague descriptions of the proper way to handle information.

As another example, consider applying the IFM to HIPAA Privacy Rule [12]. According to the Privacy Rule, "A covered entity may *use or disclose*, without an individual's authorization, the psychotherapy notes . . ." [13]. We show that, in the context of IFM, *use* and *disclose* may be interpreted in several ways.

Psychotherapy notes are defined in the Rule as:

. . . notes recorded (in any medium) by a health care provider who is a mental health professional documenting or analyzing . . . that are *separated from the rest of the individual's medical record*. Psychotherapy notes exclude medication prescription and monitoring, counseling session start and stop times, the modalities and frequencies of treatment furnished, results of clinical tests, and any summary of the following items: diagnosis, functional status, the treatment plan, symptoms, prognosis, and progress to date.

A psychotherapist collects information from four sources:

1) The original medical record

2) Information created by the psychotherapist during a session

3) New information revealed by the patient during a session

4) Created information related to the session (e.g., start and stop times: *John's session is at* 4 *pm, John arrived late for the session*).

These clear categorizations of information can be supplemented with the type of descriptions given by the rule (e.g., summary of this type or that type). Each category may require different constraints on its use and disclosure. For example, releasing information of type (2) may need the approval of the psychotherapist who wrote it, while the decision to release information of category (1) has nothing to do with the psychotherapist.

The point here is that the IFM allows more refined interpretation of natural language description involving operations on different types of information, such as disclosure. The example above includes at least three types of disclosures: disclosure of collected information, disclosure of created information, and disclosure of processed information. Each type may need different disclosure constraints [14].

4. Personal Identifying Information

This paper focuses on the HIPAA Privacy Rule; for that purpose we regard personal identifiable information (PII) as the main type of information and object of study [12]. The "circulation of PII" can be modeled in IFM as information flow.

It is typically claimed that what makes data "private" or "personal" is either specific legislation (e.g., a company must not disclose information about its employees) or individual agreements (e.g., a customer has agreed to an electronic retailer's privacy policy); however, this line of thought blurs the difference between personal identifiable information and other "private" or "personal" information. Personal identifiable information has an "objective" definition in the sense that it is independent of such authorities as legislation or agreement.

4.1 Definition of Personal Identifiable Information

In the information sphere, information is classified as personal identifiable information (PII) and non-identifiable information (NII). Personal identifiable information is information *about* singly identifiable persons, called *proprietors*. PII is information that has *referents* who are natural persons. Two types of PII can be identified:

1) Atomic personal information, where the information refers to a single proprietor, e.g., *John is 25 years old*. "Referent" here implies an identifiable (natural) person.

2) Compound personal information, where the information refers to more than one proprietor, e.g., *Mary donated her kidney to Alice*.

Any compound PII is privacy reducible to a set of atomic PIIs [12].

4.2 Elaborated IFM Model for PII

"Handling information" involves observing the progress of information from its arrival at the infosphere through the various information stages until it exits, or disappears from, the "information circulation system." "Pieces of information" circulating in the IFM (Figure 2) are envisioned here as "informational objects" that have an existence within the information realm. This type of information, the so-called "meme," is "a hypothetical unit of cultural transmission conceived not as an inert object but as a quasi-organic entity endowed with the capacity of self-replication" [4].

The flow model makes a piece of information visible as soon as it enters the circulation system of IFM. In most cases, the piece of information then moves repeatedly among the stages of the model. IFM includes types of acts on information (labelled 1 through 14 in Figure 2) that transform information from one stage or sub-stage to another.

Creation stage: In Figure 2, the creation stage includes two attached sub-stages: *Storage* and *Actions*. New information is created (e.g., data mining generates new information). The created information is utilized in some way (called sub-stage actions in Figure 2; e.g., decision making), stored, or immediately moved on to the

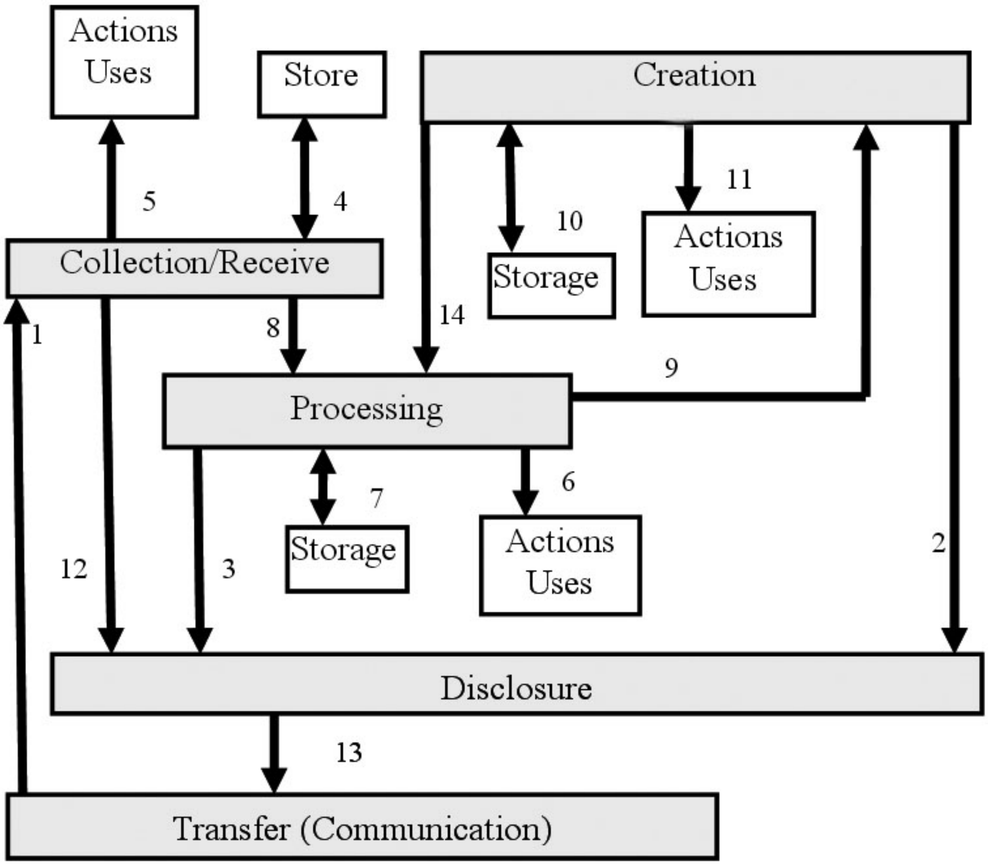


Figure 2. The Information Flow Model (IFM)

stages of processing or departure (disclosure and transfer). In IFM, "action" refers to any utilization of information.

For example, suppose a physician makes a new diagnosis of a disease (creates new information), such as *John Smith has AIDS*. Such new information may be generated by examining (mining) some data. *John Smith has AIDS* is then communicated to an expert (disclosed and transferred), or processed further (cycled through processing-mining-creation-processing) to verify the medical judgment. The physician *creates* (act 9) new information and also *uses* the information in the patient's file for "treatment" (physical action). Another example of *uses* in this context is "home delivery of medicine to a patient" (physical action) that *uses* the address information.

Collection stage: The collection stage is the information acquisition stage, when information is accepted from external suppliers and fed into the IFM circulation system. This stage includes the possibility of *using* the arriving (raw) information; the sub-stage, *Actions*, in IFM is therefore a way for information to exit the system (i.e., arriving information generates the action of physical treatment, without further propagation in IFM). It also includes the possibility of storing the collected information.

Processing stage: The processing stage involves acting on information (e.g., anonymization, data mining, summarizing). Processing is performed on acquired information from the collection stage or the creation stage; see Figure 2. In actual processing, information is modified in form or content.

IFM distinguishes between two types of processing: ordinary processing, which does not produce new or inferred information, and creation of new information (e.g., by mining). An example of creation of new information is the categorization of diverse information used to reach a decision, e.g., *John is a risk*. Other types of processing that do not generate new information, but only change the appearance of information, include comparing, compressing, translating, and deleting.

Disclosure and transfer/communication stages: The disclosure stage involves releasing information to those outside the infosphere. It relies on the transfer stage to carry information from the current infosphere to the collection stage in another infosphere. When information is in the transfer state, it is flowing (e.g., through a channel) between two infospheres.

5. Application to Extraction Constraints

Healthcare regulations, such as HIPAA, are often difficult to grasp and interpret. Interpretation of HIPAA regulations is of utmost importance because of the costs associated with any misinterpretation. According to the U.S. Department of Health and Human Services., the healthcare industry will spend $17.6 billion over the next few years to bring their systems and procedures into compliance with HIPAA [15].

Consequently, we limit this paper to applying the IFM model to the problem of extracting software requirements from HIPPA regulations; however, this focus does not mean the model cannot be applied to any other area with information-handling regulations. Additionally, our treatment is obviously incomplete in the sense that it is not a rewriting of the entire regulation, since it aims mainly to present a general methodology for interpreting informational privacy regulations.

We will concentrate specifically on methodology recently introduced by Breaux and Antón [10] for extracting access rights and obligations directly from the text of regulations. According to Breaux and Antón [10], "'rules' are often precursors to software requirements that must undergo considerable refinement and analysis before they are implementable." Breaux and Antón [10] then present "a methodology to extract access rights and obligations directly from regulation texts . . . to support the software engineering effort to derive security requirements from regulations." The methodology will then "identify and infer six types of data access constraints, handle complex cross-references, resolve ambiguities, and assign required priorities between access rights and obligations to avoid unlawful information disclosures."

The following excerpt from Privacy Rule §164.510 (b)(1)(i) will be used to illustrate use of IFM to develop a complete picture of situations specified by the Rule.

A CE [*Covered Entity*] *may, in accordance with paragraphs* (*b*)(2) *or* (3) *of this section, disclose to a family member, other relative, or a close personal friend of the individual, or any other person identified by the individual, the PHI* [*Protected Health Information*] *directly relevant to such person's involvement with the individual's care or payment related to the individual's healthcare*.

We discuss some of the constraints extracted by Breaux and Antón [10] to exemplify ambiguities uncovered by the definitions of PII and IFM.

Constraint 1: The PHI is directly relevant to the person's involvement in the individual's care.

Comments: We should explicitly declare whether the atomic PII is collected PII, processed PII, or created PII. For example, the proprietor may agree to release his or her telephone number (collected information), but not the results of his or her lab examination (created PII), or his or her doctor's diagnosis (created PII).

Notice that the point here is not *constraints* on the scope of information (e.g., address and phone number); it is, rather, the *right* of the proprietor to be presented with all alternative actions to which he or she can consent, and the *obligations* of the stakeholder to provide the proprietor with complete information for making informed consent. The choices here are as follows:

- Disclosing only PII collected by the stakeholder from other sources

- Disclosing PII processed by the stakeholder, such as implied PII, translated PII, anonymized PII, etc.

- Disclosing created PII by the stakeholder, such as lab results, doctors' diagnoses, etc.

These actions deal with categories of PII and not specific PIIs; thus, it is practical to list this small number of categories of which the proprietor approves disclosure. Such classification of PII is also suitable for software design and implementation in a decision-making system.

In general, the Privacy Rule permits uses and disclosures for "treatment, payment and health care operations," as well as certain other disclosures, without the individual's prior written authorization. What we propose is that such permission be categorized by the system according to the type of PII, for example,

- *Collected* PII uses and disclosures for "treatment, payment and health care operations" has a different access policy than

- *Created* PII uses and disclosures for "treatment, payment and health care operations."

For example, "uses and disclosures" of created PII may need the approval of its creator (e.g., the physician).

Constraint 2: PHI is directly relevant to the person's involvement in payment related to that person's healthcare.

Comment: PII "directly relevant to the person's involvement in payment related to the individual's healthcare" may also be collected PII, processed PII, or created PII. For example, disclosing such information may not include a hospital's own evaluation of the proprietor's financial reliability. It is similar to releasing "information I have" (collected PII), but not "information I have inferred" (created PII).

Constraint 3: The use is to carry out treatment, payment, or healthcare operations.

Comments: The "uses" of collected PII, processed PII, and created PII are different. In particular, created information carries different types of responsibilities from those of collected PII that is then disclosed. The disclosure that (*According to our diagnoses*) *John Smith is an AIDS patient* (collected PII) has a different weight than the disclosure that *John Smith is an ABC employee* (collected PII) even though both disclosures are for the same use.

We conclude that IFM can enhance such a methodology for extracting constraints from regulation texts. IFM simply reveals all paths of information circulation; thus, we can take into account all possible interpretations in the text. Next, we apply IFM to the process called "semantic parameterization," proposed by Breaux and Antón [10].

6. Enhanced Semantic Parameterization

According to Breaux and Antón [10], semantic parameterization is a process to "support engineers who map natural language domain descriptions to models expressed in first-order predicate logic for the purpose of performing automated reasoning and analysis" [16]. It provides:

1) A reference system that systematically detects and resolves such ambiguities.

2) Automated support for placing natural language-like inquiries across collections of requirements that answer who, what, where, when, how, and why questions.

3) A means to formalize and compare different stakeholder viewpoints.

Breaux and Antón [10] consider the following "properties" of information to be access–related activities:

a) The *subject* is the actor who performs an action on an object in the activity. In the IFM, the subject is one of the entities that acts on PII.

b) The *action* is the verb that affects information, such as access, use, disclose, etc. The IFM limits the number of acts on PII that can be applied here.

c) The *modality* modifies the action by denoting the action as a right, obligation, or refrainment. Such a "modality" can be added after the backbone of the information handling processes is designed, as described next.

d) The *object* is limited to information, including the name or date of birth of a patient, or an accounting of disclosures. In IFM, information is limited to PII and includes all acts and uses of information defined by the model in Figure 2.

e) The *target* is the recipient in a transaction, such as the recipient of a disclosure. The target in IFM is one of the agents in PII transactions. There may be several agents, as illustrated in Figure 3. It is also possible that both sides of the transaction belong to the same organization.

In the IFM, we can identify all entities that may be involved in handling PII in order to accomplish the same thing that Breaux and Antón [10] extract from legal wording. IFM gives a complete picture that includes all entities and processes needed to produce a design description of the software, and no more. For example, a policy for access-related activities can be specified according to the type of entity (e.g., a *collector*—a clerk who fills patient data cannot access *processed* information).

Thus, the software engineer can design a system that takes into consideration all cases of information flow. The entities that may be involved in handling PII are several, including proprietor, possessor, collector, processor, different types of users, different types of storekeepers, (data) miner, creator, releaser, and transferor.

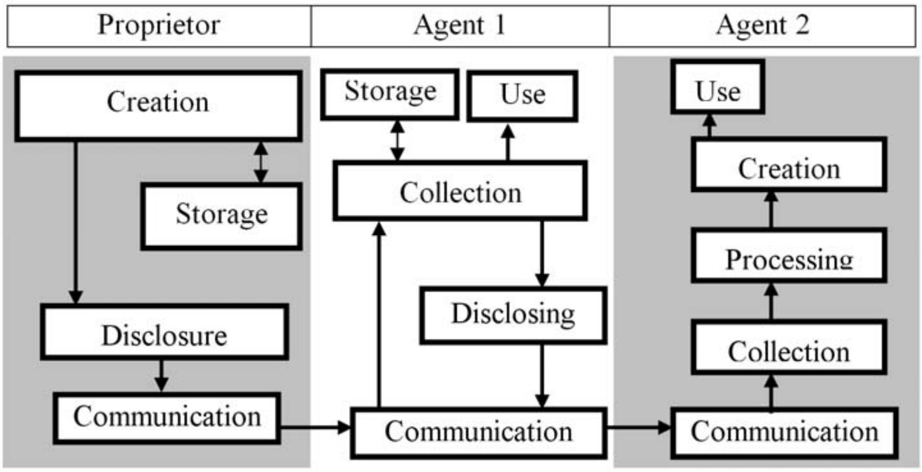


Figure 3. PII flow involving a proprietor and two agents

Consider the following example from Breaux and Antón [10]:

Excerpt from Privacy Rule §164.522(a)(1):

(*i*) *A CE must permit an individual to request that the CE restrict*:

(*A*) *Uses or disclosures of PHI about the individual to carry out treatment, payment or healthcare operations*; *and*

(*B*) *Disclosures permitted under §*164.51*0*(*b*)

(*ii*) *A CE is not required to agree to a restriction.*

(*iii*) *A CE that agrees to a restriction under paragraph* (*a*)(*1*)(*i*) *of this section may not use or disclose PHI in violation of such restriction, except that, if the individual who requested the restriction is in and the restricted PHI is needed to provide emergency treatment, the CE may use the restricted PHI, or may disclose such information to a HCP, to provide such treatment to the individual.*

Applying the Breaux and Antón [10] method to this excerpt yields constraints and rules, some of which are as follows:

- The *use* is to carry out treatment, payment, or healthcare operations.

- The *disclosure* is to carry out treatment, payment, or healthcare operations.
- The CE agrees to a restriction.
- The individual requested the restriction.
- The individual is in need of emergency treatment.
- The PHI is needed to provide emergency treatment.
- A CE must permit an individual to request a restriction.
- A CE is not required to agree to a restriction.

A software engineer then applies *patterns* to identify rights, obligations, and constraints. For example, “the *basic activity pattern* describes a subject who performs an action on an object and *modality* distinguishes the activity as a right, obligation or refrainment. Each rule uses these two patterns to ensure that the statement has precisely one subject, action, object and modality” [10].

The IFM introduces an intermediate stage of analysis that provides a foundation for the legal text. The excerpt from Privacy Rule §164.522(a)(1) specifies relationships between a CE and a proprietor: use and disclosure, as described next.

6.1 The Use Relationship

Figure 4 shows the *Use* relationship embedded in the excerpt. The CE is a possessor of PII about the proprietor. The CE may be a collector, processor, creator, storekeeper, or (data) miner of this PII, or all or some of these roles. In all cases, the relationship between the CE and proprietor leads to some *use* (actions in IFM). The *uses* are of two types:

- Use to “carry out treatment, payment, or healthcare operations”
- Use for “need of emergency treatment” (dotted lines in Figure 4).

The software engineer can represent all possible cases in this situation. Thus, he or she can put constraints and rules on the *use* of PII generated by the CE that are different from constraints and rules on *use* of PII collected from outsiders, including the proprietor him/herself. If PII, e.g., a bank account number, is given by the proprietor, then obviously there are no restrictions on the use of such PII to “carry out payment.” It would make no sense for a patient to give his or her bank account number and then request restrictions on CE use of such information to “carry out payment;” therefore, the software engineer can design a system such that it traces the source of PII, hence focusing the rules accordingly. The IFM magnifies all types of relationships between entities such that ambiguity in the legal text can be easily spotted.

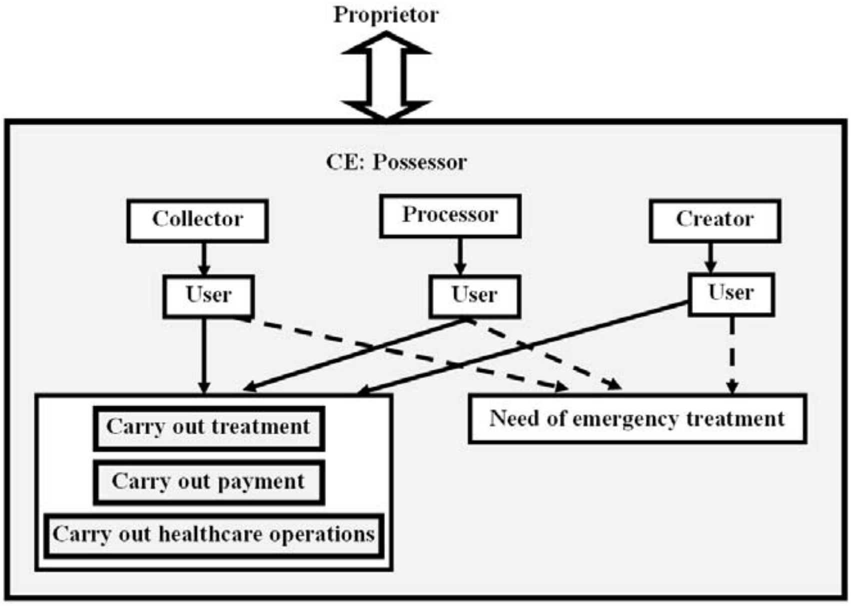


Figure 4. A relationship between a proprietor and CE involving uses of PII

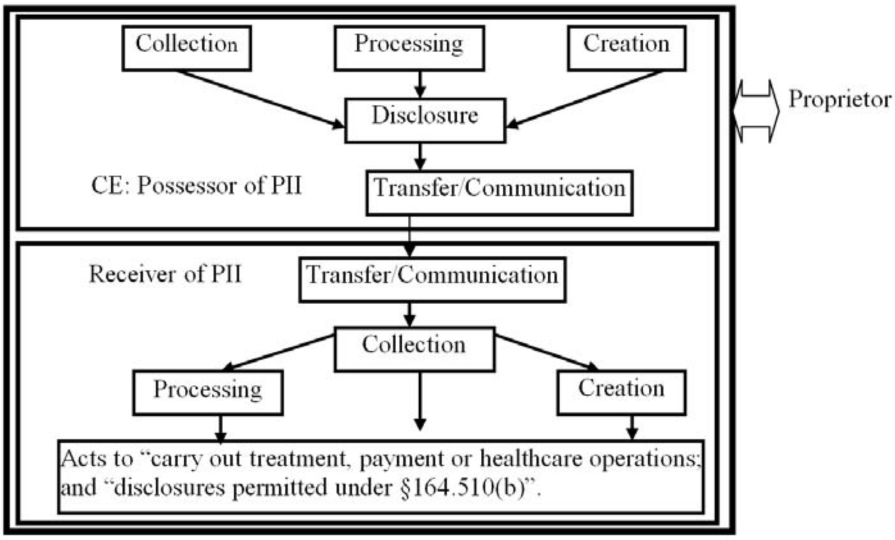


Figure 5. Relationship between proprietor and a situation involving CE disclosure of PII

6.2 The Disclosure Relationship

Now consider the disclosure relationship between the CE and the proprietor embedded in Privacy Rule excerpt §164.522(a)(1) given previously. The first question that comes in mind is, what type of PII is disclosed? Is it collected, processed, or created PII by the CE? Privacy Rule §164.522(a)(1) lumps these types together.

Since the CE is the party who discloses PII, then the user is the party who receives this information from the CE. Thus, the relationship involves three entities: the CE, the proprietor, and the receiver of PII, who uses it to "carry out treatment, payment or healthcare operations" and "disclosures permitted under §164.510(b)." Figure 5 represents this relationship.

Figure 5 depicts a proprietor (wide arrow) in a situation that includes the CE disclosing PII to a collecting agent, who uses it to "carry out treatment, payment or healthcare operations" and "disclosures permitted under §164.510(b)." Figure 6 represents the disclosure condition as described by the Privacy Rule. In comparison with Figure 5, the Privacy Rule is very brief and thus may create ambiguity in disclosure possibilities.

6.3 Assigning Roles to Users

Previous sections demonstrate that the Privacy Rule's

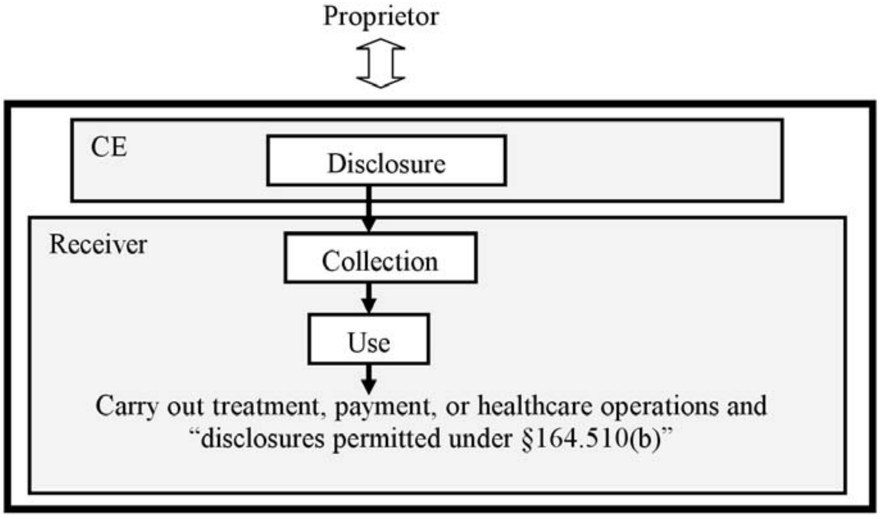

Proprietor
CE
Disclosure
Receiver
Collection
Use
Carry out treatment, payment, or healthcare operations and
"disclosures permitted under §164.510(b)"

Figure 6. The Privacy Rule is very brief in describing the disclosure condition in comparison with the IFM

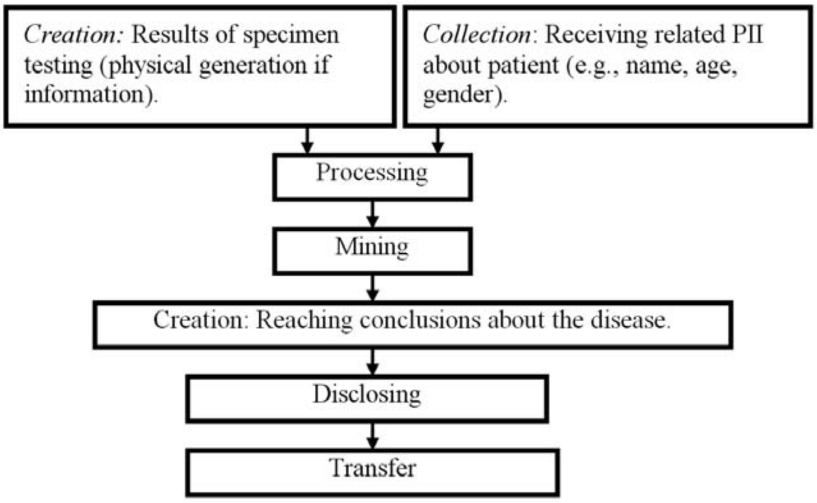

Creation: Results of specimen testing (physical generation if information).
Collection: Receiving related PII about patient (e.g., name, age, gender).
Processing
Mining
Creation: Reaching conclusions about the disease.
Disclosing
Transfer

Figure 7. Laboratory as a user of PII

brevity hides types of uses and disclosures that may require different constraints and access rules. The notion of *purpose* that Breaux and Antón [10] use to indicate the goal of an activity remains to be discussed. In Role-Based Access Control (RBAC) systems [17], stakeholders (users) are permitted or denied access to information based on their *roles*. Roles are assigned to users, whereas purposes (we call them "uses") are assigned to data. In IFM, instead of a mapping between roles and purposes, a sub-graph of IFM is assigned to the users. For example, Figure 7 shows a sub-graph assigned to the laboratory as a user of PII in the hospital system. The hospital system discloses PII to the lab as part of a request for specimen testing. The lab possesses PII from two sources: collected PII from the hospital system, and created PII in the lab itself (the top two boxes in Figure 7). This PII is processed; mined and final conclusions are generated, then disclosed back to the hospital system.

The whole sub-graph (which probably also includes "Storage" sub-stages) represents the role of the lab. The sub-graph replaces the purpose, "resting specimen in laboratory." Access permission/denial of lab workers is decided according to their IFM (sub-graph). It is possible that inside the lab's IFM, several IFMs exist for the superintendent of the lab, lab technician, etc.

In such a system, where the infospheres of different PII possessors and users are very clear, it is possible to implement disclosure and access requirements in the fullest detail within a framework that can be understood even by laymen. The proprietor can request restrictions on disclosers to inside or outside handlers of PII in the possession of the CE. The software engineer can design the PII flow diagram as the network engineer draws the communication network. The drawing can be complemented with all types of requirements at different points of PII flow. Complacence level can be claimed accordingly.

7. Conclusions

In this paper we have proposed complementing current software engineering methods to extract requirements from legal text, with a model that includes all possibilities of information flow. We have shown through examples that such a model provides a foundation for the interpretation process. The IFM is very general, such that it can be applied in all kinds of work in this area. We have concentrated on the analysis of Breaux and Antón [10] in order to reach some depth in illustrating the application of the IFM.

A great deal of work is needed to integrate the IFM into a workable product that generates the software requirement. The most productive approach is to incorporate the model into an existing methodology, rather than developing an IFM-based scheme from scratch. On the other hand, as a future work, theoretical analysis can be developed for the methodology to formally characterize its features.

REFERENCES

[1] D. Reinsel, C. Chute, W. Schlichting, J. McArthur, I. Xheneti, A. Toncheva, and A. Manfrediz, "A for- ecast of worldwide information growth through 2010." An IDC White Paper, 2007. http://www.emc.com/about/destination/digital_universe/pdf/Expanding_Digital_Universe_IDC_WhitePaper_022507.pdf

[2] Nexsan Technologies Inc, White paper on enabling information lifecycle management, 2005. http://www.meganet1.com/pdf/Enabling%20Information%20Lifecycle%20management.pdf

[3] M. J. May, C. A. Gunter, and I. Lee, "Privacy APIs: Access control techniques to analyze and verify legal privacy policies," 19th IEEE Workshop Computer Security Foundations, pp. 85−97, 2006.

[4] T. D. Breaux and A. I. Antón, "Deriving semantic models from privacy policies," 6th IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 67−76, 2005.

[5] S-W. Lee, R. Gandhi, D. Muthurajan, D. Yavagal, and G-J. Ahn, "Building problem domain ontology from security requirements in regulatory documents," International Workshop on Software Engineering for Secure Systems, Shanghai, China, pp. 43−50, 2006.

[6] A. I. Antón, J. B. Earp, Q. He, W. Stufflebeam, D. Bolchini, and C. Jensen, "Financial privacy policies and the need for standardization," IEEE Security and Privacy, Vol. 2, No. 2, pp. 36−45, 2004.

[7] A. I. Antón, "Goal-based requirements analysis," 2nd IEEE International Conference on Requirements Engineering, pp. 136−144, 1996.

[8] T. D. Breaux and A. I. Antón, "Analyzing goal semantics for rights, permissions and obligations," 13th IEEE International Conference on Requirements Engineering, pp. 177−186, 2005.

[9] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, "Modeling security requirements through ownership, permission and delegation," 13th IEEE International Conference on Requirements Engineering, pp. 167−176, 2005.

[10] T. Breaux and A. I. Antón, "Analyzing regulatory rules for privacy and security requirements," IEEE Transactions on Software Engineering, Vol. 34, No. 1, pp. 5−20, January 2008.

[11] D. Tindal, "Safety officer's briefing book," Civil Air Patrol, United States Air Force Auxiliary, February 1 2000. http://www.iawg.cap.gov/archives/ iawgsafety-manual.pdf.

[12] S. Al-Fedaghi, "Scrutinizing the rule: Privacy realization in HIPAA," International Journal of Healthcare Information Systems and Informatics (IJHISI), Vol. 3, No. 2, 2008.

[13] HHS, "Summary of the HIPAA privacy rule," U.S. Department of Health & Human Services, 2003. http://www.hhs.gov/ocr/privacysummary.pdf.

[14] S. Al-Fedaghi, "Software engineering interpretation of information processing regulations", IEEE 32nd Annual International Computer Software and Applications Conference (IEEE COMPSAC 2008), Turku, Finland, July 28–August 1, 2008.

[15] Office for Civil Rights, US Department of Health and Human Services, "Medical privacy: National standards to protect the privacy of personal health information," 2000 http://www.hhs.gov/ocr/hipaa/finalreg.html.

[16] T. D. Breaux and A. I. Antón, "Semantic parameterization: A conceptual modeling process for domain descriptions," North Carolina State University Computer Science Technical Report TR-2006-35, October 2006.

[17] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," IEEE Computer, Vol. 29, No. 2, pp. 38−47, 1996.

J. Software Engineering & Applications, **2009, 2: 77-85**
doi:10.4236/jsea.2009.22012 Published Online July 2009 (www.SciRP.org/journal/jsea)

Refinement in Formal Proof of Equivalence in Morphisms over Strongly Connected Algebraic Automata

Nazir Ahmad Zafar, Ajmal Hussain, Amir Ali

Faculty of Information Technology, University of Central Punjab, 31-A, Main Gulberg, Lahore, Pakistan
Email: {dr.zafar, ajmal, amiralishaihid}@ucp.edu.pk

Received January 13th, 2009; revised February 27th, 2009; accepted March 24th, 2009.

ABSTRACT

Automata theory has played an important role in computer science and engineering particularly modeling behavior of systems since last couple of decades. The algebraic automaton has emerged with several modern applications, for example, optimization of programs, design of model checkers, development of theorem provers because of having properties and structures from algebraic theory of mathematics. Design of a complex system not only requires functionality but it also needs to model its control behavior. Z notation is an ideal one used for describing state space of a system and then defining operations over it. Consequently, an integration of algebraic automata and Z will be an effective computer tool which can be used for modeling of complex systems. In this paper, we have combined algebraic automata and Z notation defining a relationship between fundamentals of these approaches. At first, we have described algebraic automaton and its extended forms. Then homomorphism and its variants over strongly connected automata are specified. Finally, monoid endomorphisms and group automorphisms are formalized, and formal proof of their equivalence is given under certain assumptions. The specification is analyzed and validated using Z/EVES tool.

Keywords: *Formal Methods, Automata, Integration of Approaches, Z Notation, Validation*

1. Introduction

Almost all large, complex and critical systems are being controlled by computer software. When software is used in a complex system, for example, in a safety critical system its failure may cause a huge loss in terms of deaths, injuries or financial damages. Therefore, constructing correct software is as important as its other counterparts, for example, hardware or electromechanical systems [1]. Formal methods are approaches used for specification of properties of software and hardware systems insuring correctness of a system [2]. Using formal methods, we can describe a mathematical model and then it can be analyzed and validated increasing confidence over a system [3]. At the current stage of development in formal approaches, it is not possible to develop a system using a single formal technique and as a result its integration is required with other traditional approaches. That is why integration of approaches has become a well-researched area in computing systems [4,5,6,7,8,9,10]. Further, design of a complex system not only requires capturing functionality but it also needs to model its control behavior. There are a large variety of specification techniques which are suitable for specific aspects in the software development process. For example, algebraic techniques, Z, VDM, and B are usually used for defining data types while process algebra, petri nets and automata are some of the examples which are best suited for capturing dynamic aspects of systems [11]. Because of well-defined mathematical syntax and semantics of the formal techniques, it is required to identify, explore and develop relationships between such approaches for modeling of complete, consistent and correct computerized systems.

Although there exists a lot of work on integration of approaches but there does not exist much work on formalization of graphical based notations. The work [12,13] of Dong *et al.* is close to ours in which they have integrated Object Z and Timed Automata. Another piece of good work is listed in [14,15] in which R. L. Constab- le has given a constructive formalization of some important concepts of automata using Nuprl. A combination of Z with statecharts is established in [9]. A relationship is investigated between Z and Petri-nets in [16,17]. An integration of UML and B is given in [18,19]. Wechler, W. has introduced algebraic structures in fuzzy automata [20]. A treatment of fuzzy automata and fuzzy language theory is given when the set of possible values is a closed interval [0, 1] in [21]. Ito, M., has described formal languages and automata from the algebraic point of view in which he has investigated the algebraic structures of automata and then a kind of global theory is treated [22]. Kaynar, D. K *et al*. has presented a modeling framework of timed computing systems [23]. In [24], Godsil, C. *et al*. has presented some ideas of algebraic graphs with an emphasis on current rather than classical topics of graphs.

Automata theory has proved to be a cornerstone of theoretical computer science since last couple of decades. Compiler constructions, modeling of finite state systems, natural language processing, defining a regular set of finite words are some of the traditional applications of automata. The algebraic automaton is an advanced form of automata having properties and structures from algebraic theory of mathematics. It has emerged with several modern applications, for example, optimal representation and efficient implementation of algorithms, optimization of programs, speech recognition, design of model checkers, image processing and verification of protocols. The applications of algebraic theory of automata are not limited to computers but are being seen in many other disciplines of science and engineering, for example, representing characteristics of natural phenomena in biology [25]. Modeling of chemical systems using cellular automata is another important application area of it [26]. Another interesting application of automata is presented in [27] in which it is described a system for specifying and automatically synthesizing physics-based animation programs based on hybrid automata.

It is obvious that theory of automata has various application areas as discussed above. Because of having some interesting properties from algebra, the algebraic automata has increased its importance in some special application domain of computer science. For example, algebraic automata can be used in static as well as in dynamic part of distributed systems. To understand it, we can suppose objects or entities representing a collection of distributed systems. As in this case, for many applications, there is no precedence of order in computation that means the entities can be concatenated in any order and hence the associative property is satisfied. Further, after identifying the identity element in this collection of objects, the structure produced is called a monoid which is an abstract algebraic structure. It is to be mentioned that the identity element can be identified if it exists based on the nature of the problem. Because of having these special algebraic characteristic, an automaton can be extended to develop algebraic automata which can be used for specification and capturing control behavior over such systems. After understanding the importance of algebraic automata, a relationship is identified and proposed, in this paper, between algebraic automata and Z notation thus facilitating the modeling techniques for complex systems. To achieve the objective of proposed integration, at first, we have given formal description of algebraic automaton and its other extended forms. The strongly connected automaton is described by reusing the structure defining algebraic automata. Then homomorphism, which is an important structure in verifying symmetry of the algebraic structures, and its other variants over strongly connected automata are formalized. Next, monoid endomorphisms and group automorphisms are described. Finally, a formal proof of their equivalence is given under certain assumptions. The specification is analyzed and validated using Z/EVES tool. The major objectives of this paper are: 1) to identify a linkage between automata and Z notation to be useful for modeling the systems and 2) providing a syntactic and semantic relationship between Z and algebraic automata. In Section 2, an introduction to Z notation is given. In Section 3, an overview of algebraic automata is provided. Formal construction of proof showing equivalence of algebraic structures is given in Section 4. Finally, conclusion and future work are discussed in Section 5.

2. An Introduction to Z Notation

Formal methods are approaches, based on the use of mathematical techniques and notations, for describing and analyzing properties of software systems [28]. That is, descriptions of a system are written using notations which are mathematical expressions rather than informal notations. These mathematical notations are based on discrete mathematics such as logic, set theory, graph theory and algebraic structures. There are several ways in which formal methods may be classified. One frequently-made distinction is between property oriented and model oriented methods [29]. Property oriented methods are used to describe the operations which can be performed on a system and then defining relationships between these operations. Property oriented methods usually consist of two parts. The first one is the signature part which is used for defining the syntax of operations and the second one is an equations part used for defining the semantics of the operations by a set of equations called the rules. Algebraic specification of abstract data types [30] and the OBJ language [31] are examples of property oriented methods.

Model oriented methods are used to construct a model of a system's behavior and then allow us to define operations over it [32]. Z notation is one of the most popular specification languages which is a model oriented approach based on set theory and first order predicate logic [33]. It is used for specifying behavior of abstract data types and sequential programs.

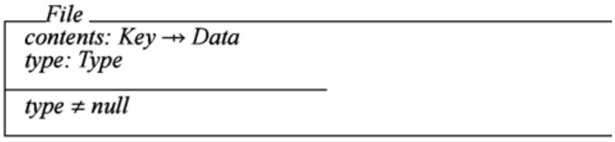
A brief overview of some important structures and operators of Z, used in our research, is given by taking a case from a book on "Using Z: specification, refinement and proof" by Woodcock J. and Davies J., [34]. A programming interface is taken as case study for file systems.

A list of operations which is defined after defining file and an entire system is, 1) read: used to read a piece of data from a file, 2) write: used to write a piece of data to a file, 3) access: may change the availability of a file for reading and writing over the file of the system.

A file is represented as a schema using a relation between storage keys and data elements. For simple specification, basic set types are used. In the formal notation, name, type, keys and data elements of a file are represented as *Name*, *Type*, *Key*, and *Data* respectively in Z notation as given below. An axiomatic definition is used to define a variable null which is used to prove that the type of a file cannot be null even there are no contents on a file.

$[Name, Type, Key, Data]; \mid \quad null: Type$

A file consists of two components, i.e., file *contents* and its type which are specified by contents and type respectively. The schema structure is usually used be-cause of keeping specification both flexible and extensible. In the predicate part, an invariant is described proving that the file type is non null even there are no con-tents in it. As a file can associate a key with at most one piece of a data and hence the relation *contents* is sup-posed to be a partial function.



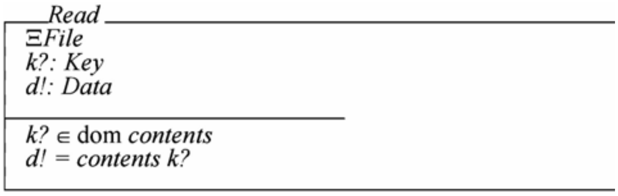
File

$contents: Key \rightarrowtail Data$

$type: Type$

$type \neq null$

The read operation is defined over the file to interrogate the file state. A successful read operation requires an existing key as input and provides the corresponding data as output. The symbol Ξ is used when there is no change in the state of a component. Now the structure Ξ *File* means that the bindings of *File* and *File*' are equal. The decorated file, *File*', represents the next state of the file. Here, it is in fact unchanged because the *k?* is given as input and the output is returned in output variable *d!*. The symbols *?* and *!* are used with input and output variables respectively in the schema given below. In the predicate part of the schema, first it is ensured that the input key *k?* must be in the domain of *contents* which is a partial function. Then the value of data against the given key is returned in the output variable *d!*.



Read

$\Xi File$

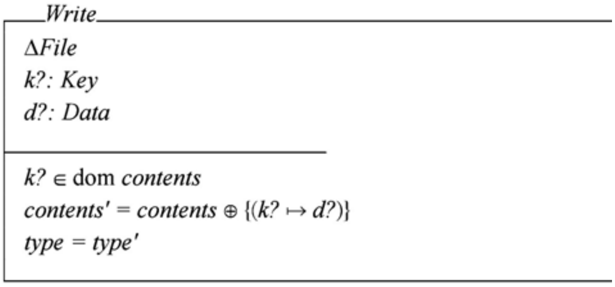
$k?: Key$

$d!: Data$

$k? \in \mathrm{dom}\ contents$

$d! = contents\ k?$

Another operation is defined to write contents over the file. The symbol Δ is used when there is a change in the state of a component (schema). In the schema defined below, the structure Δ *File* gives a relationship between *File* and *File*', representing that the binding of *File* is now changed. The meaning of *File*' is the same as defined above. In this case, the write operation defined below replaces the data stored under an existing key and provides no output. The old value of contents is updated with maplet $k? \mapsto d?$. It is to be noted that file type remained unchanged as defined in the predicate part of the schema. The symbol $\oplus$ is an override operator which is used to replace the previous value of a key with the new one in a given function.



Write

$\Delta File$

$k?: Key$

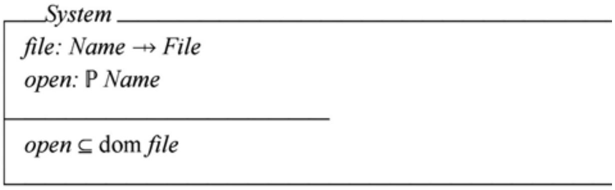
$d?: Data$

$k? \in \mathrm{dom}\ contents$

$contents' = contents \oplus \{(k? \mapsto d?)\}$

$type = type'$

The structure file is reused in description of a file system. As a system may contain a number of files indexed using a set of names and some of which might be open. Hence, the system consists of two components namely collection of files known to the system and set of files that are currently open. The variable *file* is used as a partial function to associate the file name and its contents. The variable *open* is of type of power set of *Name*. The set of files which are open must be a subset of set of total files as described in the predicate part of the schema given below.



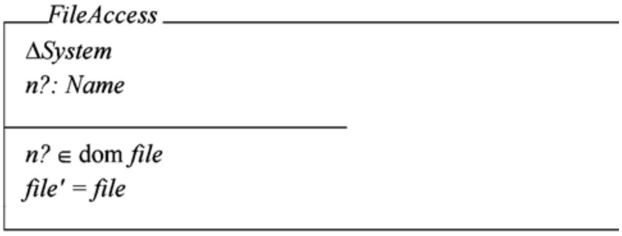
System

$file: Name \rightarrowtail File$

$open: \mathbb{P}\ Name$

$open \subseteq \mathrm{dom}\ file$

As the open and close operations neither change name of any file nor add and remove files from the system. It means both of these are access operations. It may change the availability of a file for reading or writing. The schema described below is used for such operations. The variable *n?* is used to check if the file to be accessed exist in the system. In the schema, it is also described that the file is left unchanged.



FileAccess

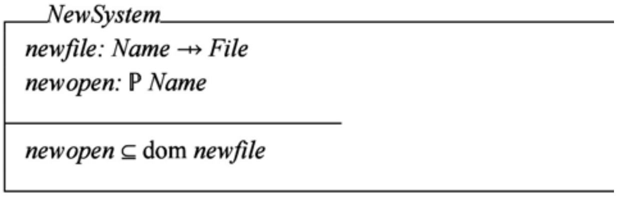
$\Delta System$

$n?: Name$

$n? \in \mathrm{dom}\ file$

$file' = file$

Renaming is another important concept of Z which we have used in our research. For example, if we require creating another system with same pattern but with different components then renaming can be used rather than creating the new system from the scratch. Renaming is sometimes useful because in this way we are able to introduce a different collection of variables with the same pattern. For example, we might wish to introduce variables *newfile* and *newopen* under the constraint of existing system *System*. In this case, the new system *NewSystem* can be created in horizontal form by defining: *NewSystem* A *System*[*newfile*/*file*, *newopen*/*open*] which is equivalent to the schema given below in the vertical form.



NewSystem

newfile: Name ⇸ *File*
newopen: ℙ *Name*

newopen ⊆ dom *newfile*

3. Algebraic Automata

As we know that automata theory has become a basis in the theoretical computer science because of its various applications and playing a vital role in modeling scientific and engineering problems [35]. Modeling control behavior, compiler constructions, modeling of finite state systems are some of the traditional applications of it [36,37,38]. Automata can be classified because of its deterministic and nondeterministic nature. Both types of automata have their own pros and cons in modeling of systems. Both of the automata are equivalent in power because if a language is accepted by one, it is also accepted by the other. Nondeterministic finite automata (NFA) are useful because constructing an NFA is easier than deterministic finite automata (DFA). On the other hand, DFA is useful when implementation is required. Consequently, both of the automata can be used based on the requirements and nature of a problem.

Algebraic automaton which is an abstract form of automata, however, has some properties and structures from algebraic theory of mathematics. The algebraic automata have emerged with several modern applications in computer science. Further, the applications of algebraic theory are not limited to computers but are being seen in other disciplines of science, e.g., representing chemical and physical phenomena in chemistry and biology. It is a well known fact that a given automata may have different implementations and consequently its time and space complexity must be different, which is another issue in modeling using automata. Therefore it is also required to describe the formal specification of automata for its optimal implementation. Further, this relationship will result a useful tool at academic as well as industrial level. A formal verified linkage of algebraic automata and Z is given in the next section.

4. Formal Proof of Equivalence

Now we give formal description of some important concepts of algebraic automata using Z notation. And an equivalence of endomorphisms and automorphisms over strongly connected automata is formalized. The definitions used in this section are based on a book on “Algebraic Theory of Automata and Languages” [22].

4.1 Formalizing Automaton and its Extensions

An algebraic automaton (AA) is a 3-tuple (Q, Σ, δ), where 1) Q is a finite nonempty set of states, 2) Σ is a finite set of alphabets and 3) δ is a transition function which takes a state and an alphabet and produces a state. To formalize AA, Q and Σ are denoted by *S* and *X* respectively.

[Q, *X*]

In modeling using sets in Z, we do not impose any restriction upon number of elements and a high level of abstraction is supposed. Further, we do not insist upon any effective procedure for deciding whether an arbitrary element is a member of the given collection or not. As a consequent, our *Q* and *X* are sets over which we cannot define any operation. For example, cardinality to know the number of elements in a set cannot be defined. Similarly, subset and complement operations over these sets are not defined as well.

To describe a set of states for AA, a variable *states* is introduced. Since, a given state q is of type Q therefore *states* must be of type of power set of Q. For a set of alphabets, the variable *alphabets* is used which is of type of power set of *X*. As we know that δ is a function because for each input (*q1*, *a*), where *q1* is *a* state and a is an alphabet there must be a unique state, which is image of (q1, a) under the transition function δ. Hence we can declare δ as, *delta*: *Q x X* → *Q*.

For a moment, we have used mathematical language of Z which is used to describe various objects. The same language can be used to define the relationships between the objects. This relationship will be used in terms of constraints after composing the objects. The schema structure is used for composition because it is very powerful at abstract level of specification and helps in describing a good specification approach. All of the above components are encapsulated and put in the schema named as *Automaton*. The formal description of it is given below.

Automaton

$states: \mathbb{P}\, Q$

$alphabets: \mathbb{P}\, X$

$delta: Q \times X \rightarrow Q$

$\forall q: Q;\ x: X \mid q \in states \wedge x \in alphabets$

$\bullet\ \exists t: Q \mid t \in states \bullet delta\ (q, x) = t$

Invariants: For each input (s, x) where s is an element of *states* and x is a member of *alphabets*, there is a unique state t such that: *delta* (*s*, *x*) = *t*.

An extended form of algebraic automaton is denoted by *EA*. In the extended form, two more components are added in addition to the three components of the algebraic automaton defined above. In the schema given below, the variables *states* and *alphabets* have the same meaning but the third one *delta* function is refined. In the extended form, the *delta* function takes a states and a string as inputs and produces the same state or new state as output. We also need to compute the set of all the strings based on the set of alphabets and hence a fourth variable is used and denoted by *strings* which is of type of power set of set of all the sequences (strings: P (seq *X*)). As we know that a sequence can be empty and hence a fifth variable is used representing it and is denoted by *epsilon* of type *seq X*.

EA

$states: \mathbb{P}\, Q$

$alphabets: \mathbb{P}\, X$

$strings: \mathbb{P}\, (\mathrm{seq}\ X)$

$delta: Q \times \mathrm{seq}\ X \rightarrow Q$

$epsilon: \mathrm{seq}\ X$

$epsilon \in strings$

$\forall q: Q \mid q \in states \bullet delta\ (q, epsilon) = q$

$\forall q: Q;\ a: X;\ u: \mathrm{seq}\ X \mid q \in states \wedge a \in alphabets \wedge u \in strings$

$\bullet\ delta\ (q, (\langle a \rangle \frown u)) = delta\ ((delta\ (q, \langle a \rangle)), u)$

Invariants: 1) The null string is an element of strings. 2) If transition function takes a state and the null string epsilon, and it produces the same state. 3) For each input where s is an element of states, a is an alphabets and u is an element of strings, delta function is defined as: $delta\ (s, (\langle a \rangle\ ^{\circ}\ u)) = delta\ ((delta(s, \langle a \rangle)), u)$.

Another extended form of algebraic automaton is a strongly connected automaton. A strongly connected is a one for which if for any two given states there exists a string s such that the delta function connects these states through the string s. The strongly connected automaton is represented by SCEA as a schema in Z notation and described as given below. It has the same number of components and properties in addition to one more constraint defined here.

SCEA

$states: \mathbb{P}\, Q$

$alphabets: \mathbb{P}\, X$

$strings: \mathbb{P}\, (\mathrm{seq}\ X)$

$delta: Q \times \mathrm{seq}\ X \rightarrow Q$

$epsilon: \mathrm{seq}\ X$

$epsilon \in strings$

$\forall q: Q \mid q \in states \bullet delta\ (q, epsilon) = q$

$\forall q: Q;\ a: X;\ u: \mathrm{seq}\ X \mid q \in states \wedge a \in alphabets \wedge u \in strings$

$\bullet\ delta\ (q, (\langle a \rangle \frown u)) = delta\ ((delta\ (q, \langle a \rangle)), u)$

$\forall q1, q2: Q \mid q1 \in states \wedge q2 \in states$

$\bullet\ \exists s: \mathrm{seq}\ X \mid s \in strings \bullet delta\ (q1, s) = q2$

Invariants: 1) The null string is an element of *strings*. 2) If the transition function takes a state and null string as input, then it produces the same state. 3) For each $(s, (\langle a \rangle\ ^{\circ}\ u))$, where s is state, a an alphabet and u is a string, the *delta* function is defined as: $delta\ (s, (\langle a \rangle\ ^{\circ}\ u)) = delta\ ((delta(s, \langle a \rangle)), u)$. 4) For any two states q1 and q2, there exists a string s such that: $delta(q1, s) = q2$.

4.2 Homomorphism and its Variants

The word homomorphism means “same shape” and is an interesting concept because a similarity of structures can be verified by it. It is a structure in abstract algebra which preserves a mapping between two algebraic structures, for example, monoid, groups, rings, vector spaces. Now we give formal specification of it and its variants over strongly connected automata. In [22], Ito M. has given a concept of homomorphism and its variants over algebraic automata.

Let SCEA1 = (Q1, $\sum 1$, δ1) and SCEA2 = (Q2, $\sum 2$, δ2) be two strongly connected automata, and let ρ be a mapping from Q1 into Q2. If $\rho\,(\delta 1(q, x)) = \delta 2(\rho(q), x)$ holds for any $q \in Q1$ and $x \in \sum 1$, then ρ is called a homomorphism from Q1 to Q2. The above pair of schemas is represented by $\Delta\, SCEA$ in Z which shows a relationship between SCEA1 and SCEA2. A formal definition of homomorphism from SCEA1 into SCEA2 in terms of a schema is given below. It consists of two components $\Delta\, SCEA$ and *row*. The variable *row* is a mapping from *Q*1 into *Q*2. The sets *Q*1 and *Q*2 are used for states of *SCEA*1 and *SCEA*2 respectively.

Homomorphism

$\Delta SCEA$

$row: Q \rightarrow Q$

$\forall q: Q;\ s: \mathrm{seq}\ X \mid q \in states \wedge s \in strings$

$\bullet\ row\ (delta\ (q, s)) = delta'\ ((row\ q), s)$

Invariants: 1) For every q in set of states and s in set of strings of the first automata, if the mapping *row* satisfies the following condition then it conforms a homomorphism from automata SCEA1 into SCEA2:

$$row\,(deltal(q,s)) = delta2\,((rowq),\,s)\,.$$

If SCEA1 = SCEA1 in the homomorphism then it is called an endomorphism. The mapping *row* is defined from set of states *S* into itself. We have induced formal definition of endomorphism from the definition of homomorphism because it is a special case of it.

```
Endomorphism
ΞSCEA
row: Q → Q
---
∀q: Q; s: seq X | q ∈ states ∧ s ∈ strings
  • row (delta (q, s)) = delta ((row q), s)
```

Invariants: 1) For every state q and string s, if the mapping *row* satisfies the condition: $row\,(deltal(q,s)) = delta2\,((rowq),\,s)$, then it conforms an endomorphism between *SCEA* into itself.

Let us define the bijection over two given sets. Let A and B are two nonempty sets. A mapping π from A to B is called one to one if different elements of A have different images in B. That is ∀ a1, a2∈A; b∈B • π (a1) = b and π(a2) = b ⇒ a1 = a2. The mapping π is called onto if each element of B is an image of some element of A. If a mapping is one to one and onto then it is called bijective. If the mapping defined in case of homomorphism is bijective from algebraic automata SCEA1 to SCEA2 then it is called an isomorphism and the automata are said to be isomorphic. Now we give a formalism of isomorphism from SCEA1 to SCEA2 using the schema given below. For this purpose, we simply define the required constraints of bijection over the homomorphism and it results an isomorphism.

```
Isomorphism
ΞHomomorphism
---
∀q1, q2: Q1; q: Q2 |q1 ∈ states ∧ q2 ∈ states ∧ q ∈ states'
  • (q1, q) ∈ row ∧ (q2, q) ∈ row ⇒ q1 = q2
ran row = states'
```

Invariants: 1) For all q1, q2 in set of states of SCEA1 and q in set of states of SCEA2, if images of q1 and q2 are same under the mapping row then the elements q1 and q2 must be same. 2) Each element of the set of states of automata SCEA2 is an image of some element of automata SCEA1 under the mapping *row*.

If SCEA1 = SCEA2 then an isomorphism becomes an automorphism. Its formal description is given below along with its invariants which are not explained here because it is a repetition as given above in the schema *Isomorphism*.

```
Automorphism
ΞEndomorphism
---
∀q1, q2, q: Q | q1 ∈ states ∧ q2 ∈ states ∧ q ∈ states
  • (q1, q) ∈ row ∧ (q2, q) ∈ row ⇒ q1 = q2
ran row = states
```

4.3 Formalizing Endomorphisms

Let G be a nonempty set. The structure (G,*) under binary operation * is monoid if 1) ∀ x, y∈G, x*y∈G, 2) ∀ x, y, z∈G, (x*y)*z = x*(y*z), that is associative property is satisfied, 3) ∀ x∈G, there exists an e∈G such that x*e = e * x = x, e is an identity of G.

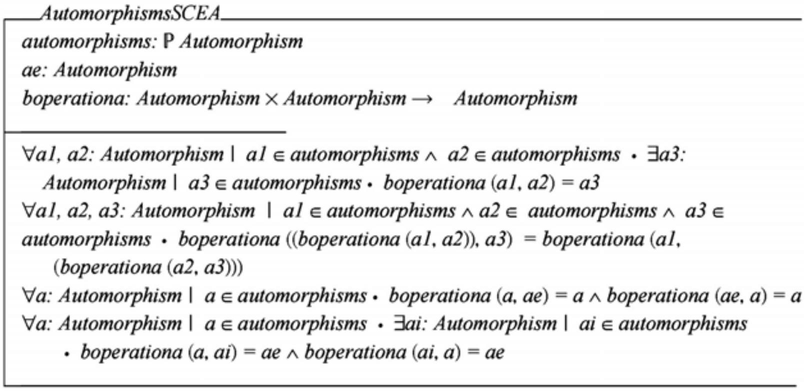
Let A be a automaton and E (A) = a set of all the endomorphisms over an automaton A. It is proved in [21] that E (A) forms a monoid which is an algebraic structure as defined in Section 1. To formalize it, two variables are assumed. The first one is a set of all endomorphisms which is of type of power set of *Endomorphism*. The second one is a binary operation which is denoted by the variable *boperation*. It takes two endomorphisms as input and produces a new endomorphism as an output. The formal definition of the above structure is given below.

Invariants: 1) This property defines binary operation over the set of endomorphisms. 2) Associative is verified here in this property. 3) This property ensures the existence of an identity element in the collection of endomorphisms.

```
EndomorphismsSCEA
endomorphisms: ℙ Endomorphism
boperation: Endomorphism × Endomorphism→ Endomorphism
---
∀e1, e2: Endomorphism | e1 ∈ endomorphisms ∧ e2 ∈ endomorphisms • ∃e3:
Endomorphism | e3 ∈ endomorphisms • boperation (e1, e2) = e3
∀e1, e2, e3: Endomorphism | e1 ∈ endomorphisms ∧ e2 ∈ endomorphisms ∧ e3 ∈
endomorphisms • boperation ((boperation (e1, e2)), e3) = boperation (e1,(boperation (e2, e3)))
∀e: Endomorphism | e ∈ endomorphisms • ∃ee: Endomorphism | ee ∈ endomorphisms
      • boperation (e, ee) = e ∧ boperation (ee, e) = e
```

4.4 Formalizing Automorphisms

The algebraic structure (G, *) is called a group if 1) it is a monoid and 2) for any element x in the set G, there exists an element y in G such that x*y = y*x = e. That is the inverse of each element of G exists. Now let us suppose that G (A) = set of all the automorphisms over the algebraic automata A. For its formal specification, three variables are assumed. The first one is a set of all automorphism which is of type of power set of *Automorphism*. The second one is an identity element. And the last one is binary operation denoted by *boperation*. It takes two automorphisms as input and produces a new automorphism as an output.

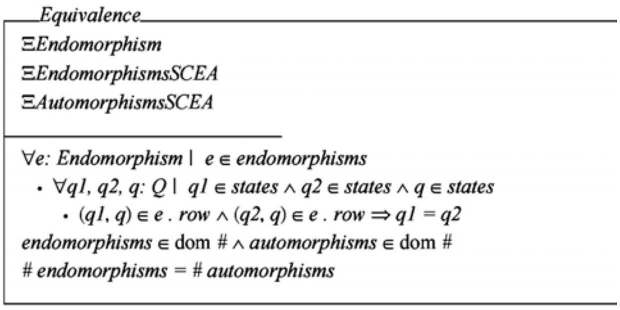


```
AutomorphismsSCEA
automorphisms: ℙ Automorphism
ae: Automorphism
boperationa: Automorphism × Automorphism →  Automorphism
----------
∀a1, a2: Automorphism | a1 ∈ automorphisms ∧ a2 ∈ automorphisms • ∃a3:
  Automorphism | a3 ∈ automorphisms • boperationa (a1, a2) = a3
∀a1, a2, a3: Automorphism | a1 ∈ automorphisms ∧ a2 ∈ automorphisms ∧ a3 ∈
automorphisms • boperationa ((boperationa (a1, a2)), a3) = boperationa (a1,
    (boperationa (a2, a3)))
∀a: Automorphism | a ∈ automorphisms • boperationa (a, ae) = a ∧ boperationa (ae, a) = a
∀a: Automorphism | a ∈ automorphisms • ∃ai: Automorphism | ai ∈ automorphisms
    • boperationa (a, ai) = ae ∧ boperationa (ai, a) = ae
```

Invariant: The last one property verifies existence of inverse of each element in set G (A). The first three properties are same as in case of endomorphisms.

4.5 Proof of Equivalence

In this section, a formal proof of equivalence of endomorphisms and automorphisms is described. That is we have to verify that the set of all endomorphisms and set of all automorphisms over strongly connected algebraic automata are same. This verification is done in terms of a schema named as *Equivalence*. There are three inputs to this schema which are *Endomorphism*, *EndomorphismsSCEA* and *AutomorphismsSCEA*. At first, it is described that set of all endomorphisms are bijective over the strongly connected automata. Then it is verified that the number of elements must be same in both of the endomorphisms and automorphisms.



```
Equivalence
ΞEndomorphism
ΞEndomorphismsSCEA
ΞAutomorphismsSCEA
----------
∀e: Endomorphism | e ∈ endomorphisms
  • ∀q1, q2, q: Q | q1 ∈ states ∧ q2 ∈ states ∧ q ∈ states
    • (q1, q) ∈ e . row ∧ (q2, q) ∈ e . row ⇒ q1 = q2
endomorphisms ∈ dom # ∧ automorphisms ∈ dom #
# endomorphisms = # automorphisms
```

5. Conclusions

The main objective of this paper was proposing an integration of some fundamental concepts in strongly connected automata and Z notation. To achieve this objective, first we described formal specification of strongly connected automaton. Then a relationship was identified and proposed between automata and Z structures. Next we described two important concepts of homomorphism and isomorphism between algebraic structures. Extended forms over strongly connected automata were formalized for these structures. Finally, a formal proof of equivalence between endomorphisms and automorphisms over strongly connected automata was described. It is to be mentioned here that preliminary results of this research were presented in [39].

Why and what kind of integration is required, were two basic questions in our mind before initiating this research. Strongly connected algebraic automaton is best suited for modeling behavior while Z is an ideal one used describing state space of a system. This distinct in nature but supporting behavior of Z forces us to integrate it with strongly connected algebraic automata.

An extensive survey of existing work was done before initiating this research. Some interesting work [40,41,42] in addition to given in Section 2 was found but our work and approach are different because of abstract and conceptual level integration of Z and automata. We believe that this work will be useful in development of integrated tools increasing their modeling power. It is to be mentioned that most of the researchers discussed here have either taken some examples in defining integration of approaches or addressed only some aspects of these approaches. Further, there is a lack of formal analysis which can be supported by computer tools. Our work is different from others because we have given a generic approach to link Z and algebraic automata with a computer tool support.

Our idea is original and important because we have observed after integrating that a natural relationship exists there. This work is important because formalizing graph based notation is not easy as there has been little tradition of formalization in it due to concreteness of graphs [43]. Our work is useful for researchers interested in integration of approaches. We believe that this research is also useful because it is focused on general principles and concepts and this integration can be used for modeling systems after required reduction. Formalization of some other concepts in algebraic automata is under progress and will appear soon.

REFERENCES

[1] Hall, "Correctness by construction: Integrating formality into a commercial development process," LNCS, Springer, Vol. 2391, pp. 139−157, 2002.

[2] J. Burgess, "The role of formal methods in software engineering education and industry," Technical Report: CS-EXT-1995-045, University of Bristol, UK, 1995.

[3] A. L. Gwandu and D. J. Creasey, "The importance of formal specification in the design of hardware systems," School of Electron & Electrical Engineering, Birmingham University, UK, 1994.

[4] H. A. Gabbar, "Fundamentals of formal methods," Modern Formal Methods and Applications, Springer, 2006.

[5] A. Boiten, *et al.*, "Integrated formal methods (IFM04)," Springer, 2004.

[6] J. Davies and J. Gibbons, "Integrated formal methods (IFM07)," UK, Springer, 2007.

[7] J. Romijn, G. Smith, and J. V. D. Pol, "Integrated formal methods (IFM 05)," Springer, 2005.

[8] K. Araki, A. Galloway, and K. Taguchi, "Integrated formal methods (IFM99)," Springer, 1999.

[9] R. Bussow and W. Grieskamp, "A modular framework for the integration of heterogeneous notations and tools," Proceedings of the 1st International Conference on Integrated Formal Methods, UK, Springer, pp. 211−230, 1999.

[10] W. Grieskamp, T. Santen, and B. Stoddart, "Integrated formal methods (IFM 2000)," Germany, Springer, 2000.

[11] T. B. Raymond, "Integrating formal methods by unifying abstractions," LNCS, Springer, Vol. 2999, pp. 441−460, 2004.

[12] J. S. Dong, R. Duke, and P. Hao, "Integrating object-Z with timed automata," in Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems, pp. 488−497, 2005.

[13] S. Dong, *et al.*, "Timed patterns: TCOZ to timed automata," in the 6th IEEE International Conference on Formal Engineering Methods, USA, 2004.

[14] R. L. Constable, *et al.*, "Formalizing automata II: Decidable properties," Cornell University, 1997.

[15] R. L. Constable, *et al.*, "Constructively formalizing automata theory," Foundations of Computing Series, MIT Press, 2000.

[16] X. He, "Pz nets a formal method integrating Petri nets with Z," Information & Software Technology, Vol. 43, No.1, pp. 1–18, 2001.

[17] M. Heiner and M. Heisel, "Modeling safety critical systems with Z and Petri nets," International Conference on Computer Safety, Reliability and Security, LNCS, Springer, Vol. 1698, pp. 361–374, 1999.

[18] H. Leading and J. Souquieres, "Integration of UML and B specification techniques: Systematic transformation from OCL expressions into B," in the Proceedings of Asia-Pacific Software Engineering Conference (APSEC 02), Australia, 2002.

[19] H. Leading and J. Souquieres, "Integration of UML views using B notation," in the Proceedings of Workshop on Integration and Transformation of UML Models, Spain, 2002.

[20] W. Wechler, "The concept of fuzziness in automata and language theory," Akademic-Verlag, Berlin, 1978.

[21] N. M. John and S. M. Davender, "Fuzzy automata and languages: Theory and applications," Chapman & HALL, 2002.

[22] M. Ito, "Algebraic theory of automata and languages," World Scientific Publishing, 2004.

[23] K. Kaynar and N. Lynchn, "The theory of timed I/O automata," Morgan & Claypool Publishers, 2006.

[24] C. Godsil and G. Royle, "Algebraic graph theory," Springer, 2001.

[25] Z. Aleksic, "From biology to computation," IOS Press Publishers, 1993.

[26] L. B. Kier, P. G. Seybold, and C. K. Cheng, "Modeling chemical systems using cellular automata," Springer, 2005.

[27] T. Ellman, "Specification and synthesis of hybrid automata for physics-based animation," Automated Software Engineering, Vol. 13, No. 3, pp. 395–418, 2006.

[28] D. Conrad and B. Hotzer, "Selective integration of formal methods in development of electronic control units," in Proceedings of 2nd International Conference on Formal Engineering Methods, Vol. 9, No. 11, pp.144–155, 1998.

[29] M. Brendan and J. S. Dong, "Blending object-Z and timed CSP: An introduction to TCOZ," in Proceedings of the 1998 International Conference on Software Engineering, Vol. 19, No. 25, pp. 95–104, 1998.

[30] J. V. Guttag and J. J. Horning, "The algebraic specification of abstract data types," Acta Informatica, Springer Berlin, pp. 27–52, 2004.

[31] A. T. Nakagawa, *et al.*, "Cafe an industrial-strength algebraic formal method," Elsevier Science & Technology, 2000.

[32] J. M. Spivey, “The Z notation: A reference manual,” Prentice Hall, 1989.

[33] J. M. Wing, “A specifier: Introduction to formal methods,” IEEE Computer, Vol. 23, No. 9, pp. 8–24, 1990.

[34] J. Woodcock and J. Davies, “Using Z: Specification, refinement and proof,” Prentice Hall International, 1996.

[35] J. A. Anderson, “Automata theory with modern applications,” Cambridge University Press, 2006.

[36] L. L. Claudio, *et al.*, “Applications of finite automata representing large vocabularies,” Software Practice & Experience, Vol. 23, No. 1, pp. 15–30, 1993.

[37] Y. V. Moshe, “Nontraditional applications of automata theory,” Theoretical Aspects of Computer Software, 1994.

[38] D. I. A. Cohen, “Introduction to computer theory,” 2nd Edition, John Wiley & Sons Inc., 1996.

[39] N. A. Zafar, A. Hussain, and A. Ali, “Formal proof of equivalence in endomorphisms and automorphisms over strongly connected automata,” International Conference on Computer Science and Software Engineering (CSSE 2008), Wuhan, China, 2008.

[40] D. P. Tuan, “Computing with words in formal methods,” University of Canberra, Australia, 2000.

[41] J. P. Bowen, “Formal specification and documentation using Z: A case study approach,” International Thomson Computer Press, 1996.

[42] S. A. Vilkomir and J. P. Bowen, “Formalization of software testing criterion,” South Bank University, London, 2001.

[43] C. T. Chou, “A formal theory of undirected graphs in higher order logic,” in Proceedings of 7th International Workshop on Higher Order Logic, Theorem Proving and Application, LNCS, Vol. 859, pp. 144–157, 1994.

***J. Software Engineering & Applications*, 2009, 2: 86-95**
doi:10.4236/jsea.2009.22013 Published Online July 2009 (www.SciRP.org/journal/jsea)

Lossy-to-Lossless Compression of Hyperspectral Image Using the 3D Set Partitioned Embedded ZeroBlock Coding Algorithm

Ying Hou

School of Communication and Information Engineering, Xi'an University of Science and Technology, Xi'an, China.
Email: houying@mailst.xjtu.edu.cn

Received December 28th, 2009; revised March 23rd, 2009; accepted April 22nd, 2009.

ABSTRACT

In this paper, we propose a three-dimensional Set Partitioned Embedded ZeroBlock Coding (3D SPEZBC) lossy-to-lossless compression algorithm for hyperspectral image which is an improved three-dimensional Embedded ZeroBlock Coding (3D EZBC) algorithm. The algorithm adopts the 3D integer wavelet packet transform proposed by Xiong et al. to decorrelate, the set-based partitioning zeroblock coding to process bitplane coding and the context-based adaptive arithmetic coding for further entropy coding. The theoretical analysis and experimental results demonstrate that 3D SPEZBC not only provides the same excellent compression performances as 3D EZBC, but also reduces the memory requirement compared with 3D EZBC. For achieving good coding performance, the diverse wavelet filters and unitary scaling factors are compared and evaluated, and the best choices were given. In comparison with several state-of-the-art wavelet coding algorithms, the proposed algorithm provides better compression performance and unsupervised classification accuracy.

Keywords: *Image Compression, Hyperspectral Image, 3D Wavelet Packet Transforms, Zeroblock Coding*

1. Introduction

Hyperspectral images provide high resolution and valuable spectrum information about the Earth's surface, so they are a useful tool and extensively applied in military and civilian fields. However, due to the huge amounts of data that bring about some problems in data transmission, storage and processing, more efficient compression technique becomes an indispensable task and a hot research topic.

In recent years, some hyperspectral image compression algorithms based on three-dimensional wavelet transform [1,2,3,4,5] are particularly interested thanks to their excellent compression performances and many attractive properties, such as the three-dimensional Set Partitioning in Hierarchical Trees (3D SPIHT) [4,5], the three-dimensional Set Partitioned Embedded bloCK (3D SPECK) [1], and the JPEG2000 multi-component (JPEG 2000-MC) [2,3]. The researches on hyperspectral image compression schemes can be generally classified into lossless and lossy techniques [1]. Lossless compression can exactly reconstruct the original images without losing any information. The state-of-the-art lossless compression methods are able to achieve compression ratios of 2 ~ 3.4 : 1, which is not enough to meet the actual compression requirements. Lossy compression can achieve higher compression ratio by discarding some information. Nevertheless, thanks to the extraordinary expense to collect hyperspectral images, sometimes we would not like to lose important data information that may affect the later applications. The lossy-to-lossless compression scheme combines the characteristics of two above-mentioned compression techniques and gives the option of the reconstructed image quality (lossy or lossless coding) according to the practical demands. The lossy compression results are obtained when the decoder truncates the lossless encoded bit stream at a desired rate. If the hyperspectral image is decoded without losing any information, it can be perfectly reconstructed. Recently, the researches in the lossy-to-lossless compression for hyperspectral images have been proposed. Tang and Pearlman [6] proposed a lossy-to-lossless compression solution to support random ROI access for hyperspectral image using the 3D SPECK algorithm. Wu *et al.* [7] present an asymmetric transform 3D SPECK (AT-3D SPECK) algorithm for hyperspectral image lossy-to-lossless compression. In Reference [8], Penna *et al.* propose a unified embedded lossy-to-lossless compression framework based on the JPEG 2000 standard. Zhang, Fowler and Liu [9] present a lossy-to-lossless hyperspectral image compression algorithm by using three-dimensional tarp-based coding with classification for embedding (3D TCE) and integer Karhunen-Loève transform (KLT).

The motion-compensated Embedded ZeroBlock Coding (MC-EZBC) [10] coder proposed by Hsiang and Woods is a successful scalable video compression algorithm and provides higher compression efficiency, lower computational complexity and some attractive features such as quality, resolution and temporal scalability. Hyperspectral image has higher correlation and not motion along spectral direction [1]. Thus, the 3D EZBC algorithm without motion compensation can achieve better coding performance for hyperspectral image compression. Whereas, because it needs to establish a quadtree representation structure for each individual 2D subband before starting the bitplane coding, the amount of memory required for quadtree structure is prominent and disadvantageous for the hyperspectral images compression. For a hyperspectral image with size $512\times 512\times 224$, the memory space of quadtree representation structure needs about 299.52 Mbytes. So, Hou and Liu take into account the characteristics of hyperspectral image, the excellent performance of the 3D EZBC algorithm, as well as the attractive properties of low memory requirements and fast encoding/decoding of the 2D SPECK algorithm [11], and then propose a three-dimensional Set Partitioned Embedded ZeroBlock Coding (3D SPEZBC) algorithm which is more suitable for hyperspectral image compression [12]. Instead of the partitioning coding method based on the quadtree representation structure in 3D EZBC, this algorithm adopts the partitioning coding method based on the set representation structure in 2D SPECK to process each individual 2D subband, so it can save higher memory requirements against 3D EZBC because the quadtree structure can be eliminated. For $512\times 512\times 224$ hyperspectral image, 75.52 Mbytes memory space is economized against 3D EZBC.

In this paper, we present a hyperspectral image lossy-to-lossless compression method based on the 3D SPEZBC algorithm, which adopts the Xiong's 3D integer wavelet packet transform (3D integer WPT) to decorrelate, the set-based quadtree partitioning zeroblock technique to process bitplane coding and the context-based adaptive arithmetic coding for further entropy coding. According to the extensive experiments and theoretical analyses, 3D SPEZBC provides the same excellent compression performances compared with 3D EZBC, saves the considerable memory requirement against 3D EZBC and exhibits the speed performance that is slightly worse than 3D EZBC. Furthermore, for achieving good coding per formance, we also evaluate different wavelet filters and unitary scaling factors based on the 3D integer WPT structure, and make the best choices. Compared with several state-of-the-art wavelet-based coding algorithms, the experimental results demonstrate that our algorithm can provide excellent compression performance and unsupervised classification accuracy. So the 3D SPEZBC algorithm is a good candidate for hyperspectral images lossy-to-lossless compression.

The remainder of this paper is organized as follows: Section 2 presents an overview of wavelet transform and Xiong's 3D integer wavelet packet decomposition structures with unitary scaling. In Section 3, the 3D SPEZBC algorithm for hyperspectral image lossy-to-lossless compression is described in detail. Furthermore, the discussions on the coding characteristics and the comparison between the 3D SPEZBC and 3D EZBC algorithm are given in Section 4. Section 5 provides the comprehensive experimental results for hyperspectral image compression. Finally conclusion is drawn in Section 6.

2. Three-Dimensional Wavelet Transform

The lifting scheme presented by Sweldens is the second generation wavelet transform and provides many attractive advantages. To realize lossy-to-lossless image compression based on wavelet transform, the integer-based lifting scheme [13] is an indispensable tool. It performs the reversible integer-to-integer wavelet transform by rounding and truncating each filter output. Many integer-based lifting wavelet transforms are proposed [14,15]. In this paper, we evaluate and compare the lossy-to- lossless compression performances by using some integer wavelet transforms, such as S+P(B), (2+2, 2), 5/3, etc.

Hyperspectral images can be viewed as 3D data and the image coding performance using 3D wavelet transform (WT) obviously outperforms those using 2D WT in most cases. However, there are diverse 3D WT structures [1,2,3] according to different decomposition order in the spatial-horizontal, spatial-vertical, and spectral-slice directions, namely 3D dyadic wavelet transform (DWT), 3D wavelet packet transform (WPT) and Xiong's 3D integer WPT. In recent years, researches have proven that the statistics of hyperspectral image are not symmetric along three dimensions and that higher correlation is exhibited in the spectral direction [1]. 3D WPT allows different decomposition levels in the spatial and spectral dimensions, and further performs spatial decomposition even in the higher-frequency spectral subbands. So it can achieve more flexible decomposition structure and preferable energy convergence in the space-frequency domain. Moreover, 3D integer WPT proposed by Xiong *et al.* [15] is capable of efficiently utilizing the statistical properties to decorrelate and gaining better compression performance for lossy-to-lossless coding. In our lossy-to-lossless compression coder, Xiong's 3D integer WPT is used to decorrelate hyperspectral images.

2.1 Xiong's 3D Integer WPT Structure

As shown in Figure 1, Xiong's 3D integer WPT first carries out an $L_{spectral}$ levels 1D WPT in the spectral-slice direction, which needs to further decompose the high-frequency component at even decomposition level, and then applies an $L_{spectral}$ levels 2D DWT to each resulting spatial image. If we adopt Xiong's 3D integer WPT of $L_{spectral}$ spatial levels and $L_{spectral}$ spectral levels for the hyperspectral image with F spectral bands, $K = (L_{spatial} \times 3 + 1) \times F$ individual 2D subbands can be generated. For example, Figure 1 shows the Xiong's 3D integer WPT structure with 56 2D subbands, as performing two spatial levels and two spectral levels for a volumetric image with 8 spectral bands.

2.2 Unitary Scaling Factor for 3D Integer Wavelet Transform

Because the integer-based lifting wavelet transform is not unitary, it badly compromises integer-based lossy coding performance [15]. To obtain better lossy coding performance, some researchers have presented a simple approach via bit shifting of wavelet coefficients to make the integer WT approximately unitary. In Reference 1, Tang *et al.* adopt Xiong's 3D integer WPT with unitary scaling [15] for hyperspectral image lossy-to-lossless compression. Nevertheless, thanks to fractional scaling factors, bits will be lost for right shift on the highest-frequency subbands, so all factors must be multiplied by the correctional times in order to make them be the nonnegative powers of 2. The correctional times used by Tang equal to four [1]. Through our experimental evaluations and analysis on the compression performances of several 3D integer WT structures with unitary scaling factor, we found that the Tang's unitary scaling structure with the correctional times can achieve effective integer-based lossy coding performances, but its lossless compression performance is degraded. Furthermore, unitary scaling structure adopted by Wu *et al.* [7] can obtain slightly better lossless performance than Tang's unitary scaling structure, but its integer-based lossy coding performance is worse than that of Tang's method. So we adopt a unitary scaling structure as in Figure 2's for hyperspectral image compression, which can obtain not only better lossless performance, but also excellent integer-based lossy performance.

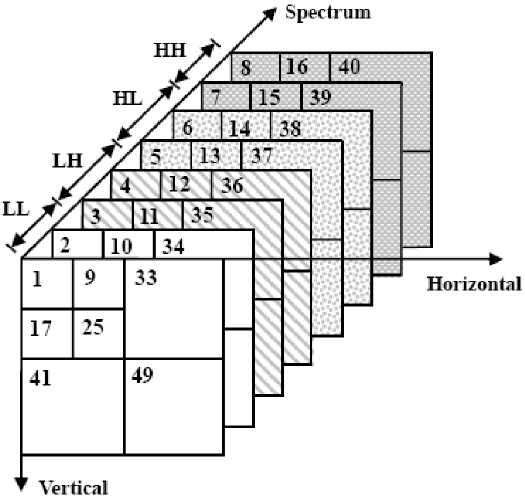
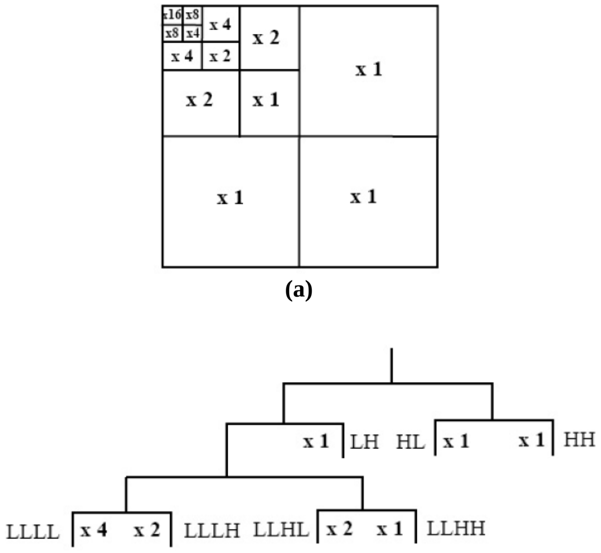


Figure 1. Xiong's 3D integer wavelet packet transform structures of two spectral levels and two spatial levels. The numbers on the front upper left corner of all subbands indicate the list initialization order of the 3D SPEZBC algorithm

3. The 3D SPEZBC Algorithm for Hyperspectral Image Coding

The 3D EZBC algorithm needs to establish a quadtree representation structure with the hierarchical pyramidal model for each individual 2D subband before starting the bitplane coding. This structure provides a fast quadtree splitting scheme, but its price paid needs much memory [10]. Especially, the memory cost is prominent and disadvantageous in the huge volumetric images compression, such as hyperspectral images, 3D medical images, etc. 3D SPEZBC is an embedded zeroblock bitplane coding algorithm by efficiently utilizing the energy clustering nature within subbands and the strong dependency across subbands. It adopts the set-based quardtree partitioning zeroblock coding and the context-based adaptive arithmetic



(a)

(b)

Figure 2. The unitary scaling factors after Xiong's 3D integer WPT of four spatial levels and four spectral levels. (a) The spatial scaling factors. (b) The spectral scaling factors

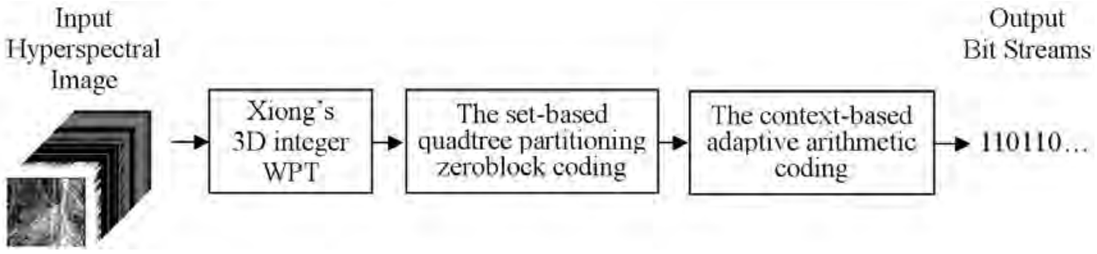

Input
Hyperspectral
Image
Xiong's 3D integer WPT
The set-based quadtree partitioning zeroblock coding
The context-based adaptive arithmetic coding
Output
Bit Streams
110110...

Figure 3. Block diagram of the hyperspectral image lossy-to-lossless coding system based on 3D SPEZBC

coding techniques. The block diagram of the hyperspectral image lossy-to-lossless compression coder based on 3D SPEZBC is illustrated in Figure 3 and the complete coding procedure is summarized as the following three steps.

1) Firstly, for the hyperspectral image, a hierarchical pyramidal structure is obtained by Xiong's 3D integer WPT with Figure 2's unitary scaling factor. In this structure, many rectangular 2D subbands with different sizes are generated, and each individual 2D subband is treated as a code block and defined as an initialization set. Whereafter, the code blocks are split and the significant coefficients are located via the set-based quadtree partitioning zeroblock coding technique.

2) Before starting the coding process, we define a set $S_{k,b}[l]$ to represent the code block of size $2^l \times 2^l$ at the spectral band *b*, subband *k* and set partitioning level *l*, as shown in Table 1. Where the set partitioning level *l* denotes the splitting depth from current code set to pixel-level sets (namely single pixel). For a set of size $M \times M$, it is defined by

$$l \;=\; \log_2 M$$

Substantively, *l* plays the same role as the quadtree level in 3D EZBC. Moreover, L_k denotes the set partitioning level of initialization set (namely k^{th} 2D subband), where *k* is the subband index ordeir (*k* = 0, 1, …, *K*-1) and *K* is the total number of the 2D subbands in wavelet decomposition image. At the same time, we define $L_{\max}$ to be the maximum set partitioning level among all initialization subbands. Table 1 lists the relationship among the code block, set and set partitioning level.

The 3D SPEZBC algorithm adopts the same list strategy used in 3D EZBC, and maintains two arrays of lists:

◆ *LIS*: List of Insignificant Sets.

◆ *LSP*: List of Significant Pixels.

In order to effectively use the statistical characteristics within individual subbands and set partitioning levels, some lists are separately established, namely $LIS_k[l]$ (*LIS* of the subband *k* and set partitioning level *l*) and LSP_k (*LSP* of the subband *k*), where *k* = 0, 1, …, *K*-1 and *l* = 0, 1, …, $L_{\max}$. Initially, k^{th} 2D subband at spectral band *b* is treated as a $S_{k,b}[L_k]$ set and added into $LIS_k[L_k]$ list according to the subband index order *k* of the marked numbers in Figure 1. In the coding procedure, the sets are successively added into corresponding $LIS_k[l]$ or LSP_k list in terms of their significance status.

3D SPEZBC adopts the set-based partitioning bitplane coding to progressively encode the wavelet coefficients of each subband from the Most Significant Bit (MSB) plane toward the Least Significant Bit (LSB) plane. In every bitplane pass, all sets in LIS_k (*LIS* of the subband *k*) list are tested and coded from the bottom set partitioning level (*l* = 0 level) to the maximum set partitioning level (*l* = $L_{\max}$ level). Therefore, the sets of size 1×1 (single pixels) are coded first, and the sets of size 2×2 are coded next, and so on. If the set $S_{k,b}[l]$ contains the significant coefficients, it is tested significant against the current threshold. So set $S_{k,b}[l]$ of size $2^l \times 2^l$ at the set partitioning level *l* is partitioned into four approximately equal subsets $O(S_{k,b}[l]) = \{S^0_{k,b}[l-1],\; S^1_{k,b}[l-1],\; S^2_{k,b}[l-1],\; S^3_{k,b}[l-1]\}$ of size $2^{l-1}\times 2^{l-1}$ at the set partitioning level *l*-1. Subsequently, each subset is treated as a new set, and in turn these new sets $O(S_{k,b}[l])$ are further tested and processed in the same way above, as shown in Figure 4(b). Whole partitioning process is recursively executed until the pixel-level sets are reached, so that all significant pixels in subband are located and then added into LSP_k list for further refinement coding. The 3D SPEZBC coding algorithm is described later in detail.

3) Finally, in order to further improve the coding performance, 3D SPEZBC makes use of the context-based adaptive arithmetic coding approach in 3D EZBC to encode the significance map, signs and refinement bitstreams. Although our algorithm gets rid of the quadtree structure, it can also make use of the set-based partitioning process to build upon the similar context models with hierarchical pyramidal structure as 3D EZBC. Nevertheless, unlike the 3D EZBC context models which are built for the quadtree nodes from different subbands and quadtree levels [10], 3D SPEZBC build the independent context models for all sets within individual subbands and set partitioning levels. And it effectively employs two statistical dependencies — the intra-band correlation among sets at the same set partitioning level within subband and the inter-band correlation among set across subbands. For entropy coding of significance testing bitstreams, the 3D SPEZBC context models registers the significance testing status of each set, and the context of every set $S_{k,b}[l]$ is located as node of the

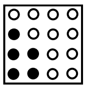
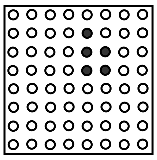
pyramidal context models at the spectral band *b*, subband *k* and set partitioning level *l*, as illustrated in Figure 4(c). Moreover, the sign coding employs the similar scheme of JPEG 2000, namely the output sign bitstream of the significant coefficient is coded according to its sign and significance status of its eight neighboring pixels. Finally, the entropy coding of refinement bitstreams utilizes the same context models of the significance map coding. The details about the context models and look-up tables can refer to Reference 10.

The significance testing function of the set $S_{k,b}[l]$ against a certain threshold 2^n is defined as follows:

$$\Gamma_n(S_{k,b}[l]) = \begin{cases} 1, & \text{if } 2^n \le \max\limits_{(i,j,b)\in S_{k,b}[l]} |c(i,j,b)| < 2^{n+1} \\ 0, & \text{else} \end{cases}$$

where $c(i, j, b)$ denotes the transformed wavelet coefficient at the coordinate (i, j, b)

Table 1. The relationship of code block, set and set partitioning level

Code block	Set	Set size	Set partitionin level l
	$S_{k,b}[0]$	1 x 1	0
	$S_{k,b}[1]$	2 x 2	1
	$S_{k,b}[2]$	4 x 4	2
	$S_{k,b}[3]$	8 x 8	3

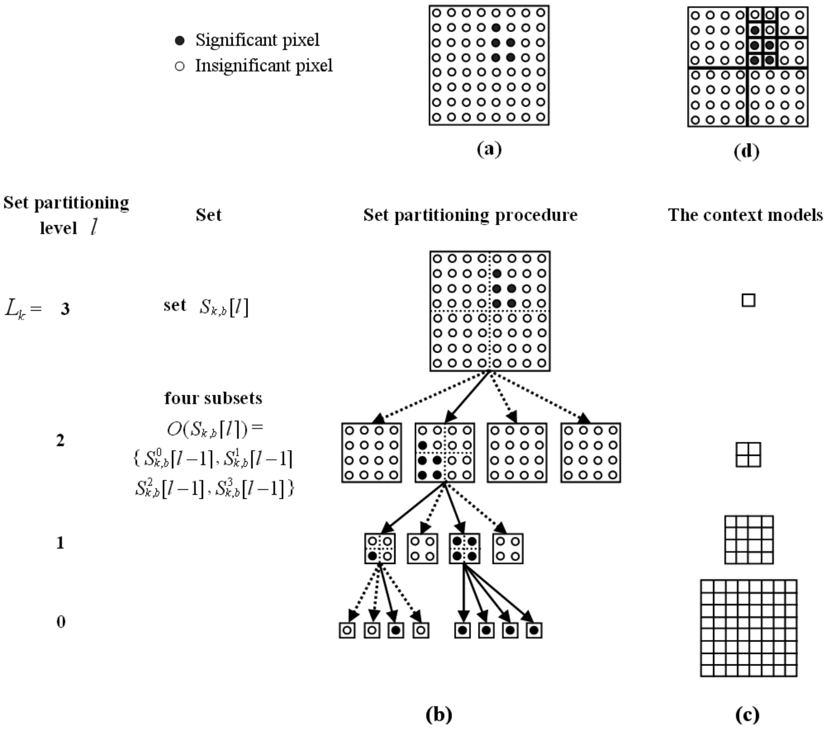

● Significant pixel
○ Insignificant pixel
(a)
(d)
Set partitioning level l
Set
Set partitioning procedure
The context models
$L_k =$ 3
set $S_{k,b}[l]$
2
four subsets
$O(S_{k,b}[l]) = \{S^0_{k,b}[l-1], S^1_{k,b}[l-1], S^2_{k,b}[l-1], S^3_{k,b}[l-1]\}$
1
0
(b)
(c)

Figure 4. Illustration of the set-based quadtree partitioning procedure and the context models of the 3D SPEZBC algorithm. (a) The original k^{th} subband of b^{th} spectral band. Initially, two arrays of lists (LIS_k and LSP_k) is maintained for subband *k*. (b) The set-based partitioning procedure. (c) The context models for subband *k*. (d) The Partitioning result for subband *k* of spectral band *b*

3.1 Initialization

```
➢ Output $n = \left\lfloor \log_2\{\max_{(i,j,b)} |c(i,j,b)|\} \right\rfloor$
➢ for $k = 0 : K-1$
    ❖ Set
      $LIS_k[l] = \begin{cases} \{S_{k,b}[l], \text{ namely } k^{th} \text{ subband}\}, & l = L_k \\ \phi, & \text{otherwise} \end{cases}$
    ❖ Set $LSP_k = \phi$.
```

3.2 Sorting Pass

```
for $l = 0 : L_{\max}$
    for $k = 0 : K-1$
            CodeLIS ($k$, $l$);

CodeLIS ($k$, $l$)
{
    For each set $S_{k,b}[l] \in LIS_k[l]$,
    ➢ Output $\Gamma_n(S_{k,b}[l])$;
    ➢ if $\Gamma_n(S_{k,b}[l]) = 1$
        ❖ if $l = 0$ ( namely the set $S_{k,b}[l]$ is a pixel)
            Output sign of $S_{k,b}[l]$, and remove $S_{k,b}[l]$ to $LSP_k$;
            else
            CodeSubSets( $S_{k,b}[l]$ ), and remove $S_{k,b}[l]$ from $LIS_k$[l].
}

CodeSubSets ( $S_{k,b}[l]$ )
{
    ➢ Partition $S_{k,b}[l]$ into four approximately equal sizes of subsets
      $O$ ( $S_{k,b}[l]$ ) ={ $S^0_{k,b}[l-1]$, $S^1_{k,b}[l-1]$, $S^2_{k,b}[l-1]$, $S^3_{k,b}[l-1]$ },
      where $S'_{k,b}[l-1] \in O(S_{k,b}[l])$.
    ➢ For each $S'_{k,b}[l-1]$
        ❖ Output $\Gamma_n(S'_{k,b}[l-1])$;
        ❖ if $\Gamma_n(S'_{k,b}[l-1]) = 1$
            ✧ if $l = 1$ ( namely the set $S'_{k,b}[l-1]$ is a pixel)
                  Output sign of $S'_{k,b}[l-1]$, and add $S'_{k,b}[l-1]$ to $LSP_k$;
              else
                  CodeSubSets ( $S'_{k,b}[l-1]$ ).
          else
              add $S'_{k,b}[l-1]$ to $LIS_k[l-1]$.
}
```

3.3 Refinement Pass

```
for $k = 0 : K-1$
    CodeLSP ( $k$ );

    CodeLSP ( $k$ )
{
    ➢ For each pixel set $S_{k,b}[l] \in LSP_k$ which correspond to pixel $c(i, j, b)$,
      output the $n^{th}$ MSB of | $c(i, j, b)$|except those included in the last sorting pass.
}
```

3.4 Quantization Step

Decrement n by 1 and go to step 2.

4. Discussion

The difference between two partitioning mechanism, the partitioning zeroblock coding based on set representation structure in 3D SPEZBC and the partitioning zeroblock coding based on quadtree representation structure in 3D EZBC, are only the different representation structures and testing mode for splitting the code block, but their splitting and coding results are same to each other. Moreover, our experimental results also show that the compression performances of 3D SPEZBC and 3D EZBC are totally the same. However, 3D SPEZBC saves considerable memory requirement in comparison with 3D EZBC due to the fact that the quadtree representation structure can be eliminated. For the k^{th} subband of size $M \times M$, the quadtree depth D_k of 3D EZBC is equal to $\log_2 M$. If the transformed wavelet coefficients are stored as the binary floating-point numbers (4 bytes), its quad-tree representation structure needs to be allotted

$$\sum_{i=0}^{D_k} (\frac{1}{4})^i \times M \times M \times sizeof(float) = \sum_{i=0}^{D_k} (\frac{1}{4})^i \times M^2 \times 4 \qquad (bytes)$$

memory usage. The quadtree nodes at the bottom quad-tree level 0 (namely i = 0) consist of the magnitudes of the wavelet coefficients in subband, so it can't be deleted. Therefore, when using the 3D SPEZBC algorithm, $\sum_{i=1}^{L_k} (\frac{1}{4})^i \times M^2 \times 4$ $(bytes)$ memory can be saved for this subband.

If four spatial levels and four spectral levels 3D WPT is employed for the hyperspectral image of size 512× 512×224, the pyramidal wavelet structure has 224 bands, and each band has 4 subbands of size 32×32, 3 subbands of size 64×64, 3 subbands of size 128×128, and 3 sub-bands of size 256×256. So the saved memory space in our algorithm against 3D EZBC is computed as

$$224 \times [4\times\sum_{i=1}^{5}(\frac{1}{4})^i \times 32^2 \times 4 + 3\times\sum_{i=1}^{6}(\frac{1}{4})^i \times 64^2 \times 4$$

$$+\ 3\times\sum_{i=1}^{7}(\frac{1}{4})^{i}\times 128^{2}\times 4\ +\ 3\times\sum_{i=1}^{8}(\frac{1}{4})^{i}\times 256^{2}\times 4]\ \ (bytes)$$

$$\approx\ 79185344\ \ (bytes)\ \approx\ 75.52\ \ (Mbytes)$$

The quadtree structure of 3D EZBC provides a fast quadtree splitting scheme by reducing the number of significance test. Nevertheless, the set-based partitioning zeroblock coding method of 3D SPEZBC is very simple, and its 2D code sets are smaller and are processed according to the increasing order of set size using the multi-list structure at the particular set partitioning level, as well as the 3D SPEZBC algorithm does not need time to establish quadtree structure, so it also exhibits excellent speed performance that is slightly worse than 3D EZBC. When our coder compresses four 512×512×224 AVIRIS hyperspectral images (such as Cuprite, Jasper Ridge, Low Altitude and Lunar Lake) on a AMD Athlon 3800+ CPU 2GHz machine, there are averagely 87.36 s for encoding and 125.23 s for decoding at 1.0 bpppb, 105.88 s for encoding and 170.74 s for decoding at 2.0 bpppb, as well as 135.10 s for encoding and 209.42 s for decoding at 3.0 bpppb, respectively.

5. Experimental Results

We performed coding experiments on four signed 16-bit radiance AVIRIS hyperspectral images [16], namely Cuprite scene 1, Jasper Ridge scene 1, Low Altitude scene 1 and Lunar Lake scene 1. In our experiments, we extracted the 256×256 lower left corner, so that the dimensions of the test image were 256×256×224 pixels. For lossy compression the rate distortion performance was compared by means of the signal-to-noise (SNR) values for a variety of bit rates in bits per pixel per band (bpppb), and for lossless compression performance we used those rates to evaluate the size of the compressed data streams. SNR is defined as $10\log_{10}\frac{\sigma_x^2}{MSE}$, where σ_x^2 is the average squared value (power) of the original AVIRIS image and *MSE* is the mean squared error over the entire sequence.

5.1 Lossless Compression Performance

Table 2 presents the lossless compression results for the 3D SPEZBC algorithm using various integer wavelet filters, which adopts the four spatial levels and four spectral levels Xiong's 3D integer WPT with Figure 2's unitary scaling factor. We can see that no certain wavelet filter is optimal for all test images. The 5/11-A, 13/7-C and 5/3 integer filters all provide good compression performances. Furthermore, Adams *et al.* [14] have found that the 5/3 filter evidently required the least computation, and experimental results in Subsection 5.2 further show that the integer-based lossy compression performance using the 5/3 integer filter clearly outperform that using the 5/11-A and 13/7-C integer filters at the medium and high bit rates. Therefore, Figure 5 displays the lossless compression ratios using the 5/3 integer filter in comparison with several state-of-the-art wavelet-based algorithms. In our experiments, JPEG2000-MC used the four spatial levels and four spectral levels 3D integer WPT and other algorithms used the four spatial levels and four spectral levels Xiong's 3D integer WPT with the unitary scaling factor in Figure 2. For all test images, the results show that 3D SPEZBC outperforms 3D SPECK, 3D SPIHT and AT-3D SPIHT, and it is worse than JPEG2000-MC. The average compression ratio of 3D SPEZBC is 5.70 % lower than 3D SPECK, 7.14 % lower than 3D SPIHT, 4.96 % lower than AT-3D SPIHT, and 1.07 % higher than JPEG2000-MC.

5.2 Integer-Based Lossy Compression Performance

The integer-based lossy compression results can be obtained when the decoder truncates the lossless encoded bitstreams in Subsection 5.1 at a desired bit rate. If we decode the hyperspectral image without losing any information, it is perfectly reconstructed. Table 3 shows the rate distortion results of the 3D SPEZBC algorithm using various integer wavelet filters for "Cuprite" image. We can see that these wavelet filters exhibit different coding performance at various bit rates. The 5/3 integer filter requires the least computation proved by Adams *et al.* [14] and provides excellent compression performance at medium and high bit rates. Moreover, when we apply the ISODATA and K-means unsupervised classification methods in comparison further (in Subsection 5.3), at more than 1.0 bpppb (16:1 compression ratio) the classification accuracy is higher than 99%. The experimental results on "Jasper Ridge", "Low Altitude" and "Lunar Lake"

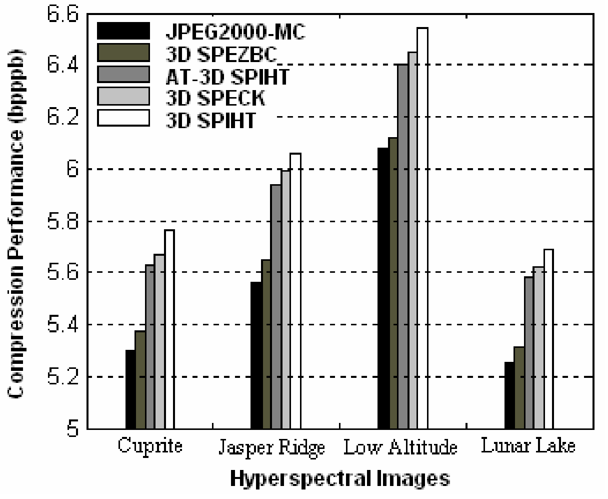


Figure 5. Lossless compression results (bpppb) in comparison with the state-of-the-art wavelet-based algorithms using 5/3 integer filter

Table 2. Lossless compression results (bpppb) for 3D SPEZBC using the various wavelet filters

Wavelet	Commpression Performance (bpppb)				
	Cuprite	Jasper Ridge	Low Altitude	Lunar Lake	Average
	5.44	5.66	6.17	5.39	5.67
(2+2,2)	5.39	5.63	6.11	5.35	5.62
(2,4)	5.38	5.67	6.14	5.32	5.63
(6,2)	5.42	5.66	6.14	5.38	5.65
5/3	5.37	5.65	6.12	**5.31**	5.61
2/6	5.42	5.69	6.21	5.36	5.67
2/10	5.46	5.70	6.21	5.40	5.69
9/7-M	5.39	5.63	6.11	5.35	5.62
9/7-F	5.40	5.67	6.12	5.34	5.63
5/11-A	5.37	**5.62**	**6.09**	5.32	**5.60**
13/7-C	**5.36**	5.63	6.10	**5.31**	**5.60**

Table 3. Integer-based lossy compression performance (SNR, in dB) in comparison with the various integer-based wavelet filters for the 3D SPEZBC algorithm

hyperspectral Image	Wavelet	Bit Rates (bpppb)								
		0.1	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
Cuprite	**S+P (B)**	39.33	47.95	51.95	55.05	56.69	59.01	60.38	62.27	63.17
	(2+2,2)	40.79	49.01	52.99	55.62	57.37	59.65	61.22	63.14	64.09
	(2,4)	40.50	48.90	53.13	55.99	57.94	**60.21**	61.75	63.60	64.49
	(6,2)	40.82	48.78	52.75	55.39	57.05	59.29	60.70	62.52	63.43
	5/3	40.71	49.19	**53.28**	**56.00**	**57.95**	**60.21**	**61.81**	**63.68**	**64.64**
	2/6	39.40	48.22	52.22	55.40	57.26	59.58	60.98	62.80	63.65
	2/10	39.28	47.95	51.82	55.03	56.84	59.09	60.57	62.43	63.36
	9/7-M	40.92	48.94	52.95	55.61	57.32	59.60	61.02	62.87	63.74
	9/7-F	40.55	48.96	52.90	55.55	57.26	58.73	59.72	60.84	61.37
	5/11-A	40.84	49.13	53.17	55.83	57.69	59.98	61.55	63.47	64.38
	13/7-C	**41.04**	**49.28**	**53.28**	55.96	57.74	59.93	61.23	63.05	63.75

hyperspectral images demonstrate the similar conclusions. Taking into these causes consideration, we think that the 5/3 integer filter is a very good choice for hyperspectral image integer-based lossy compression by using the 3D SPEZBC algorithm. For four hyperspectral images, Table 4 shows that the rate-distortion performance of the proposed algorithm is better than several state-of-the-art wavelet-based coding algorithms by using the 5/3 integer filter. We can see that the 3D SPEZBC algorithm outperforms the 3D SPECK, 3D SPIHT, AT-3D SPIHT and JPEG2000-MC algorithms at various bit rates. For all of the four hyperspectral images at 2 bpppb (8:1 compression ratio), 3D SPEZBC averagely overcomes 3D SPECK by 0.76 dB, 3D SPIHT by 1.22 dB, AT-3D SPIHT by 0.40 dB and JPEG2000-MC by 0.18 dB, respectively.

5.3 Classification Performance Comparison

In order to measure the influence of the aforementioned compression algorithms on the application performance of the reconstructed hyperspectral images, we applied the ISODATA and K-means unsupervised classification methods for comparison further, where we set the maximal number as ten classes and the maximal iterations as three. For the hyperspectral images, Table 5 gives the results of ISODATA and k-means unsupervised classification. The accuracy of the classification on 3D SPEZBC outperforms those of 3D SPECK, 3D SPIHT and AT-3D SPIHT, and is very close to those of JPEG2000-MC. For 3D SPEZBC at 1.0 bpppb (16:1 compression ratio), the classification accuracy is higher than 99%.

6. Conclusions

In this paper, we propose the 3D SPEZBC algorithm for hyperspectral image lossy-to-lossless compression, which is an improved 3D EZBC algorithm. It adopts the partitioning coding technique based on the set representation structure so as to avoid the problem with higher memory requirements for establishing the quadtree representation structure. According to the theoretical and experimental

Table 4. Integer-based lossy compression performance (SNR, in dB) in comparison with the state-of-the-art wavelet-based coding algorithms using 5/3 integer filter

hyperspectral Image	Coding Methods	Bit Rate (bpppb)						
		0.1	0.5	1.0	1.5	2.0	2.5	3.0
Cuprite	3D SPEZBC	**40.71**	**49.19**	**53.27**	**56.00**	**57.95**	**60.21**	**61.81**
	3D SPECK	39.92	48.62	52.61	55.49	57.57	59.71	61.45
	3D SPIHT	38.59	47.79	51.79	54.96	57.35	59.37	61.15
	AT-3D SPIHT	40.23	48.93	52.89	55.73	57.80	59.94	61.66
	JPEG2000-MC	40.56	49.02	53.18	55.85	57.89	60.01	61.53
Jasper Ridge	3D SPEZBC	**30.74**	**41.03**	**46.35**	**50.10**	**52.91**	**55.08**	**57.00**
	3D SPECK	30.11	40.13	45.50	49.25	52.18	54.27	56.39
	3D SPIHT	29.07	39.42	45.14	48.91	51.60	53.86	56.03
	AT-3D SPIHT	30.26	40.65	46.05	49.49	52.52	54.51	56.64
	JPEG2000-MC	30.62	40.89	46.14	49.78	52.81	54.65	56.58
Low Altitude	3D SPEZBC	**27.33**	**37.94**	**44.33**	**48.38**	**51.11**	**53.52**	**55.68**
	3D SPECK	26.74	37.05	43.28	47.42	50.12	52.78	54.69
	3D SPIHT	25.84	36.56	42.57	46.82	49.82	52.37	54.40
	AT-3D SPIHT	26.76	37.59	43.81	47.82	50.68	53.21	55.10
	JPEG2000-MC	27.18	37.72	44.06	47.99	50.75	53.15	54.91
Lunar Lake	3D SPEZBC	**43.20**	**50.88**	**54.76**	**57.33**	**59.25**	**61.35**	**62.94**
	3D SPECK	42.41	50.44	54.30	56.94	58.77	60.92	62.73
	3D SPIHT	41.11	49.70	53.49	56.31	58.58	60.55	62.39
	AT-3D SPIHT	42.59	50.64	54.51	57.14	58.94	61.08	62.81
	JPEG2000-MC	43.02	50.76	54.68	57.20	59.07	60.98	62.69

Table 5. Overall classification accuracy comparison (in %) based on ISODATA and K_mean at the various bit rates (bpppb) for lossy compression based on integer wavelet transform

hyperspectral Image	Coding Methods	K_mean					ISODATA				
		0.1	0.5	1.0	1.5	2.0	0.1	0.5	1.0	1.5	2.0
Cuprite	3D SPEZBC	**90.88**	**99.07**	**99.53**	**99.72**	**99.79**	**92.97**	**99.28**	**99.62**	**99.76**	**99.78**
	3D SPECK	90.58	99.03	99.49	99.70	99.78	92.72	99.20	99.58	99.75	99.77
	3D SPIHT	87.63	98.83	99.45	99.69	99.75	90.30	99.13	99.56	99.74	99.76
	AT-3D SPIHT	90.83	99.04	99.50	99.71	**99.79**	92.90	99.25	99.59	99.75	99.77
	JPEG2000-MC	90.86	99.06	99.52	99.71	**99.79**	92.93	99.27	99.61	**99.76**	99.77
Jasper Ridge	3D SPEZBC	**89.74**	**98.77**	**99.48**	**99.68**	**99.71**	**92.23**	**99.08**	**99.72**	**99.80**	**99.86**
	3D SPECK	89.63	98.70	99.43	99.53	99.66	92.07	98.97	99.65	99.75	99.84
	3D SPIHT	87.53	98.58	99.27	99.51	99.65	90.38	98.91	99.54	99.76	99.83
	AT-3D SPIHT	89.69	98.73	99.45	99.65	99.68	92.10	98.99	99.69	99.79	99.84
	JPEG2000-MC	89.72	98.74	99.46	99.66	**99.71**	92.20	99.04	99.70	**99.80**	99.85

analysis, our algorithm not only provides the same excellent compression performance as 3D EZBC, but also can save considerable memory requirements against 3D EZBC. For hyperspectral image lossy-to-lossless compression based on 3D SPEZBC, Xiong's 3D integer WPT with unitary scaling factor in Figure 2 and the 5/3 integer filter are good options. Compared with several state-of-the-art wavelet-based coding algorithms, the experimental results indicate that our coder provides better compression performance and unsupervised classification accuracy.

7. Acknowledgments

This work is supported by the Engagement Foundation of Xi’an University of Science and Technology (Project No.200722).

REFERENCES

[1] X. Tang and W. A. Pearlman, “Three-dimensional wavelet-based compression of hyperspectral images,” Hyperspectral Data Compression, MA: Kluwer Academic Publishers, pp. 273−308, 2006.

[2] J. E. Fowler and J. T. Rucker, “3D wavelet-based compression of hyperspectral imagery,” Hyperspectral Data Exploitation: Theory and Applications, John Wiley & Sons Inc., Hoboken, NJ, pp. 379−407, 2007.

[3] B. Penna, T. Tillo, E. Magli, and G. Olmo, “Transform coding techniques for lossy hyperspectral data compression,” in the Proceedings of IEEE Transactions on Geoscience and Remote Sensing, Vol. 45, No. 5, pp. 1408 −1421, 2007.

[4] T. W. Fry and S. Hauck, “Hyperspectral image compression on reconfigurable platforms,” in the Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 251−260, April 2002.

[5] X. Tang, S. Cho, and W. A. Pearlman, “3D set partitioning coding methods in hyperspectral image compression,” in the Proceedings of IEEE International Conference on Image Processing, pp. 239−242, September 2003.

[6] X. Tang and W. A. Pearlman, “Lossy-to-lossless block-based compression of hyperspectral volumetric data,” in the Proceedings of IEEE International Conference on Image Processing, pp. 1133−1136, 2006.

[7] J. J. Wu, Z. S. Wu, and C. K. Wu, “Lossy to lossless compressions of hyperspectral images using three-dimensional set partitioning algorithm,” SPIE Optical Engineering, Vol. 45, No. 2, pp. 0270051−0270058, 2006.

[8] B. Penna, T. Tillo, E. Magli, and G. Olmo, “Embedded lossy-to-lossless compression of hyperspectral images using JPEG 2000,” in the Proceedings of IEEE International Geoscience and Remote Sensing Symposium, Vol. 1, pp. 25−29, 2005.

[9] J. Zhang, J. E. Fowler, and G. Z. Liu, “Lossy-to-lossless compression of hyperspectral imagery using 3D-TCE and an integer KLT,” IEEE Geoscience and Remote Sensing Letters, Vol. 4, No. 2, pp. 201−205, 2008.

[10] S. T. Hsiang, “Highly scalable subband/wavelet image and video coding,” Ph.D dissertation, Rensselaer Polytechnic Institute, Troy, 2002.

[11] A. Islam and W. A. Pearlman, “An embedded and efficient low-complexity hierarchical image coder,” in the Proceedings of SPIE Conference on Visual Communications and Image Processing, Vol. 3653, pp. 294−305, 1999.

[12] Y. Hou and G. Z. Liu, “3D set partitioned embedded zero block coding algorithm for hyperspectral image compression,” in the Proceedings of SPIE Symposium on MIPPR, Vol. 6790, pp. 561−567, 2007.

[13] A. R. Calderbank, I. Daubechies, W. Sweldens, and B. L. Yeo, “Wavelet transforms that map integers to integers,” Applied and Computational Harmonic Analysis, Vol. 5, No. 3, pp. 332−369, 1998.

[14] M. D. Adams and F. Kossentini, “Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and analysis,” IEEE Transactions on Image Processing, Vol. 9, No. 6, pp. 1010−1024, 2000.

[15] Z. X. Xiong, X. L. Wu, S. Cheng, and J. P. Hua, “Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms,” IEEE Transactions on Medical Imaging, Vol. 22, No. 3, pp. 459−470, 2003.

[16] http://aviris.jpl.nasa.gov/html/aviris.overview.html.

J. Software Engineering & Applications, **2009, 2: 96-102**
doi:10.4236/jsea.2009.22014 Published Online July 2009 (www.SciRP.org/journal/jsea)

Adaptive Motion Segmentation for Changing Background

Yepeng Guan[1,2]

[1]School of Communication and Information Engineering, Shanghai University, Shanghai, China; [2]Key Laboratory of Advanced Displays and System Application, Ministry of Education, 149 Yanchang Rd., Shanghai, China.
Email: ypguan@shu.edu.cn

Received December 25th, 2008; revised April 12th, 2009; accepted May 4th, 2009.

ABSTRACT

Segmentation of moving objects efficiently from video sequence is very important for many applications. Background subtraction is a common method typically used to segment moving objects in image sequences taken from a statistic camera. Some existing algorithms cannot adapt to changing circumstances and require manual calibration in terms of specification of parameters or some hypotheses for changing background. An adaptive motion segmentation method is developed according to motion variation and chromatic characteristics, which prevents undesired corruption of the background model and does not consider the adaptation coefficient. RGB color space is selected instead of introducing complex color models to segment moving objects and suppress shadows. A color ratio for 4-connected neighbors of a pixel and multi-scale wavelet transformation are combined to suppress shadows. The mentioned approach is scene-independent and high correct segmentation. It has been shown that the approach is robust and efficient to detect moving objects by experiments.

Keywords: *Motion Segmentation, Background Update, Background Subtraction, Motion Variation, Shadow Suppression*

1. Introduction

Moving objects segmentation is an important topic in computer vision applications, including video conferences, vehicle tracking, and three-dimensional object identification, and has been actively investigated in recent years [1]. The most widely adopted approach for moving object segmentation with a fixed camera is based on background subtraction. A background (called as background model also) is computed and evolved frame by frame. A reliable background model has to account for background at each time instant. Mistake in labeling foreground and background points could cause wrong update of the background model. A particularly critical situation occurs whenever moving objects stop for a long time and become a part of the background. When these objects start again, a ghost is detected in the area where they stopped. This will persist for all the following frames, preventing the area to be updated in the background forever.

In addition, moving object segmentation is easily affected by shadow problem. Researchers try to select an optimal color space for shadow eliminating among a set of color spaces, such as *HSV*, *YCrCb*, *XYZ*, $L^*a^*b^*$, $L^*u^*v^*$,$C_1C_2C_3$, $l_1l_2l_3$, normalized *rgb* and so on, however, it remains open-ended how important is the appropriate color space, and which color space is the most effective [2]. Many approaches in literature have been developed so far. Some existing methods require manual calibration in terms of the specification of parameters which are related to the environment and the lighting conditions or make some hypotheses.

An approach to adaptive background updating and shadow suppressing is developed. RGB color space is selected instead of introducing complex color models to segment moving objects. Motion evaluation is introduced to prevent giving erroneous segmentation in those corresponding with an un-updated background model. A color ratio and multi-scale wavelet transformation are combined to suppress shadows. The main contribution of the proposal is that the developed approach is scene-independent and automatic background updating according to motion variations caused by moving objects. The second contribution is that when segmenting motion objects it does not require any complex supervised training or manual calibration in terms of the specification of parameters or makes any hypotheses. Experimental results from indoor and outdoor environments have shown

[*]Tel: +86 21 56331967; Fax: +86 21 56336908.

the developed approach is efficient and flexible during segmenting moving objects and suppressing shadows in applications.

The remainder paper is organized as follows. Section 2 briefly reviews some related previous works. In the next section, background model update would be discussed. Shadow suppression is described in Section 4. Experimental results from indoor and outdoor environments are given in Section 5 and followed by conclusions in Section 6.

2. Related Works

Many works have been put forward in literature for moving objects detection. Background subtraction based moving objects in the scene are detected by the difference between the current frame and the background model. When the deviation is greater than some critical value, the pixel is considered as foreground (moving object) [3]. The most simple background model is the previous frame. The difference between the observed frame and the previous frame is thresholded to determine which pixel is background and which pixel is the foreground. Another way to model the background is to take the mean, median, or minimum and maximum values of the previous *N* pixel [4]. It needs to keep track of the pixel value history. To avoid this problem a first order recursive filter is used to update the background model [5]. It easily causes ‘tailing’ or ‘ghosting’. A particularly critical situation occurs whenever the moving object stopped for a long time and became a part of the background. When these objects start again, a ghost is detected in the area where they stopped [6]. This will persist for all the following frames, preventing the area to be updated in the background image forever [7]. One method is by modeling each pixel as unimodal Gaussian distribution [8]. It fails to model background pixels that are subject to repetitive motions which have multiple background colors. To overcome these difficulties, a parametric background modeling is done by modeling each background pixel value as a mixture Gaussian distribution [9,10]. The parametric background model still lacks flexibility when dealing with non-static backgrounds, a highly flexible non-parametric technique is proposed for estimating background probabilities from recent samples over time using Kernel density estimation [11]. False detections due to fluctuating backgrounds are still not covered in the algorithm until now. Codebook technique is a different approach proposed for the background subtraction in [12]. One of drawbacks is that the algorithm cannot adapt to changing circumstance when the environment was not present in the training phase. Moving objects that stop moving and should be adopted into the background will get difficulties in the algorithm.

Shadows cause serious problems while segmenting moving objects, due to the misclassification of shadow-points as a foreground. Many works have been developed to suppress shadow [1,2,6,13,14,15,16,17,19,21,22]. By shadow suppression, the major problem is how to distinguish moving cast shadows from moving object points [15]. Cucchiara *et al*. [6] defined a shadow mask for each point resulting from motion segmentation. However, it often makes additional assumptions such as small changes in hue and saturation, necessary a prior knowledge of the bands of the changes in the value channel. Moreover choice of some parameters is less straightforward and for now is done empirically. Tattersall *et al*. [16] proposed adaptive shadow identification through automatic parameter estimation based on the above method. The single variable parameter is only used. However, much additional assumptions must be made also in [16]. Salvador *et al*. [17] proposed invariant color features to detect cast shadows through using chrominance color components. It is found that several assumptions are needed regarding the reflecting surfaces and the lightings. In outdoor scene, shadows will have a blue color cast due to the sky, while the lighting regions have a yellow cast (sunlight), hence the chrominance color values corresponding to the same surface point may be significantly different in shadow and sunlit regions [18]. Cavallaro *et al*. [19] proposed the normalized *rgb* space to detect shadows. It is known that the practical application of normalized *rgb* suffers from a problem inherent to the noise at low intensities which would result in unstable chromatic components [20]. Texture analysis can be potentially effective in solving the problem. Heikkila *et al*. [21] proposed a texture-based method for modeling the background and detecting moving objects from a video sequence. Each pixel is modeled as a group of adaptive local binary pattern histograms that are calculated over a circular region around the pixel. Because of the huge amount of different combinations, it must be done more or less empirically to find a good set of parameter values. Spagnolo *et al*. [22] proposed a ratio-based algorithm to detect shadows with an empirically assigned ratio threshold. This ratio-based algorithm considering the ratio between only two adjacent pixels considerably shortens the computation time, but it easily misclassifies shadows as objects, because ratio magnitude of shadows may have a similar magnitude value.

3. Background Model Update

The main idea of the proposed approach is to update pixels according to motion variations caused by moving objects, which has been proven to be more reliable and less sensitive to noise.

Assuming the background model B^{t+1}(x, y) at *t*+1 time, extract possible moving regions *P* based on background subtraction (seen from Figure 1(c)). According to the fact that the variation of sensible motion target in the scene can be found out from the sequence, the difference between the two adjacent frames is used to extract moving regions R (seen from Figure 1(d)). Extract moving object regions P_1 from the regions P according to moving objects with the same connective characteristic as regions R (seen from Figure 1(e)).

When the object moves slowly in situ, the extracted region P_1 may be incorrect (seen from Figure 2(e)). In order to overcome the problem, chromatic information is used to further extract moving region P_2 from P_1 according to moving objects with the same chromatic in P_1 as in R (seen from Figure 2(f)). According to the mentioned above, the background model is updated as follows,

$$B^{t+1}(x,y) = \begin{cases} I^{i+1}(x,y) \to B^{t+1}(x,y), & \text{if} \quad \arg XOR(P,P_2) \geq T \\ B^{t}(x,y), & \text{otherwise} \end{cases} \tag{1}$$

where

$$P_1 = connect(R,P), P_2 = chrom(P_1,R) \tag{2}$$

In (1), I^{t+1} is a current frame at t+1 time, "→" represents updating, Connect (R, P) represents selecting a region with the same connective characteristic as R from P, chrom (P_1, R) represents selecting a region with the same chromatic information in R as in P_1, XOR is a logical exclusive-or operator, T is a threshold value.

The proposed background update approach reveals some advantages. Firstly it smoothes and reduces the effects of noise in the image since sudden variations of a single pixel are not included in the background model. Secondly it does not consider the adaptation coefficient or the learning rate used in the existing literatures. Finally it does not depend on static or moving objects in the image. In order to demonstrate the last point, some results of two sequences where lena walks, and she bends in two scenes from http://www.tele.ucl.ac.be/~gaitanis/results/Human_Action_Video_Database/2Feet/ are given in Figures 1 and 2, respectively.

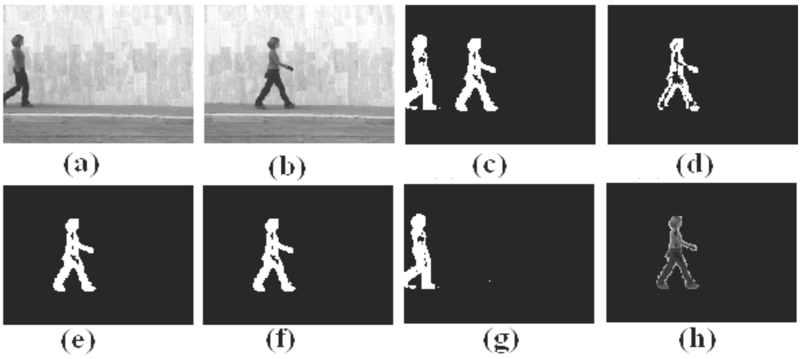


Figure 1. Lena walks in the scene and the extracted results. (a) First frame in the sequence. (b) The 25th frame. (c) Segmentation based on subtraction of (a) and (b). (d) Segmentation based on difference between the 24th and 25th frames. (e) Extracted object region by connectivity. (f) Extracted object region by chromatic consistency. (g) Detected varied background region. (h) Extracted final foreground region

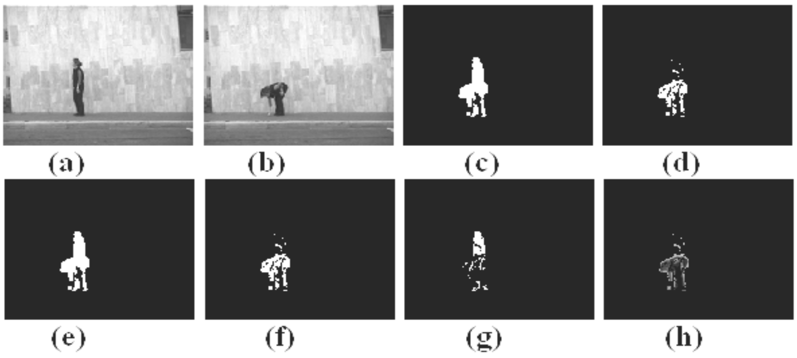


Figure 2. Lena bends in situ and the extracted results. (a) First frame in the sequence. (b) The 25th frame. (c) Segmentation based on subtraction of (a) and (b). (d) Segmentation based on difference between the 24th and 25th frames. (e) Extracted object region by connectivity. (f) Extracted object region by chromatic consistency. (g) Detected varied background region. (h) Extracted final foreground region

In the second example, some existing algorithms in the literature are difficult to correctly extract moving objects. In Figures 1 and 2, the ghosts are removed since their connective pixels do not contain any frame-difference pixels. From Figures 1(g) and 2(g), one can see that some varied background pixels are detected in the region no matter how lena walks or bends in situ. The results obtained with the approach show correct foreground segmentation and effective background updating from Figures 1 (h) and 2(h). It highlights that the aperture problem resulting from the adjacent frame-difference is overcome by combining region connectivity and the chromatic consistency.

4. Shadow Suppression

Since RGB color camera system is one of the most popular color spaces, and all colors are seen as a variable combination of the three primaries in the RGB color space, RGB color space is selected to eliminate shadows in the paper. Before segmenting image, start by applying a smoothing operator both to the background image *B*(*x*, *y*) and the current frame *F*(*x*, *y*).

Calculate a color ratio difference between *B*(*x*, *y*) and *F*(*x*, *y*) for 4-connected neighbors of a pixel and for all three color components as:

$$p^n_{k\in R,G,B}(x,y,i,j)=\frac{I^F_k(x,y)}{I^F_k(x+i,y+j)}-\frac{I^B_k(x,y)}{I^B_k(x+i,y+j)}(i=\pm 1,\ j=\pm 1, i\neq j, n=1,2,3,4) \tag{3}$$

where $I^F_k(x,y)$ is the intensity of *F*(*x*, *y*) for the *k*th color component at location (*x*, *y*), $I^F_k(x+i,y+j)$ is the intensity of its neighbor color components; $I^B_k(x,y)$ is the intensity of *B*(*x*, *y*) for the kth color component at location (*x*, *y*), and $I^B_k(x+i,y+j)$ is the intensity of its neighbor color components.

Since moving shadow makes the region covered by itself darker than the background and it has similar chromaticity, a suitable threshold *Th* is applied to the color ratio difference (3) to obtain a candidate foreground region $SP^n(x,y)$ as:

$$SP^n(x,y)=\begin{cases}1 & if \quad p^n_k \geq Th\\ 0 & otherwise\end{cases} \tag{4}$$

If the distortion distribution of p^n_k is assumed to be a Gaussian one, we can threshold the distortion by $\sigma_{p^n_k}$, where $\sigma_{p^n_k}$ is a standard deviation of p^n_k. However, it is found from experiments that the distribution of p^n_k is not a Gaussian one, but is has a very sharp peak at zero. The standard deviation of p^n_k is used to select the threshold *Th*, and (4) can be rewritten as:

$$SP^n(x,y)=\begin{cases}1 & if \quad p^n_k \geq \sigma_{p^n_k}\\ 0 & otherwise\end{cases} \tag{5}$$

The value of color ratio mentioned above may not uniquely highlight the property of a particular material, and there may be instable for particular values. To obtain a robust segmentation result, the results from the color ratio and multi-scale wavelet transformation method proposed in [23] are combined.

5. Experimental Results

To confirm the effectiveness of the above proposed method, we have conducted experiments with different indoor and outdoor video sequences.

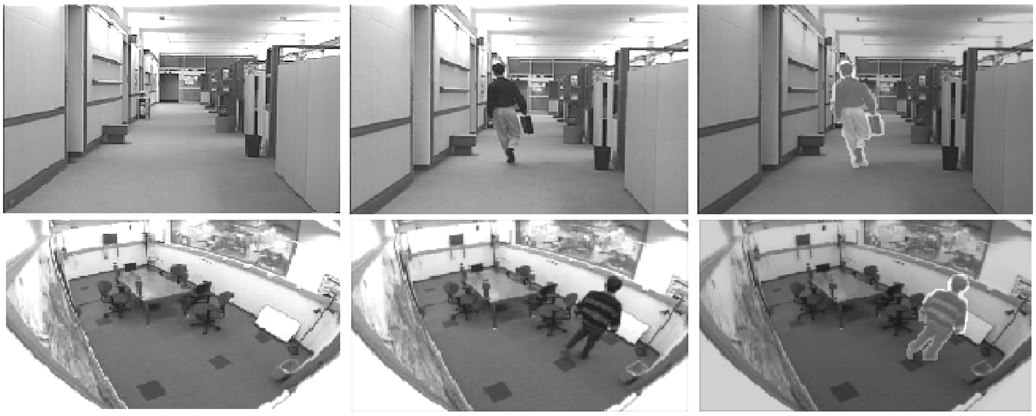


Figure 3. Detected foregrounds for indoor circumstances. The first column is the first frame of the sequences, the second column is the current frames (67#, and 300#, from top to bottom, respectively), and the third column is the detected foreground masks with light grey pixels superimposed on the original image

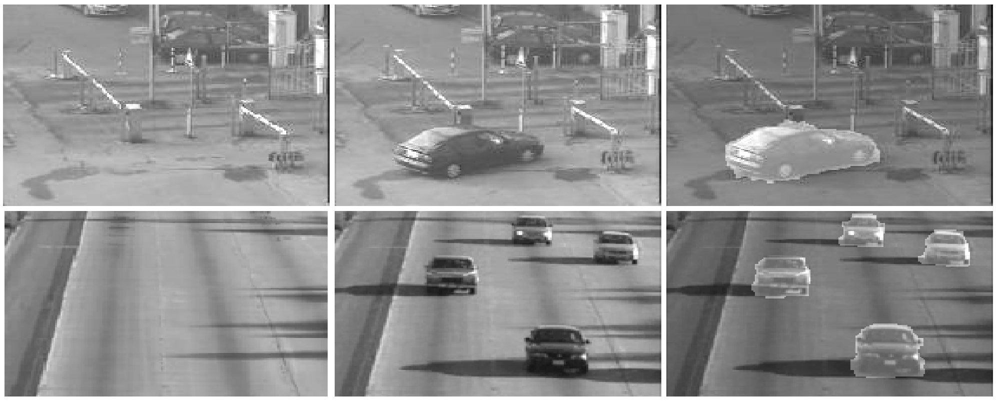


Figure 4. Detected foregrounds in outdoor circumstances. The first column is the first frame of the sequences, the second column is the current frames (69#, and 84#, from top to bottom, respectively), and the third column is the detected foreground masks with light grey pixels superimposed on the original image

In the experiments, the threshold T used in (1) is chosen as 100, which represents the minimum number of expected background updating in consecutive frames. The threshold T can be chosen lower in which more background pixels can be updated. The selection of lower T means more time to process background updating and some noises may drift into updating. The background reference image used is the first frame of sequence no matter objects enter the field of view before captured or not.

The sequences with different indoor conditions including the MPEG-4 test sequence *Hall Monitor*, aton project test sequence *intelligent room* from http://cvrr.ucsd.edu/aton/shadow have been used to test the developed algorithm. Some results are given in Figure 3.

In Figure 3, the results obtained by the developed approach are quite good, considering that the contrast between the object and background is very low.

In order to test the proposed approach further in outdoor conditions, the sequences with different outdoor conditions including *campus* and *highway* from the *ATON* project are used. The choice of such different scenes is made to emphasize the reliability and robustness of the proposed approach in outdoor circumstances. Some results are shown in Figure 4.

In Figure 4, the results show the robustness of the proposed algorithm to cope with different outdoor circumstances.

The above results show qualitative information about the effectiveness of the developed approach. It is necessary to quantitatively evaluate the performance of the method with a ground-truth.

The main goal of the proposed method is not accurate detection or discrimination of shadow pixels, but the improvement of moving object detection because accurate object detection is crucial for further applications. The performance of moving object detection is measured in term of correct segmentation rate (*csr*) and false segmentation rate (*fsr*) as:

$$csr = 2\frac{TP \cap GT}{TP + GT},\ fsr = \frac{FP + FN}{2GT} \tag{6}$$

where *TP* (true positive) is the number of correctly detected foreground pixels, *GT* (ground-truth) is the ground-truth for the foreground, *FP* (false positive) is the number of background falsely marked as foreground pixels, *FN* (false negative) is the number of foreground pixels falsely classified as background ones.

Using the *csr* and *fsr* measurements, the total correct segmentation will have 100% by *csr* while the total agreement with the ground-truth will have 0% by *fsr*.

The available sequence *intelligent room* and its ground-truth data are available from http://cvrr.ucsd.edu/aton/shadow. In our work, two recent proposed methods including the invariant color features (ICF) proposed in [17] and the ratio map (RM) developed in [22] are quantitatively evaluated together, and the results are shown in Figure 5 for a comparison.

The symbols in the legend of Figure 5 refer to object extraction results: ICF [17], RM [22], and the proposed method. The mean values of *csr* corresponding to the plots of Figure 5 are the following: RM 0.84, ICF 0.86, proposed 0.98. The mean values of *fsr* are the following: RM 3.42, ICF 4.11, proposed 1.61.The proposed method outperforms the investigated methods with the best object detection over time.

6. Conclusions

This paper has presented an approach to adaptive motion segmentation and shadow suppression. The ghosts are detected and removed by the developed background update function, which prevents undesired corruption of the background model and does not consider the adaptation coefficient or the learning rate used in the literature. By

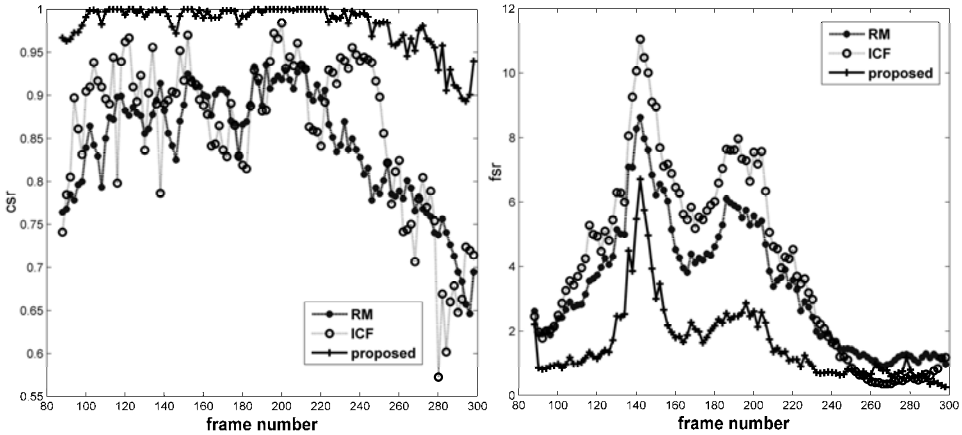


Figure 5. Comparison of video object segmentation rate for the test sequence intelligent room

comparison, it has been shown that the proposed method outperforms the investigated methods and is robust and efficient to detect moving objects during coping with different indoor or outdoor circumstances.

7. Acknowledgements

This work in part is supported by the National Natural Science Foundation of China (Grant No. 60872117).

REFERENCES

[1] W. Zhang, X. Z. Fang, X. K. Yang, and Q. M. J. Wu, “Moving cast shadows detection using ratio edge,” IEEE Transactions on Multimedia, Vol. 9, No. 6, pp. 1202–1214, September 2007.

[2] C. Benedek and T. Sziranyi, “Study on color space selection for detecting cast shadows in video surveillance,” International Journal of Imaging Systems and Technology, Vol. 17, No. 3, pp. 190–201, October 2007.

[3] T. Horprasert, D. Harwood, and L. S. Davis, “A statistical approach for real-time robust background subtraction and shadow detection,” Proceedings of IEEE International Conference on Computer Vision Frame-Rate Workshop, pp. 1–19, September 1999.

[4] I. Haritaoglu, D. Harwood, and L. S. Davis, “W4: real time surveillance of people and their activities,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 8, pp. 809–830, August 2000.

[5] J. Heikkila and O. Silven, “A real-time system for monitoring of cyclists and pedestrians,” Proceedings of 2nd IEEE Workshop on Visual Surveillance, pp. 74–81, July 1999.

[6] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, “Improving shadow suppression in moving object detection with HSV color information,” Proceedings of IEEE International Conference on Intelligent Transportation Systems, pp. 334–339, August 2001.

[7] P. Spagnolo, T. D'Orazio, M. Leo, and A. Distante, “Moving object segmentation by background subtraction and temporal analysis,” Image and Vision Computing, Vol. 24, No. 5, pp. 411–423, May 2006.

[8] C. Wren, A. Azabayejani, T. Darrell, and A. Pentland, “Pfinder: Real-time tracking of the human body,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, pp. 780–785, July 1997.

[9] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” Proceedings of Computer Vision and Pattern Recognition, Vol. 2, pp. 246–252, June 1999.

[10] Z. Zivkovic and F. van der Heijden, “Efficient adaptive density estimation per image pixel for the task of background subtraction,” Pattern Recognition Letters, Vol. 27, No. 7, pp. 773–780, May 2006.

[11] A. Elgammal, D. Harwood, and L. S. Davis, “Nonparametric model for background subtraction,” Proceedings of 6th European Conference on Computer Vision, Vol. 2, pp. 751–767, June 2000.

[12] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, “Real-time foreground-background segmentation using codebook model,” Real-Time Imaging, Vol. 11, No. 3, pp. 172–185, June 2005.

[13] J. Stander, R. Mech, and J. Ostermann, “Detection of moving cast shadows for object segmentation,” IEEE Transactions on Multimedia, Vol. 1, No. 1, pp. 65–76, March 1999.

[14] I. Mikic, P. C. Cosman, G. T. Kogut, and M. M. Trivedi, “Moving shadow and object detection in traffic scenes,” Proceedings of International Conference on Pattern Recognition, Vol. 1, pp. 321–324, September 2000.

[15] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, “Detecting moving shadows: Formulation, algorithms and evaluation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 7, pp. 918–924, July 2003.

[16] S. Tattersall and K. Dawson-Howe, “Adaptive shadow identification through automatic parameter estimation in

video sequences," Proceedings of Irish Machine Vision and Image Processing, pp. 57−64, September 2003.

[17] E. Salvador, A. Cavallaro, and T. Ebrahimi, "Cast shadow segmentation using invariant color features," Computer Vision and Image Understanding, Vol. 95, No. 2, pp. 238−259, August 2004.

[18] E. A. Khan and E. Reinhard, "Evaluation of color spaces for edge classification in outdoor scenes," Proceedings of International Conference on Image Processing, Vol. 3, pp. 952−955, September 2005.

[19] A. Cavallaro, E. Salvador, and T. Ebrahimi, "Detecting shadows in image sequences," Proceedings of IEEE Conference on Visual Media Production, pp. 165−174, March 2004.

[20] M. Kampel, H. Wildenauer, P. Blauensteiner, and A. Hanbury, "Improved motion segmentation based on shadow detection," Electronic Letters on Computer Vision and Image Analysis, Vol. 6, No. 3, pp. 1−12, December 2007.

[21] M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, No. 4, pp. 657−662, April 2006.

[22] P. Spagnolo, T. D. Orazio, M. Leo, and A. Distante, "Advances in shadow removing for motion detection algorithms," Proceedings of 2nd International Conference on Vision Video Graphics, pp. 69−75, July 2005.

[23] Y. P. Guan, "Wavelet multi-scale transform based foreground segmentation and shadow elimination," The Open Signal Processing Journal, Vol. 1, No. 6, pp. 1−6, November 2008.

J. Software Engineering & Applications, 2009, 2: 103-110
doi:10.4236/jsea.2009.22015 Published Online July 2009 (www.SciRP.org/journal/jsea)

Autonomic Software Component QoS Matchmaking Algorithm Based on Fidelity Factor in Agent-Based Autonomic Computing System

Kun Zhang[1,2], Manwu Xu[2], Hong Zhang[1]

[1]School of Computer Science & Technology, Nanjing University of Science & Technology, Nanjing, China; [2]State Key Laboratory for Novel Software, Nanjing University, Nanjing, China.
Email: zhangkun@mail.njust.edu.cn

Received December 8th, 2008; revised March 20th, 2009; accepted March 31st, 2009.

ABSTRACT

Autonomic software component (ASC) QoS matchmaking problem for autonomic element has been taken as one of the most important issue in field of autonomic computing based on agent. Aimed at overcoming drawbacks such as subjectiveness and unfairness, and improving the self-configuration capability for autonomic element, we introduce evaluation mechanism of confidence of individual QoS attributes during ASC QoS matchmaking, i.e., fidelity factor for each attribute, and propose an ASC QoS matchmaking algorithm based on fidelity factor. Simulation experiments demonstrate that our proposed algorithm performs best performance in terms of QoS than other existing algorithms, and has better compromise between attribute quality and users' evaluation when selecting ASC.

Keywords: *Autonomic Software Component, QoS Matchmaking, Fidelity Factor, Autonomic Computing, Autonomic Element*

1. Introduction

In mid-October 2001, aimed at the problem of looming software complexity crisis, IBM Company innovatively proposed autonomic computing [1] technology – computing systems that can manage themselves given high-level objectives from administrators. An autonomic software component (ASC, or element, in IBM parlance) [1] is—

"The fundamental atom of autonomic applications and systems—a modular unit of composition with contractually specified interfaces, explicit context dependencies, and mechanisms for self management, responsible for providing its own services, constraints (e.g., system resource requirements, performance requirements, etc.), managing its own behavior in accordance with context, rules, and policies, and interacting with other autonomic components. [1]"

Autonomic software components will have complex life cycles, continually carrying on multiple threads of activity, and continually sensing and responding to the environment in which they are situated. Autonomy, proactivity, and goal-directed interactivity with their environment are distinguishing characteristics of software agents. Viewing autonomic software components as agents and autonomic systems as multi-agent systems makes it clear that agent-oriented architectural concepts will be critically important [2].

In order to efficiently accomplish the self-configuration and self-management between ASCs, one of the most important issues is how to design an efficient autonomic software component matchmaking algorithm to find appropriate provider(s) for a consumer according to the providers' advertising description.

However, most of the existing algorithms are only concerned with functional matchmaking on software component. Few of them focused on non-functional factors, such as software component cost, time, reliability, satisfaction, i. e. quality-of-service (QoS) of software component. Therefore, when more ASCs than one can provide a functional component, these algorithms just randomly selected one from these ASCs, and could not be obtained optimal ASC with high QoS performance. Moreover, only a few existing algorithms are considered QoS factors during software component matchmaking. However, these algorithms were paid regard to QoS level as a whole, ignoring confidence of individual QoS attributes. Due to QoS data was often advertised by software component providers, it suffered from the drawbacks such as subjectiveness and unfairness. And these algorithms didn't be considered the user's feeling and satisfaction.

Aimed at overcoming the above difficulties, we introduce evaluation mechanism of confidence of individual QoS attributes during ASC QoS matchmaking, i.e., fidelity factor for each attribute, and propose an ASC QoS matchmaking algorithm based on fidelity factor, which will improve the facility and fairness concerned with QoS attributes, and objectively select optimal ASC with QoS performance.

The rest of the paper is organized as follows. Some of the existing work related to this paper is described in Section 2. Section 3 gives our proposed ASC matchmaking algorithm in detail. A case study and simulation results are presented in Section 4. Section 5 concludes the paper.

2. Related Work

In recent years, much research has been devoted to the development of software component matchmaking algorithms for agent or autonomic element [3,4,5,6,7,8]. Wickler [3] addressed the problem of capability brokering agent, and proposed a new capability description language (CDL) for the representation of agent capabilities. Sycara [4] defined a language called LARKS for agent advertisements and requests, and presented a flexible and efficient matchmaking process that used LARKS. LARKS performed both syntactic and semantic matching, and in addition allowed the specification of concepts (local ontologies) via ITL, a concept language. The establishment for semantic distance consumed much workload, so that their matching algorithm had limitations on practicability and reliability. Arisha *et al.* [5] provided approximate software agent service matchmaking by using semantic distance. Whereas, the algorithm could not support definition of data type and descript software component efficiently. However, neither of the above algorithms considered the factors of quality-of-service attribution, such as cost, time, and reliability. Zhang [6] and Jiang [7] analyzed the drawbacks in previous papers, i.e., the matchmaking was only based on the advertised capabilities of provider agents or software components. They considered the matchmaking influenced by QoS of software components, and individually presented software components matchmaking algorithm based on QoS. In [6], it was argued that the practical performance of provider agents had a significant impact on the matchmaking outcomes of middle agents. The authors' proposed algorithm could pick up the provider agents based on the history information in accomplishing similar tasks in the past rather than choosing randomly. At the launching of an agent system, the proposed algorithm provided initial values of the track records. With agents' history information and the initial values of the track records, the quality of matchmaking algorithms could be improved significantly, and the returned results were more accurate and reasonable. Jiang [7] pointed out that there were two drawbacks using the track records in [6], i.e., the value of track records was too subjective, and track record model is too simple to judge agent services' performance more accurately. So, the authors in [7] presented QoS-driven matchmaking algorithm, which aimed at matching the best satisfying agent for user. It is pity for the above two algorithms that the authors didn't consider the user's feeling and satisfaction when selecting agents or software components. The authors in [8] studied agent service selection and matchmaking in manufacturing industry field. They applied the idea of rough sets theory and fuzzy information filter to reclaim agent selection problem and realized reasonable evaluation for recycling quotient. Their algorithm was easy to select optimum reclaiming agent for manufacturer. However, their method belonged to specific field and non-universal.

Besides software component or agent service matchmaking, some researchers took software component or agent as web service, and proposed many QoS-aware web services matchmaking or selection algorithms [9,10,11,12,13,14]. Some theoretic methods or principle in these algorithms can be used for reference when matchmaking software component, due to many similarities between software component and web service. Ma [9] proposed a semantic QoS-aware framework for semantic web services discovery, and presented a selection algorithm to obtain the optimal offer from clients' viewpoint under complex QoS conditions through confirming the compatibility of concepts. Hu *et al.* [10] proposed a novel and extended web service QoS model (attributes include *time*, *cost*, *reliability* and *interest*) by adding an interest degree property, and then put forward to a QoS matchmaking algorithm based on this model. Aimed at resolving such conflicts to ensure consensus on the QoS characteristics in the selection of web services, Wei [11] proposed a QoS Consensus Moderation Approach (QCMA) in order to perform QoS consensus and to alleviate the differences on QoS characteristics in the selection of web services. QCMA enhanced the moderation for opinion similarity and preference on QoS attributes and was been designed a mechanism for providing group consensus moderation on QoS. Giallonardo *et al.* [12] used ontology to describe QoS-based WS specifications and reasoners to perform the matchmaking. They addressed QoS semantics to improve the recall of the matching process by exploiting ontology knowledge, and their QoS specification approach used metrics to understand, describe, and control the QoS in the matching process. However, they didn't offer useful results for over-constrained demands. In addition, although they were capable of, they did not perform semantic QoS metric matching. Guo *et al.* [13] proposed three-dimensional QoS model of web services, and designed a web services selection algorithm. In their algorithm, the concept and measure method of web service effectiveness were proposed, and then a web service ranking algorithm based on the effectiveness of web service was designed. The comprehensive experiment showed their proposed model possessed high precision, high response rate and better influence on the load balance of web service. Liang Kai-jian [14] studied QoS support problem in Service-oriented Grid system, and presented a new parameter called GSQN(Grid Service Quality Number) to depict the profile of qualitative characteristics of a grid service. Based on this idea, the author proposed a strategy of matchmaking with a price model to match grid service.

3. Our Proposed Matchmaking Algorithm

3.1 Problem Description

Definition 1. Autonomic software component matchmaking based on QoS problem. Given the set of ASC providers $PASC = \{ pasc_1, pasc_2, ..., pasc_n \}$, an ASC requester *rasc*, autonomic software component matchmaking based on QoS problem is to find an ASC such that their QoS level is maximized.

3.2 QoS Model for Autonomic Software Component

Definition 2. Autonomic software component QoS model for autonomic element is a 6-tuple as follows:

$$qos\_model\_asc = (t, c, rel, m, rep, fid) \qquad (1)$$

where *t* is ASC response time from sending request to receiving result, including process time and transmission delay time, i.e., $q_{time}(asc) = T_{process}(asc) + T_{trans}(asc)$; *c* is ASC cost, and represents the expenses paid by users to ASC provider; *rel* is a metric of reliability, denoting the probability of ASC providing its registered software component; *rel* is a technical measure related to hardware and/or software configuration of ASCs and the network connections between the ASC requesters and providers. *m* denotes the probability of accurate maintenance when an exception occurs for an ASC; *rep* is a measure of ASCs' trustworthiness or satisfaction degree, and denotes users' (or requesters') satisfaction to ASC. *rep* mainly depends on end user's experiences of using the ASC. Different end users may have different opinions or satisfaction degree on the same ASC. Usually, the end users or requesters are given a range to rank or score ASCs, for example, in Amazon.com, the range is [0,5]. In this paper, the range is [0,1].

The first four QoS attributes are often published by ASC providers themselves, and describe the basic QoS performance of an ASC, so most of the authors took them as quality criteria for an ASC. However, such criterion is too subjective to reflect actual quality of ASC. In this paper, we introduce attribute "*reputation*" to measure users' satisfaction degree. On the other hand, *fidelity* vector is added to the QoS model to improve the impartiality and objectivity. Fidelity is treated as a vector composed of fidelity attributes. Each fidelity attribute refers to the confidence or fidelity of above first four QoS attributes, i.e.,

$$fid = \left\langle fid_t, fid_c, fid_{rel}, fid_m \right\rangle \qquad (2)$$

3.3 Matchmaking Algorithm Description

In our proposed QoS matchmaking algorithm, the drawbacks in [6] and [7] such as subjectiveness and unfairness are overcame through introducing evaluation mechanism of confidence of individual QoS attributes, i.e., fidelity factor for each attribute, which can improve the self-configuration capability for autonomic element.

The basic idea of our algorithm is as follows: firstly, normalizing each attribute in QoS description to range [0,1] for each ASC in initial set; then fidelity of each attribute is considered to evaluate QoS overall performance for every ASC comprehensively and objectively; finally, an ASC whose total QoS value is maximal is picked out from all candidates.

Concretely, suppose that there is a set of ASC providing a certain software function, i.e. $PASC = \{ pasc_1, pasc_2, ..., pasc_n \}$. By merging the quality vectors of all these candidates, a matrix $\mathbf{Q} = \left(Q_{i,j}, 1 \le i \le n, 1 \le j \le 5 \right)$ is built according with Definition 2, in which each row Q_j corresponds to an ASC $pasc_i$ while each column corresponds to a QoS attribute value. Let a matrix $\mathbf{F} = \left(fid_{i,j}, 1 \le i \le n, 1 \le j \le 4 \right)$ denote confidence fidelity of the first four QoS attributes for $pasc_i$. There are two phases as follows:

1) Scaling phase.

Some of the QoS attributes could be negative, i.e., the higher the value, the lower the quality, such as *time*, *cost*. Other QoS attributes are positive, i.e., the higher the value, the higher the quality, such as *reliability*, *maintainability*, and *reputation*. For negative attributes, values are scaled according to (3). For positive criteria, values are scaled according to (4).

$$M_{i,j} = \begin{cases} \dfrac{Q_j^{\max} - Q_{i,j}}{Q_j^{\max} - Q_j^{\min}} & if\ Q_j^{\max} - Q_j^{\min} \ne 0 \\ 1 & if\ Q_j^{\max} - Q_j^{\min} = 0 \end{cases}, \quad j = 1, 2 \quad (3)$$

$$M_{i,j} = \begin{cases} \dfrac{Q_{i,j} - Q_j^{\min}}{Q_j^{\max} - Q_j^{\min}}, if\ Q_j^{\max} - Q_j^{\min} \neq 0 \\ 1 \qquad\qquad if\ Q_j^{\max} - Q_j^{\min} = 0 \end{cases}, \quad j = 3,4,5 \quad (4)$$

In the above equations, $Q_j^{\max}$ and $Q_j^{\min}$ are the maximal and minimal value of the jth QoS attribute, respectively. Let $\mathbf{M} = \left(M_{i,j}, 1 \le i \le n, 1 \le j \le 5\right)$ be normalizing matrix according to **Q**, where $M_{i,j}$ denotes normalizing value of the *j*th QoS attribute for ASC provider *pasc$_i$*.

2) Weighting phase.

The following formula is used to calculate the overall quality score of ASC provider *pasc$_i$*.

$$qos(pasc_i) = \sum_{j=1}^{4} (w_j \cdot M_{i,j} \cdot fid_{i,j}) + w_5 \cdot M_{i,5} \qquad (5)$$

where, w_j is the weight value of the *j*th QoS attribute, $0 \le w_j \le 1, \sum_{j=1}^{5} w_j = 1$. End users express their preferences regarding QoS by providing values for the weight *w$_j$*.

3.4 Algorithm Analysis

Theorem 1. In worse case, the time complexity of our proposed algorithm is *O*(*n*), where n is the total number of ASC providers.

Proof. In scaling phase, the time complexity of getting normalizing matrix for all attributes is *O*(*kn*), where *k* is the number of attributes (constant), in this work, *k*=5. Then, weighting phase takes *O*(*n*). So, the worst time of algorithm is *O*(*n*).

4. Simulation Results

4.1 A Case Study

A case study of our proposed matchmaking algorithm will be given for explaining the effect of our proposed algorithm. The following experiment method will be used to select the most satisfactory ASC. Suppose the initial providers have 10 ASCs, i.e., $PASC = \{pasc_1, pasc_2, \ldots, pasc_{10}\}$. It means that these ten ASCs can provide the same or closely similar capabilities.

The QoS values of 10 ASCs are generated through the following simulation. Assume that there are 50 similar tasks (or ASC requests). For each request, we randomly delegate it to an ASC, *pasc$_i$*, from *PASC*, and randomly generate QoS attribute values and their fidelity values, denoting QoS value of *pasc$_i$* in this request. In these attributes, *t* is randomly distributed between 70 and 100 ms; *c* is uniformly distributed [10, 100] $; *rel*, *m*, *rep* is randomly generated in [0,1]. The fidelity for the first four attributes is in [0,1]. The QoS values of 10 ASCs for 50 requests are shown in Table 1, where *pn* denotes provided ASC number of *pasc$_i$* for all requests.

Table 1. The QoS values of 10 ASCs for 50 request

pasc	*pn*	QoS attribute value. (*time*, *cost*, *reliability*, *maintainability*, *reputation*)
pasc$_1$	4	(81.364, 94.46, 0.069, 0.614, 0.575), (73.451, 35.37, 0.838, 0.042, 0.187), (71.636, 99.96, 0.304, 0.106, 0.656), (99.294, 43.29, 0.384, 0.493, 0.011)
pasc$_2$	5	(84.663, 23.28, 0.740, 0.456, 0.716), (79.636, 70.50, 0.746, 0.767, 0.616), (91.45, 26.392, 0.427, 0.235, 0.245), (91.586, 92.89, 0.124, 0.888, 0.587), (73.615, 87.73, 0.738, 0.082, 0.804)
pasc$_3$	3	(99.547, 94.69, 0.690, 0.593, 0.930), (90.360, 86.49, 0.422, 0.235, 0.356), (90.825, 22.23, 0.678, 0.784, 0.763)
pasc$_4$	4	(77.087, 90.03, 0.947, 0.393, 0.697), (74.888, 44.23, 0.978, 0.834, 0.624), (80.114, 12.45, 0.502, 0.948, 0.990), (73.533, 35.89, 0.114, 0.325, 0.026)
pasc$_5$	9	(82.862, 32.69, 0.708, 0.043, 0.986), (73.595, 59.18, 0.611, 0.816, 0.219), (97.835, 69.63, 0.150, 0.017, 0.724), (91.110, 11.66, 0.540, 0.945, 0.221), (97.034, 54.32, 0.205, 0.832, 0.493), (93.738, 58.43, 0.659, 0.478, 0.182), (70.386, 36.56, 0.674, 0.919, 0.364), (70.894, 60.31, 0.288, 0.810, 0.637), (98.963, 65.05, 0.412, 0.013, 0.052)
pasc$_6$	4	(73.856, 41.60, 0.411, 0.647, 0.385), (89.247, 25.84, 0.409, 0.542, 0.114), (87.707, 16.07, 0.649, 0.706, 0.112), (88.005, 90.24, 0.991, 0.102, 0.045)
pasc$_7$	5	(91.256, 34.28, 0.566, 0.295, 0.087), (91.973, 61.65, 0.708, 0.659, 0.263), (89.541, 24.46, 0.096, 0.515, 0.247), (89.445, 27.47, 0.342, 0.484, 0.978), (85.628, 31.53, 0.692, 0.551, 0.706)
pasc$_8$	8	(82.750, 60.07, 0.224, 0.790, 0.624), (97.355, 89.42, 0.283, 0.307, 0.567), (71.577, 88.84, 0.643, 0.443, 0.804), (80.630, 51.27, 0.360, 0.116, 0.666), (97.848, 46.98, 0.243, 0.958, 0.529), (82.601, 50.14, 0.681, 0.785, 0.296), (75.847, 14.88, 0.266, 0.178, 0.547), (87.174, 71.81, 0.983, 0.453, 0.635)
pasc$_9$	4	(93.761, 60.72, 0.447, 0.177, 0.424), (75.342, 47.91, 0.730, 0.548, 0.783), (77.398, 70.89, 0.231, 0.295, 0.638), (78.867, 40.35, 0.624, 0.599, 0.457)
pasc$_{10}$	4	(88.814, 56.19, 0.883, 0.563, 0.029), (80.738, 64.78, 0.728, 0.648, 0.872), (75.470, 20.82, 0.637, 0.401, 0.752), (72.877, 20.72, 0.305, 0.750, 0.601)

The QoS values in matrix **Q** and the confidence fidelity matrix of the first four QoS attributes for ASC $pasc_i$ are the mean value of the results produced by its provided software component number, respectively. The QoS values matrix **Q**, confidence fidelity matrix **F**, and normalizing matrix **M** are as follows.

$$\mathbf{Q} = \begin{bmatrix} 81.44, 68.27, 0.40, 0.31, 0.36 \\ 84.19, 60.16, 0.56, 0.49, 0.59 \\ 93.58, 67.80, 0.60, 0.54, 0.68 \\ 76.41, 45.65, 0.64, 0.62, 0.58 \\ 86.27, 49.76, 0.47, 0.54, 0.43 \\ 84.70, 43.44, 0.62, 0.50, 0.16 \\ 89.57, 35.88, 0.48, 0.50, 0.46 \\ 84.47, 59.18, 0.46, 0.50, 0.58 \\ 81.34, 54.97, 0.51, 0.40, 0.58 \\ 79.47, 40.63, 0.64, 0.59, 0.56 \end{bmatrix} \qquad (6)$$

$$\mathbf{F} = \begin{bmatrix} 0.34, 0.15, 0.41, 0.40 \\ 0.68, 0.47, 0.81, 0.36 \\ 0.54, 0.53, 0.41, 0.62 \\ 0.41, 0.76, 0.38, 0.26 \\ 0.51, 0.69, 0.43, 0.65 \\ 0.35, 0.30, 0.46, 0.48 \\ 0.60, 0.18, 0.66, 0.67 \\ 0.51, 0.42, 0.37, 0.46 \\ 0.42, 0.65, 0.50, 0.47 \\ 0.51, 0.65, 0.45, 0.47 \end{bmatrix} \qquad (7)$$

$$\mathbf{M} = \begin{bmatrix} 0.71, 0.00, 0.00, 0.00, 0.37 \\ 0.55, 0.25, 0.65, 0.55, 0.83 \\ 0.00, 0.01, 0.83, 0.72, 1.00 \\ 1.00, 0.70, 0.99, 1.00, 0.81 \\ 0.43, 0.57, 0.31, 0.73, 0.51 \\ 0.52, 0.77, 0.90, 0.60, 0.00 \\ 0.23, 1.00, 0.34, 0.60, 0.56 \\ 0.53, 0.28, 0.26, 0.61, 0.81 \\ 0.71, 0.41, 0.46, 0.29, 0.79 \\ 0.82, 0.85, 1.00, 0.89, 0.77 \end{bmatrix} \qquad (8)$$

Suppose that weight values of each QoS attribute defined by users are: w_1=0.15, w_2=0.2, w_3=0.2, w_4=0.15 and w_5=0.3. The overall quality scores of each ASC provider by using Formula (5) are shown in Table 2.

It is easy to see from Table 1 that pasc10 has maximal QoS score among all ASCs, and this ASC will be selected as optimal one with QoS performance.

The matchmaking result ASCs and their normalizing matrix are shown in Table 3 by using traditional algorithm, Zhang's algorithm [6] and Jiang's algorithm [7], respectively.

In traditional matchmaking algorithm, the first ASC is usually selected as match result, i.e., $pasc_1$. In Zhang's algorithm, the track records concept is equivalent to the fifth QoS attribute in our algorithm, i.e., *reputation*. So, the result QoS scores of each agent are: -1, 1.33, 1.33, 0.67, -1.33, -3.33, -1, 1.33, 0.67, 1. As a result, $pasc_2$, or $pasc_3$, or $pasc_8$ with maximal value (1.33) will be selected. Finally, Jiang's algorithm will be compared with our proposed algorithm, and fidelity factor is not considered by Jiang's algorithm. The overall QoS scores of each ASC by using their algorithm are: 0.218, 0.594, 0.576, 0.880, 0.503, 0.501, 0.562, 0.521, 0.562, 0.858. Obviously, $pasc_4$ (score is 0.880) is optimal ASC.

We can see from normalizing matrix **M** in Table 2, QoS performance (0.82, 0.85, 1.00, 0.89, 0.77) of $pasc_{10}$ gives slightly worse than performance (1.00, 0.70, 0.99, 1.00, 0.81) of $pasc_4$. But, fidelity performance of QoS attributes of pasc10 (0.51, 0.65, 0.45, 0.47) is evidently better than that of $pasc_4$ (0.41, 0.76, 0.38, 0.26). This phenomenon shows that some ASC providers usually claim their higher QoS attribute performance like pasc4, but if the confidence fidelity or users' feeling degree is also considered when selecting ASC, the overall quality or performance for these ASCs is not necessarily optimal. This point accords with the practical situation on Internet, such as e-business, online shopping, in which, people have always compromised selection need software function between quality and users' evaluation. Therefore, our proposed QoS matchmaking algorithm is effective and reasonable.

Table 2. The QoS score for each ASC provider

ASC	overall QoS	ASC	overall QoS
1	0.147	6	0.200
2	0.463	7	0.332
3	0.437	8	0.368
4	0.525	9	0.402
5	0.363	10	0.558

Table 3. Results by using different algorithms

Algorithm	ASC	QoS normalizing value
Traditional Alg.	1	(0.71, 0.00, 0.00, 0.00, 0.37)
Zhang's Alg.	2 3 8	(0.55, 0.25, 0.65, 0.55, 0.83) (0.00, 0.01, 0.83, 0.72, 1.00) (0.53, 0.28, 0.26, 0.61, 0.81)
Jiang's Alg.	4	(1.00, 0.70, 0.99, 1.00, 0.81)
Our proposed Alg.	10	(0.82, 0.85, 1.00, 0.89, 0.77)

4.2 Simulation Experiment

To evaluate the impact of fidelity on the final matchmaking results, we conduct the following simulations and compare the matchmaking results of our algorithm with Jiang's algorithm.

For the sake of simple, suppose that there are 10 initial ASCs. At each experiment, we randomly generate QoS attributes and fidelity values of each ASC according to the following strategy: *t* is randomly distributed between 10 and 100 ms; *c* is uniformly distributed [70, 100] $; *rel*, *m*, *rep* is randomly generated in [0,1]. The fidelity for the first four attributes is in [0,1]. At each experiment point, we record average values $\overline{M}$ of normalizing matrix **M** and average fidelity values $\overline{F}$ of matrix **F** of selected ASC with highest score by using Jiang's algorithm and our algorithm, respectively, i.e.,

$$\overline{M}(i)=\frac{\sum_{j=1}^{5}M_{i,j}}{5},\overline{F}(i)=\frac{\sum_{j=1}^{4}fid_{i,j}}{4} \qquad (9)$$

where, *i* is numbering of ASC with highest QoS performance selected by using Jiang's algorithm or our algorithm. Let $Q_{ei}(i)$ in Formula (10) denote QoS evaluation index for current algorithm, in order to measure final QoS evaluation of selected ASC. The higher the Q_{ei}, the greater the comprehensive quality performance. In which, σ_1 and σ_2 are evaluation weight for QoS attribute and fidelity factor.

$$Q_{ei}(i)=\sigma_1\cdot\overline{M}(i)+\sigma_2\cdot\overline{F}(i) \qquad (10)$$

Simulations are divided into three groups for different weight values of the QoS attributes in Formula (5) considering different users' quality preference.

Group 1: $w_1=w_2=w_3=w_4=w_5=0.2$, denoting unbiasedness;

Group 2: $w_1=w_2=0.35, w_3=w_4=w_5=0.1$, showing preference for *t* and *c*;

Group 3: $w_1=w_2=0.125, \quad w_3=w_4=w_5=0.25$, showing preference for other attributes.

For each group, we run the experiments for different evaluation weights pair, i.e., σ_1 from 0.9 down to 0.1, while σ_2 from 0.1 up to 0.9. We record average value of QoS evaluation index for 100 times in each point. The simulation results are shown in Figure 1, Figure 2, and Figure 3, respectively.

It can be seen from three figures that our algorithm gives higher QoS evaluation index performance than Jiang's algorithm in most cases for every group. Moreover, we observe from all figures that the performance of our algorithm is more stable, and the performance of Jiang's algorithm decreases as σ_1 decreases. This is because that Jiang's algorithm depends too much upon QoS values themselves, whereas QoS values and users' confidence fidelity are better compromised in our algorithm.

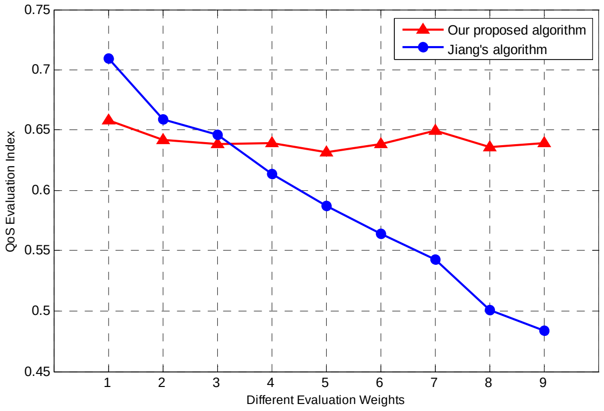


Figure 1. QoS evaluation index vs weights for Group1

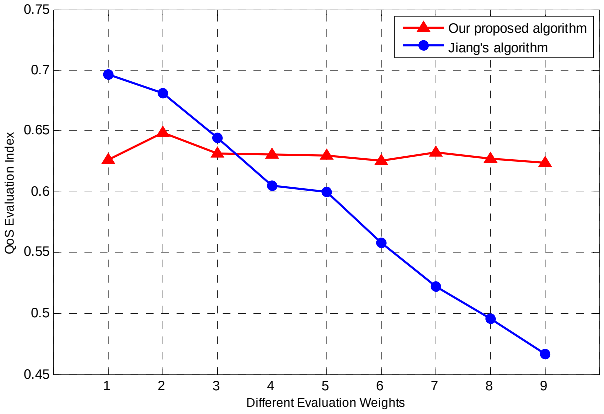


Figure 2. QoS evaluation index vs weights for Group2

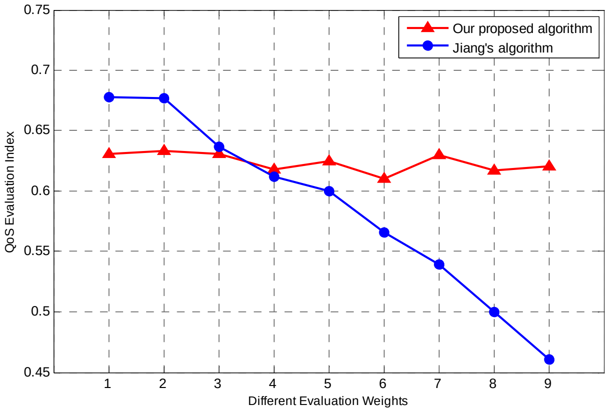


Figure 3. QoS evaluation index vs weights for Group3

To evaluate the effect of our proposed algorithm further, we conduct the following simulations and compare the QoS evaluation index of our algorithm with Jiang's algorithm, Traditional algorithm and Random algorithm. In traditional algorithm, the first ASC is selected as match result, i.e., $pasc_1$. Random algorithm randomly selects ASC. The simulations are also divided into three groups for different weight values of the QoS attributes as above. In each group, we assume there are 50 similar tasks (or ASC requests) and 10 ASC providers. At each experiment point, we record final QoS evaluation of selected ASC by using four algorithms, respectively.

The simulation results for three groups are shown in Figure 4, Figure 5, and Figure 6, respectively.

It can be seen from above three figures that our algorithm gives best QoS performance among compared four algorithms. In almost all experiment request point, the QoS evaluation index of selected optimal ASC by using our algorithm is higher than other three algorithms. The simulation results demonstrate better compromised selection between quality and users' evaluation in our algorithm.

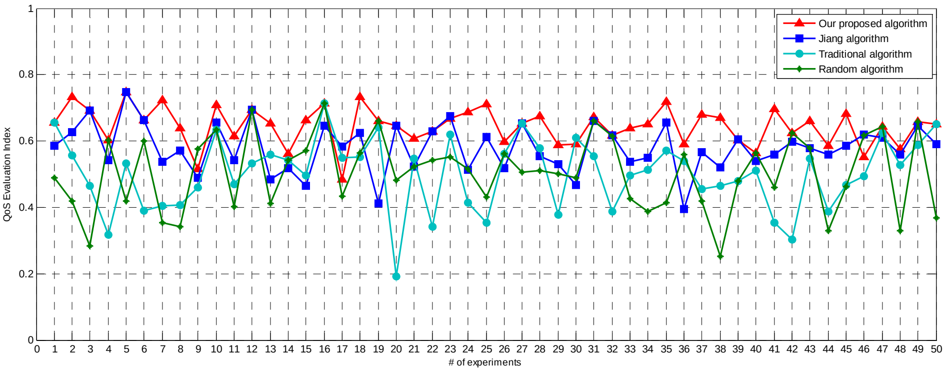


Figure 4. A comparison on QoS evaluation index of selected optimal ASC for Group1

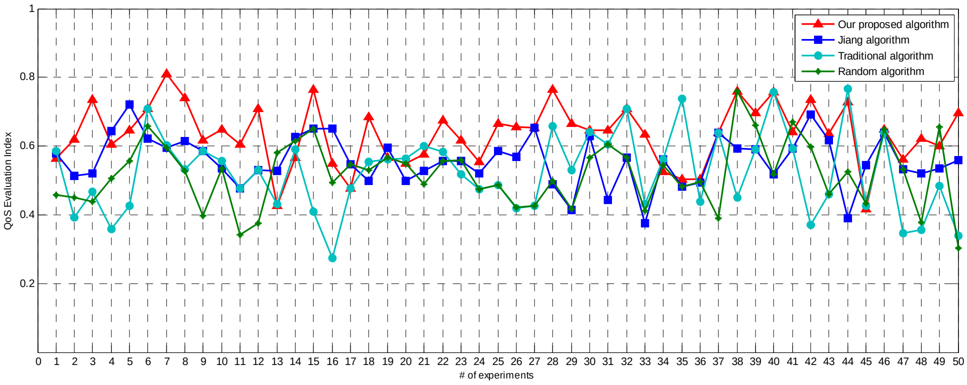


Figure 5. A comparison on QoS evaluation index of selected optimal ASC for Group2

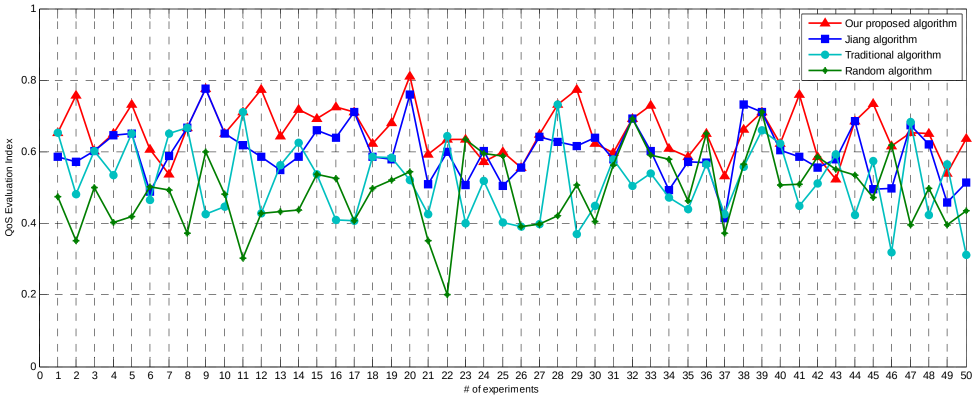


Figure 6. A comparison on QoS evaluation index of selected optimal ASC for Group3

5. Conclusions

In this paper, we discuss the autonomic software component QoS matchmaking problem for autonomic element, which has been taken as one of the most important issue in field of autonomic computing, especially in agent-based autonomic computing system. A QoS model for ASC is built, and on basis of which, an ASC QoS matchmaking algorithm based on fidelity factor is proposed. Main work in this paper has following characteristics:

1) Our proposed QoS model for ASC is simple and effective, and it does not limit type, amount, and value of QoS attributes, which shows better scalability and flexibility.

2) During ASC matchmaking process, we introduce evaluation mechanism of confidence of individual QoS attributes, i.e., fidelity factor for each attribute, which can overcome drawbacks such as subjectiveness and unfairness, and improve the self-configuration capability for autonomic element.

3) Simulation experiments demonstrate the effective and correction of our algorithm for matchmaking, and perform best performance in terms of QoS than other existing algorithms.

4) Simulations show also that our algorithm has better compromise between attribute quality and users' evaluation when selecting ASC. Our proposed algorithm is suitable for many situations on Internet, such as e-business, online shopping, in which, people have always compromised selection need software function between quality and users' evaluation.

6. Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 90718021.

REFERENCES

[1] J. Kephart and D. Chess, "The vision of autonomic computing," IEEE Computer Society, No. 1, pp. 41–59, 2003.

[2] N. R. Jennings, "On agent-based software engineering," Artificial Intelligence, Vol. 177, No. 2, pp. 277–296, 2000.

[3] G. J. Wickler, "Using expressive and flexible action representations to reason about capabilities for intelligent agent cooperation," PhD Thesis, University of Edinburgh, Edinburgh, UK, 1999.

[4] K. Sycara, S. Widoff, M. Klusch, *et al.*, "LARKS: Dynamic matchmaking among heterogenous software agents in cyberspace," Autonomous Agents and MultiAgent Systems, Vol. 5, No. 2, pp. 173–203, 2002.

[5] K. Arisha, S. Kraus, F. Ozcan, *et al.*, "IMPACT: The interactive Maryland platform for agents collaborating together," IEEE Intelligent Systems, Vol. 14, No. 2, pp. 64–72, 1999.

[6] Z. L. Zhang and C. Q, Zhang, "An improvement to matchmaking algorithms for middle agents," in AAMAS'02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, pp. 1340–1347, 2002.

[7] Y. C. Jiang and Z. Z. Shi, "Quality of service driven agent service matchmaking," Mini-Micro Systems, Vol. 26, No. 4, pp. 687–692, 2005.

[8] R. J. Wang, Y. H. Ru, and X. X. Zhu, "Study on reclaiming agent selection models and approaches," in IEEE SOLI'08: Proceedings of IEEE Service Operations and Logistics, and Informatics, Beijing, China, Vol. 1, pp. 1262–1267, 2008.

[9] Q. Ma, H. Wang, Y. Li, *et al.*, "A semantic QoS-aware discovery framework for web services," in IEEE ICWS'08: Proceedings of IEEE International Conference on Web Services, Beijing, China, pp. 129–136, 2008.

[10] R. Hu, J. X. Liu, Z. H. Liao, *et al.*, "A web service matchmaking algorithm based on an extended QoS model," in IEEE ICNSC'08: Proceedings of IEEE International Conference on Networking, Sensing and Control, Sanya, China, pp. 1565–1570, 2008.

[11] L. L. Wei, C. L. Chi, M. C. Kuo, *et al.*, "Consumer-centric QoS-aware selection of web services," Journal of Computer and System Sciences archive, Vol. 74, No. 2, pp. 211–231, 2008.

[12] E. Giallonardo and E. Zimeo, "More semantics in QoS matching," in IEEE SOCA'07: In Proceedings of International Conference on Service-Oriented Computing and Applications, Newport Beach, USA, pp. 163–171, 2007.

[13] D. K. Guo, Y. Ren, H. H. Chen, *et al.*, "A web services selection and ranking model with QoS constraints," Journal of Shanghai JiaoTong University, Vol. 41, No. 6, pp. 870–875, 2007.

[14] K. J. Liang, Q. Liang, and Y. Yang, "The strategy and method of QoS parameter processing for grid service matchmaking," in ICICIC'06: In Proceedings of First International Conference on Innovative Computing, Information and Control, Vol. 1, pp. 381–384, 2006.

J. Software Engineering & Applications, **2009, 2: 111-115**
doi:10.4236/jsea.2009.22016 Published Online July 2009 (www.SciRP.org/journal/jsea)

Scientific Research

An Optimal Algorithm for Prufer Codes*

Xiaodong Wang[1, 2], Lei Wang[3], Yingjie Wu[1]

[1]Department of Computer Science, Fuzhou University, Fuzhou, China; [2]Department of Computer Science, Quanzhou Normal University, Quanzhou, China; [3]College of Computing, Georgia Institute of Technology, Atlanta GA 30332, USA.
Email: wangxd@fzu.edu.cn

Received February 19th, 2009; revised February 23rd, 2009; accepted February 24th, 2009.

ABSTRACT

This paper studies the algorithms for coding and decoding Prufer codes of a labeled tree. The algorithms for coding and decoding Prufer codes of a labeled tree in the literatures require $O(nlogn)$ *time usually. Although there exist linear time algorithms for Prufer-like codes* [*1*,*2*,*3*], *the algorithms utilize the integer sorting algorithms. The special range of the integers to be sorted is utilized to obtain a linear time integer sorting algorithm. The Prufer code problem is reduced to integer sorting. In this paper we consider the Prufer code problem in a different angle and a more direct manner. We start from a naïve algorithm, then improved it gradually and finally we obtain a very practical linear time algorithm. The techniques we used in this paper are of interest in their own right.*

Keywords: *Design of Algorithm, Labeled Trees, Prufer Codes, Integer Sorting*

1. Introduction

Labeled trees are of interest in practical and theoretical areas of computer science. For example, Ethernet has a unique path between terminal devices, thus being a tree: labeling the tree nodes is necessary to uniquely identify each device in the network. An interesting alternative to the usual representations of tree data structures in computer memories is based on coding labeled trees by means of strings of node labels. This representation was first used in the proof of Cayley's theorem [4] to show a one-to-one correspondence between free labeled trees on n nodes and strings of length n-2. In addition to this purely mathematical use, string-based coding of trees has many practical applications. For instance, they make it possible to generate random uniformly distributed trees and random connected graphs: the generation of a random string followed by the use of a fast decoding algorithm is typically more efficient than random tree generation by the addition of edges, since in the latter case one must pay attention not to introduce cycles. In addition, tree codes are employed in genetic algorithms, where chromosomes in the population are represented as strings of integers, and in heuristics for computing minimum spanning trees with additional constraints, e.g., on the number of leaves or on the diameter of the tree itself. Not last, tree codes are used for data compression and for computing the tree and forest volumes of graphs.

Let T be a labeled tree whose nodes are numbered from 0 to n-1. For some vertex v in T, the degree of v, denoted by d[v], is the number of edges incident to v. If d[v]=1, then v is called a leaf. According to Prufer's proof, any sequence of n-2 numbers, each number in {0,1,…,n-1} can determine a unique labeled tree of n nodes. Such a number sequence is the Prufer code of a labeled tree. Algorithm A is the straightforward implementation of Prufer's proof.

```
Algorithm A
Input: A labeled tree T of n nodes as a list of n-1 edges.
Output: c, the Prufer code of T.
Method:
Step 1. B←{0,1,…,n-1}.
Step 2. For i ← 0 to n-3 do
Step 2.1. x ← min{k∈B: k is a leaf }.
Step 2.2. B←B-{x}.
Step 2.3. Remove x and its incident edge (x, y) from T.
Step 2.4. c[i]←y.
Step 3. Return c.
End of Algorithm
```

For example, let the input labeled tree T be the graph depicted in Figure 1. After the Algorithm A terminates, c = [2,4,0,1,3,3] which is the Prufer code of T .

*Supported by Natural Science Foundation of China under Grant No. 60172017 and Natural Science Foundation of Fujian under Grant No. A0510008.

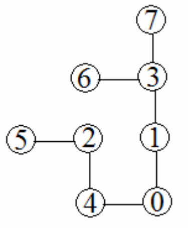


Figure 1. A labelled tree

The algorithms for coding and decoding Prufer codes of a labeled tree in the literatures require $O(nlogn)$ time usually. As stated in [3], although the problem of producing a Prufer code in linear time is an exercise in two books [5, 6], there exists no explicit publication of a solution. In [3] an $O(n)$ time algorithm for generating a Prufer code from a labeled tree was described, but the $O(n)$ time algorithm utilized the integer sorting algorithms. The special range of the integers to be sorted was utilized to obtain a linear time integer sorting algorithm. The Prufer code problem is reduced to integer sorting. In this paper we consider the Prufer code problem in a different angle and present a very practical linear time algorithm directly. The techniques we used in this paper are of interest in their own right.

2. A Linear Time Algorithm for Coding

The most time consuming step in Algorithm A is Step 2. The leaf elimination scheme of the Prufer code implicitly defines a root for the labeled tree T. Actually, it is easy to see that the last element in the code is n-1 which is the root of the labeled tree T. Given a list of n-1 edges, we can build the parent array f and the degree array d of the labeled tree T with only $O(n)$ preprocessing time, which allows checking and updating a node's degree in $O(1)$ time. With this preprocessing, the Step 2.3 of Algorithm A can be implemented in $O(1)$ time. Using a heap, the leaf with the smallest number can be found in $O(nlogn)$ time. Hence, Step 2 takes $O(nlogn)$ time totally. The time complexity of Algorithm A is also $O(nlogn)$.

We can improve the algorithms further. An insight into the problem is that for the heap operations in the Step 2.1 of Algorithm A, there is a special kind of nodes that they are deleted from the heap immediately after they are inserted into the heap. This kind of nodes can be treated easily without heap operation. The remained heap operation can be replaced by a linear scan of the degree array d. Therefore the total time to find x in the Step 2.1 is reduced to $O(n)$.

The linear time algorithm can be presented detailed as follows.

Coding Algorithm:

```
1: index ← x ← min{0≤k<n: d[k]=1}
2: **for** i ← 0 **to** n-3 **do**
3:     y ← f[x]
4:     c[i] ← y
5:     d[y] ← d[y]-1
6:     **if** y<index **and** d[y]=1 **then** x ← y
7:     **else** index ← x ← min{index<k<n: d[k]=1}
```

In the algorithm described above, d[v] is the degree of node v and f[v] is the parent node of the node v. The variable index is a cursor of degree array d. On line 6, when node y becomes a leaf and the label of y is less than the current cursor index, the node y must be the kind of nodes that are deleted from the heap immediately after they are inserted into the heap. In this case, the node y becomes the next node with minimal label. Otherwise, on line 7 the cursor index moves to the next leaf node in the degree array d. The computing time of line 1 and line 7 is $O(n)$, since the cursor index goes through the degree array d from left to right once. The remaining time is clearly $O(n)$, leading to $O(n)$ complexity for the entire algorithm.

3. A Linear Time Algorithm for Decoding

Decoding is to build the tree T corresponding to the given Prufer code c. As far as c is computed, each node label in it represents the parent of a leaf eliminated from T. Hence, in order to reconstruct T , it is sufficient to compute the ordered sequence of labels of the eliminated leaves, say s: for each i ∈{0,1,…,n-1}, the pair (c[i], s[i]) will thus be an edge in the tree.

We first observe that the leaves of T are exactly those nodes that do not appear in the code, as they are not parents of any node. Each internal node, say v, in general may appear in c more than once; each appearance corresponds to the elimination of one of its children, and therefore to decreasing the degree of v by 1. After the rightmost occurrence in the code, v is clearly a leaf and thus becomes a candidate for being eliminated. Therefore, the times of a node v appears in c is exactly the degree of v minus 1. A linear scan of code array c can determine the degree array d.

Using a heap, the leaf with the smallest number can be found in $O(nlogn)$ time, leading an $O(nlogn)$ time decoding algorithm. Like the coding algorithm, we can improve the algorithms further. An insight into the problem is also that for the heap operations of the decoding algorithm, there is a special kind of nodes that they are deleted from the heap immediately after they are inserted into the heap. This kind of nodes can be treated easily without heap operation. The remained heap operation can be replaced by a linear scan of the degree array d.

The linear time algorithm can be presented detailed as follows.

```
Decoding Algorithm:
1: index ← x ← min{0≤k<n: d[k]=1}
2: for i ← 0 to n-2 do
3:     y ← c[i]
4:     add edge (x,y) to T
5:     d[y] ← d[y]-1
6:     if y<index and d[y]=1 then x ← y
7:     else index ← x ← min{index<k<n: d[k]=1}
```

Like the coding algorithm, the time required by the decoding is also $O(n)$.

4. Examples

We use the labeled tree T depicted in Figure 1 as an example to demonstrate the algorithms for coding and decoding Prufer codes described above. The input of the labeled tree T is an edge list {0,1}{0,4}{1,3}{4,2}{3,6}{3,7}{2,5}. There are 8(n) nodes labeled 0,1,2,3,4,5,6,7 and 7(n-1) edges.

It is easy to see that the last element 7 is the root of the labeled tree T. For the given edge list, we can build the parent array f and the degree array d of the labeled tree T by a depth first search with only $O(n)$ time.

With this two arrays, the Step 2.3 of Algorithm A can be implemented in $O(1)$ time.

For our linear time coding algorithm, we first go through the degree array d to find an index such that index=min{0≤k<n: d[k]=1}. It is clear that the index equals 5 for the first time in our example as shown in Table 1.

Table 1. The parent array f and the degree array d of T

	i							
	0	1	2	3	4	5	6	7
f[i]	1	3	4	7	0	2	3	-1
d[i]	2	2	2	3	2	1	1	1
						↑ index		

Table 2. After edge {2,5} is deleted

	i							
	0	1	2	3	4	[5]	6	7
f[i]	1	3	4	7	0	2	3	-1
d[i]	2	2	1	3	2	1	1	1
						↑ index		

Table 3. After edge {4,2} is deleted

	i							
	0	1	[2]	3	4	[5]	6	7
f[i]	1	3	4	7	0	2	3	-1
d[i]	2	2	1	3	1	1	1	1
						↑ index		

Table 4. After the edges {0,4}, {0,1} and {1,3} are deleted

	i							
	[0]	[1]	[2]	3	[4]	[5]	6	7
f[i]	1	3	4	7	0	2	3	-1
d[i]	1	1	1	2	1	1	1	1
						↑ index		

Table 5. After the edge {3,6} is deleted

	i							
	[0]	[1]	[2]	3	[4]	[5]	[6]	7
f[i]	1	3	4	7	0	2	3	-1
d[i]	1	1	1	1	1	1	1	1
							↑ index	

f[index]=f[5]=2 is the first value of the Prufer code c. The edge {2,5} is deleted and d[2] is decreased by 1 on line 5 of the coding algorithm. The status of the tree T now becomes Table 2.

Follows the coding algorithm on line 6 the next node to be deleted is 2. The father node of 2 is node 4, the next value of the Prufer code c. The edge {4,2} is deleted and d[4] is decreased by 1 on line 5 of the coding algorithm. The status of the tree T now becomes Table 3.

Similarly, in the next three steps we obtain the next three values 0,1 and 3 of the Prufer code c. The edges {0,4}, {0,1} and {1,3} are deleted accordingly. The Prufer code we have obtained up to now is (2,4,0,1,3). The status of the tree T now becomes Table 4.

Look at the Table 4, the next node to be deleted is 6 which is determined on line 7 of our coding algorithm. We do not scan the degree array d from the beginning. We scan the degree array d from index (5) to right and the index is moved to 6. This is a key point to see the algorithm running in linear time.

The father node of 6 is node 3, the next value of the Prufer code c. The edge {3,6} is deleted and d[3] is decreased by 1 on line 5 of the coding algorithm. The status of the tree T now becomes Table 5.

Table 6. The initio status of the decoding algorithm

	i							
	0	1	2	3	4	5	6	7
c[i]	2	4	0	1	3	3	7	-1
d[i]	2	2	2	3	2	1	1	1
						↑ index		

Table 7. After edge {2,5} is added

	i							
	0	1	2	3	4	[5]	6	7
c[i]	2	4	0	1	3	3	7	-1
d[i]	2	2	1	3	2	1	1	1
						↑ index		

Table 8. After edge {2,4} is added

	i							
	0	1	[2]	3	4	[5]	6	7
c[i]	2	4	0	1	3	3	7	-1
d[i]	2	2	1	3	1	1	1	1
						↑ index		

We now have the n-2 values of the Prufer code (2,4,0,1,3,3). The algorithm terminates.

Using the same labeled tree T depicted in Figure 1 as an example we can demonstrate the algorithm for decoding Prufer codes as follows.

The input of the decoding algorithm is the Prufer code c of the tree T. In our example the Prufer code of the tree T is (2,4,0,1,3,3). We fist noted that the times of a node v appears in the Prufer code c is exactly the degree of v minus 1. A linear scan of code array c can determine the degree array d as shown in Table 6.

For our linear time decoding algorithm, we first go through the degree array d to find an index such that index=min{0≤k<n:d[k]=1}. It is clear that the index equals 5 for the first time in our example as shown in Table 6 which is the first leaf node deleted in the coding algorithm.

c[0]=2 is the other node label of the edge to be added. The edge {2,5} is added and d[2] is decreased by 1 on line 5 of the decoding algorithm. The status of the tree T now becomes Table 7.

Follows the decoding algorithm on line 6 the next node to be added is 2. c[1]=4 is the other node label of the edge to be added. The edge {2,4} is added and d[4] is decreased by 1 on line 5 of the decoding algorithm. The status of the tree T now becomes Table 8.

Similarly, in the next three steps we obtain the next three edges {0,4}, {0,1} and {1,3}. The status of the tree T now becomes Table 9.

Look at the Table 9. The next node to be added is 6 which is determined on line 7 of our decoding algorithm. We do not scan the degree array d from the beginning. We scan the degree array d from index (5) to right and the index is moved to 6. This is a key point to see the decoding algorithm running in linear time.

The other node label of the edge to be added is c[5]=3. The edge {3,6} is added and d[3] is decreased by 1 on line 5 of the decoding algorithm. The status of the tree T now becomes Table 10.

Follows the decoding algorithm on line 6 the next node to be added is 3. c[6]=7 is the other node label of the edge to be added. The edge {3,7} is added. We now have the n-1 edges of the tree T. The decoding algorithm terminates.

5. Conclusions

Prufer codes for labeled trees have many practical applications. The algorithms for coding and decoding Prufer codes of a labeled tree are of interest in practical and theoretical areas of computer science. The existing linear time algorithms for Prufer-like codes utilized the integer sorting algorithms. The special range of the integers to be sorted is utilized to obtain a linear time integer sorting algorithm. The optimal algorithms for coding and decoding Prufer codes presented in this paper are very practical linear time algorithms. The techniques used in these algorithms are of interest in their own right.

Table 9. After the edges {0,4}, {0,1} and {1,3} are added

	i							
	[0]	[1]	[2]	3	[4]	[5]	6	7
c[i]	2	4	0	1	3	3	7	-1
d[i]	1	1	1	2	1	1	1	1
						↑ index		

Table 10. After the edge {3,6} is added

	i							
	[0]	[1]	[2]	3	[4]	[5]	[6]	7
c[i]	2	4	0	1	3	3	7	-1
d[i]	1	1	1	1	1	1	1	1
						↑ index		

6. Acknowledgments

This work was supported by Natural Science Foundation of China under Grant No. 60172017 and Natural Science Foundation of Fujian under Grant No. A0510008.

REFERENCES

[1] S. Caminiti, I. Finocchi, and R. Petreschi, "A unified approach to coding labeled trees," in Proceedings of the 6th Latin American Symposium on Theoretical Informatics (LATIN'04), LNCS 2976, pp. 339−348, 2004.

[2] S. Caminiti, I. Finocchi, and R. Petreschi, "On coding labeled trees," To appear on Theoretical Computer Science, 2006.

[3] H. C. Chen and Y. L. Wang, "An efficient algorithm for generating Prufer codes from labeled trees," Theory of Computing Systems, Vol. 33, pp. 97–105, 2000.

[4] A. Cayley, "A theorem on trees," Quarterly Journal of Mathematics, Vol. 23, pp. 376–378, 1889.

[5] L. Devroye, "Non-uniform random variate generation," Springer-Verlag, New York, Exercise 2, pp. 666, 1986.

[6] A. Nijenhuis and H. S. Wilf, "Combinatorial algorithms for computers and calculators," Second Editon, Academic Press, New York, Exercise 46, pp. 293, 1978.

J. Software Engineering & Applications, **2009, 2: 116-121**
doi:10.4236/jsea.2009.22017 Published Online July 2009 (www.SciRP.org/journal/jsea)

Model Interpretation Development: Analysis Design of Automatic Control System

Guohua Wu, Wenning Liu, Qiuhua Zheng, Zhen Zhang

Hangzhou Dianzi University, Hangzhou, China.
Email: wugh@hdu.edu.cn

Received February 16th, 2009; revised April 10th, 2009; accepted April 14th, 2009.

ABSTRACT

Currently the development of automatic control system is mainly based on manual design. This has made the development process complicated and has made it difficult to guarantee system requirement. This paper presents a Model interpretation development architecture built on meta-models and model interpretation. In this modeling and developing process, different meta-models or domain models may be constructed in terms of various system requirements. Interpreters are used to transform the meta-model into relevant domain model and generate some other formats from domain models, typically with different semantic domains. An interpretation extension interface is introduced, which can be accelerated to develop the model interpreter. This development architecture can improve system reusability and enhance development efficiency. Finally, an example is introduced to explain the advantage of method.

Keywords: *Meta-Model, Model Interpretation, Model-Based Design*

1. Introduction

Developing embedded automatic control software is a complicated and time-consuming task nowadays. The growing complication of building and validating software is a challenge for developers of the embedded software application. The traditional development method based on manual coding is time consuming. Software developers and domain engineers are often from different fields, and the communication between them is difficult. For acquiring relevant requirement and function, software developer must learn special domain knowledge. However, the requirement may be change during the development process, which will prolong the development cycle. In addition, the development of software and hardware is separated, and it is hard to guarantee real-time interaction and verification at target platforms. In order to develop Automatic control software effectively, we must reach two goals: firstly, entire software is developed by different domain developer. Secondly, software reusability of automatic control software should be considered.

In the order to solve the problems caused by the traditional approach, the paper presents a method based on model interpretation development for the automatic control system. The meta-model created by modeling tool is considered as domain-specific modeling language through meta-model interpretation. It is a generic model that can be used to customize application models. Then, it can analyze the domain model and configure parameters and required information. Model interpreter generates output artifacts from domain models, such as code and configurable files. Using theses artifacts together, it can implement embedded automatic control software application development. In comparison to the traditional approach, it will shorten the cycle of development, reduce the cost of product development and improve the reusability of software components.

2. Model Interpretation Developments for Automatic Control Systems

2.1 Overview of Development Architecture

Model Driven Architecture (MDA) [1] is a model-based software modeling approach introduced by Object Manager Group (OMG). MDA contains two different models: Platform-Independent Model (PIM) and Platform-Specific Model (PSM), and it emphasizes the mapping relationship between them. Model Integrated Computing (MIC) [2] is also a modeling framework for the embedded system, which is based on models and generation. But this method is not suitable for certain domains, especially for automatic control systems.

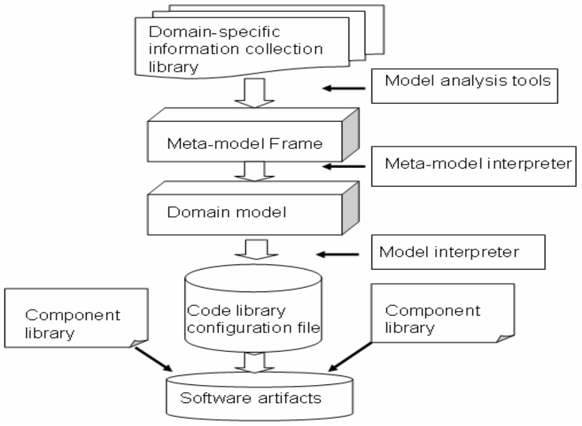


Figure 1. Model interpretation development architecture

Model interpretation development is a technology that is well conformable for the rapid design and implementation of a system. Model interpretation development employs domain-specific models to represent system software, its environment, and their relationship. Model interpretation development have two cores: First, it extends the scope and purpose of modeling in system, and improves the functions of model interpretation throughout the entire development of system process, i.e., design, verification, emulation, code generation. As a result, it can develop the system driven by model. Second, it adopts the domain-oriented concept to analyze commonness and variety character of the domain system, establishing the meta-model and the domain model towards this domain so that it can support uniform development process. This method provides three functions. 1) It provides the development idea and tools for domain-specific and creating modeling language familiarly for domain experts. It will be customized quickly based on actual requirement for experts. 2) It separates the module applicable to the embedded automatic control system from multi-aspects design hierarchy. 3) It can relate model paradigm to related format models automatically in hardware-independent platforms, thus shortening the period of software development.

The advantage to using model interpretation development is the ability to analyze and design a complex system or software at a high level of abstraction and to construct the extension interface to facilitate the design of the model interpretation. Figure 1 shows the architecture of development. The developer collects the related domain information from information libraries and constructs the meta-model of needed domain using existing modeling tools. This is the first step. The developer customizes the paradigm of meta-model, the paradigm defines the entities and associations in the given domain. Related models are assembled into classes. Each class has its own model hierarchy. Paradigm has a fixed set of class. Then, they can design the domain model employing meta-model interpreter, transform these models into essential code, configurable files and verification information documentation using model generator engines, i.e., model interpreter. For example, this model generator engine can translate GME [3] models into UPPAAL [4] models and these models can be simulated in UPPAAL model, Finally, output information documentation or artifacts deploy some component libraries to synthesize the software or documents customer want to use conveniently.

2.2 Creation of Meta-Model

Developer can construct any of meta-model in a domain employing the meta-modeling language, e.g., meta-model in vehicle, or navigation. Meta-model layer is the foundational and core layer in the implementation of domain-specific environment. Figure 2 shows the meta-model paradigm created by GME [5]. In this figure, the root model is the Compound <<Model>>. This compound contains ProcessingNode <<Model>>, which means that it has all of the properties from the ProcessingNode. In the meantime, the Compound and the Primitive are inherited from ProcessingNode which contains Signal <<Atom>>. The signal genrates two types of node: Inputsignal <<Atom>> and Outputsignal <<Atom>>. The two types of node represent input and output of signal respectively. The input and output are connected by Transmission <<Connection>>.

This figure also presents a relationship of state transition. The State <<Model>> has its own models and contains two types of node: Initstate <<Atom>> and FinalState <<Atom>>, which are represented by InitState and FinalState respectively and are associates with each other by Transition <<Connention>>. There is a special node called StateNode <<Model>>. It is inherited from Primitive and State and has all of the properties from the ProcessingNode and State. StateNode has a SignalMap <<Connection>>. This means that Signal can connect with State. This diagram denotes that state can transform into another state by invoking the correlative signal of sensor in the real environment.

2.3 Domain Model Design

The designer of domain model should be familiar with this domain and it’s essential consideration that developers communicate with domain experts in time.

Domain models are used to represent the strategy to be implemented by the run-time system. Many different types of models may be used to represent different automatic control systems. Initially, we have chosen to focus on an improved hierarchical model structure notation and to give domain models for a system as understandable components with well-defined interfaces. The “signal” that forms these components permits data to be exchanged between components. The signal flow defines

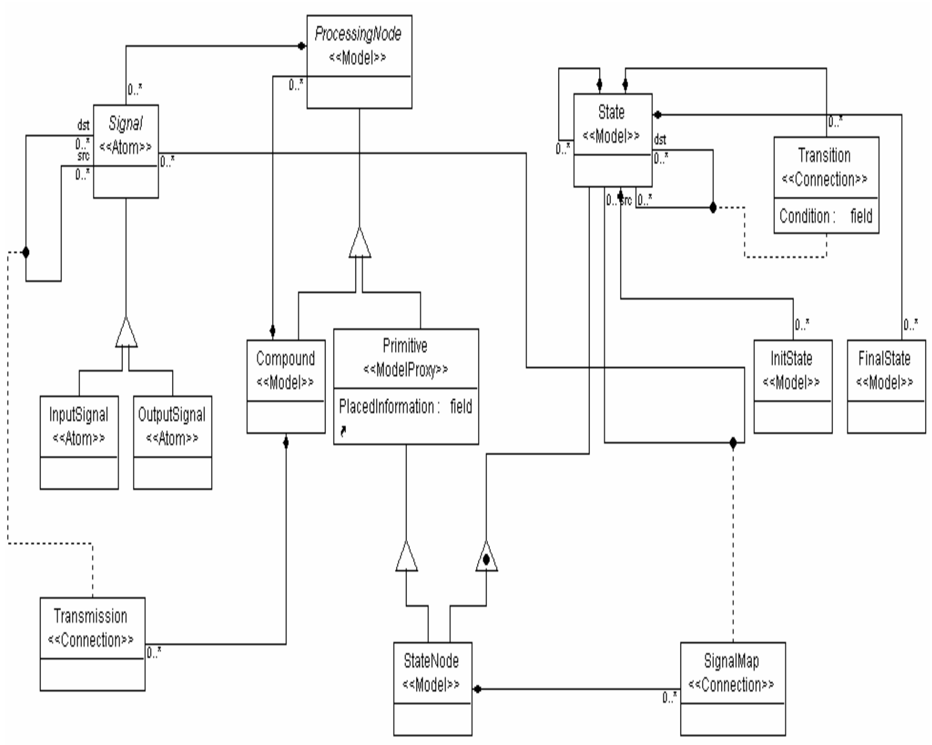


Figure 2. Screenshot of the resource meta-model created using GME

the order of processing for an application. Each component in the signal and state flow graph receives data from other components, performs some data exchange, and then outputs new data to other system components.

2.4 Model Interpretation

Model interpreters [6] i.e. model generators play a crucial key in model interpretation development. They act on the transformation engine in software development.

Model interpreters are used to transform the information captured in the models into the artifacts required by the chosen analysis tools or run-time system. Model generators have two functions:

1) To transform models into the input file and information of analysis tools or to transform the analysis results back into the modeling language.

2) To transform models into needed code, static data structure, and configuration files, etc. These can form the executable software system running on some integration platforms (OS, component integration framework, etc.). Figure 3 shows the relationship between models and generators.

The model interpretation process is somewhat similar to the back-end of compilers. The models capture information in a structured form, typically in the form of hierarchically organized objects. This graph of objects may be traversed, perhaps transformed, and used to generated output. While the process is very easy to describe in general, it is not obvious how it can be implemented [7].

The model interpreter is usually implemented in three ways [8].

1) Direct implementation. The generator acts on the mapping between abstract syntax of input model and abstract syntax of the target model. The structure of input tree is the data structure that corresponds to the output tree of compilers, and the structure of target tree corresponds to the output tree of compilers. Naturally, in the simplest of cases, the output artifacts can be directly generated from the input tree.

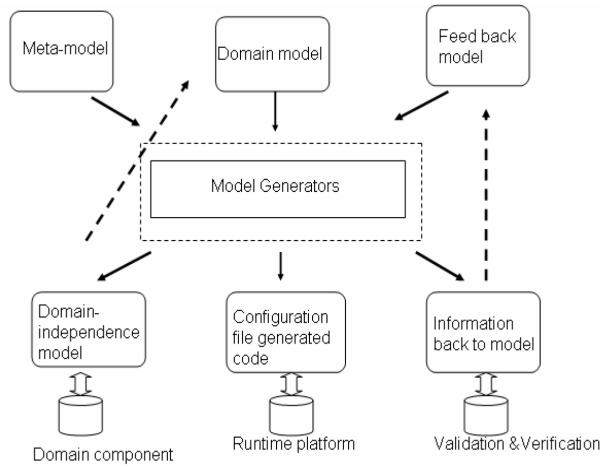


Figure 3. Relationship between models and generators

2) Visitor –based approach

A visitor object implements the actions to be performed all sorts of nodes in the input tree, while the tree nodes receive the visitor object and call the appropriate, special node operation on it. The visitor pattern allows the concise and scalable implementation of generators, both for the translation and the output phase. Generators using this approach are very simple, understandable and have successfully applied in different modeling environment.

3) Meta-generators. A better technique would allow the mathematically precise definition of the generator's internal operation and the code generation of the generator from correlation model.

Although the pattern-based approach is better than direct implementation, essentially speaking, it belongs to the category of direct implementation. Based on meta-generator approach, the mechanism of internal interpreter can be abstracted, which can be more structured in development. Common parts of interpreter can be generated automatically by meta-generator; developers can focus on the mapping between input and output data.

A generator can be created in terms of a graph traversal which describes in what order the nodes of the input tree should be visited. Additionally, a set of transformation rules should be added. Acquiring data structure and traversing input tree of models is a significant process for the development of model interpreter.

In order to be more effective, we use interpretation extension interface (IEI) to improve the effectiveness of developing interpreter. Interpretation extension interface extracts most of the functions. Configuration and analysis of the functions are time-consuming. The interface provides the extensibility to conform with both domain-specific modeling simulations or analyses and domain-independent traversal, code framework generation or graph transformation [9,10]. Such a structure based on the method of meta-generator must encapsulate functions or algorithms that are useful in a wide variety of universal environments. However, an IEI is not just a set of functions. Rather, in the way of predefined pattern, it is a flexible component that can be extended and enhanced based on design requirement.

Common basic parts of domain-independent structures depend on which types the IEI can operate or select. Domain-specific functions or detailed implementations are completed by interpretation extension interface. An IEI can be constructed that utilizes unified interface specifications.

Interpretation extension interface enables a designer to implement the skeleton of domain-specific model interpreter. The different functions of implementation are abstracted by a categorized common interface. The input models required by a given analytic domain can be determined by selecting the type of the implementation of an IEI and extracted the information (e.g., traversal, code framework, graph transformation) from the model using the APIs which provides by the modeling tools. The encapsulated algorithms or functions are invoked at specific times during the interpretation process, under such a condition that the traversal information, generated code will be invocated when various assigned events occur during an actual interpreter run. Figure 4 shows high-level design of the interpretation extension interface.

IEI can provide extended interface to the developer of interpreter so that they can be convenient to design the appropriate interpretation for software development or model analysis. Furthermore, GME offers interface based on the component object model (COM) technology, which can facilitate for developer dependent on the IEI to design and construct model interpreter rapidly. Figure 5 shows development of GME interpreter hierarchy.

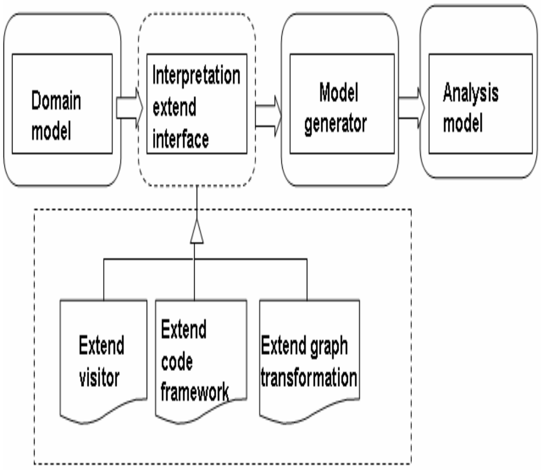


Figure 4. High-level design of the interpretation extension interface

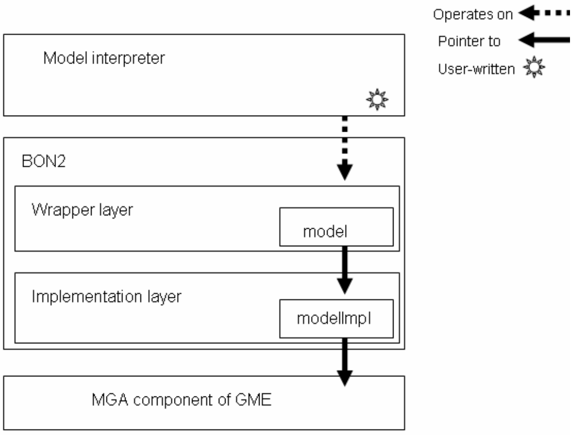


Figure 5. Visiting GME interpreter interface hierarchy

3. Case Study

Using the approach discussed above, a simple example of Washer model graph illustrates the interpretation capability of model design. Figure 6 is a screen shot of Washer model graph. It illustrates the domain models built in the graphical user interface of GME.

According to the meta-model defined above, an implementation model has been created by GME. These implementation models define and design the structure of system components and the associations between them.

The example contains six states and signals. The signals are a source actor denoted as “sensor” and the states are a sink denoted as “Actuator”.

Each signal model will send one signal to inform the state model to change its action so that Washer can switch to the next step to execute the operation. Figure 7 illustrates a sample output code after executing the interpretation extension interface in the model paradigm about this example model. Model interpreter will parse the required parameters or information from the resource model of component and configure the simulator [11]. The C files specified as part of the application model, also can be compiled with the compiler of platform-independent. The simulation results can be verified degree of satisfaction of the performance.

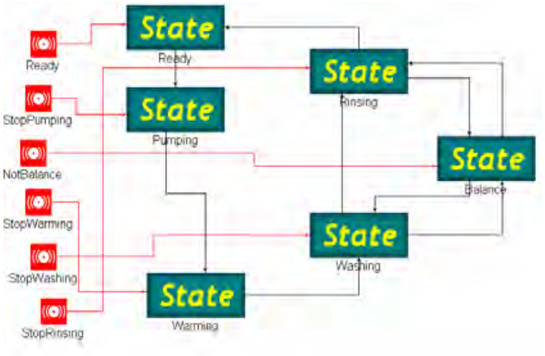
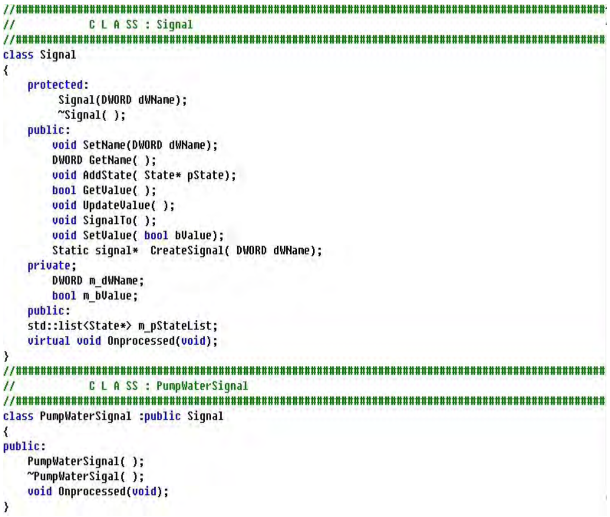


Figure 6. Screenshot of Washer model graph application



```
//##################################################################################################
//            C L A SS : Signal
//##################################################################################################
class Signal
{
    protected:
         Signal(DWORD dWName);
         ~Signal( );
    public:
        void SetName(DWORD dWName);
        DWORD GetName( );
        void AddState( State* pState);
        bool GetValue( );
        void UpdateValue( );
        void SignalTo( );
        void SetValue( bool bValue);
        Static signal*  CreateSignal( DWORD dWName);
    private;
        DWORD m_dWName;
        bool m_bValue;
    public:
    std::list<State*> m_pStateList;
    virtual void Onprocessed(void);
}
//##################################################################################################
//            C L A SS : PumpWaterSignal
//##################################################################################################
class PumpWaterSignal :public Signal
{
public:
    PumpWaterSignal( );
    ~PumpWaterSigal( );
    void Onprocessed(void);
}
```

Figure 7. Screenshot of output code from generator

The configuration file also can be output, and then the generated code and configuration file are read and parsed by the runtime platform. The runtime platform configures system requirements and other data structures based on this configuration file and code file. For the sake of simplicity, executed functions are given in code in Figure 7.

Finally, the generated C code would be checked by the compiler during compilation on the runtime platform. The generated code is compiled, linked and synthesized. The developer will develop software artifacts by using the C code.

4. Conclusions and Future Work

This paper presents model interpretation development architecture for the automatic control system and uses a tool chain to support the convenient and rapid development. The case study has shown that this method can improve system reusability, reduce the cost of products, and gain the expected effect.

In the future, we should do the following work. 1) Domain meta-model needs to be more effective. We should design more understanding models in this domain, communicating with domain experts for accumulating experience. 2) Model interpretation extension interface needs to enhance the functionality of resources traversal, code generation and model transformation. We will provide broad coverage of the analysis techniques present in the software architecture. This is the core work in our research. 3) A research tool kit should be emulated, verified, tested and analyzed in the development of automatic control system so that it can integrate more related tools to support automatic assemblage. This is an onerous task.

5. Acknowledgements

This work is supported by National Fundamental Research Program of China under Grant NO.A1420080190.

REFERENCES

[1] Object Management Group, “Model driven architecture: A technical perspective,” 2001.

[2] J. Davis, “Model integrated computing: A framework for creating domain specific design environments,” The 6th World Multi-Conference on Systemic, Cybernetics and Informatics, Orlando, pp. 14–18, 2002.

[3] A. Ledeczi, *et al*., "The generic modeling environment," Workshop on Intelligent Signal Processing, Budapest, Hungary, May 17, 2001.

[4] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," International Journal on Software Tools for Technology Transfer.

[5] GME 2000 User's Manual, available from http://www.isis.vanderbilt.edu.

[6] G. Edwards, N. Medvidovic, *et al*., "Model interpreter frameworks: A foundation for the analysis of domain-specific software architectures," [J] Journal of Universal Computer Science, Vol. 14, No. 8, pp. 1182–1206, 2008.

[7] G. Karsai, "Structured specification of model interpreters," IEEE Conference and Workshop on Engineering of Computer-Based Systems, Proceedings ECBS'99, 1999.

[8] G. Karsai, J. Sztipanovits, A. Ledeczi, *et al*., "Model-integrated development of embedded software [J]," Proceedings of the IEEE, 2003.

[9] K. Chen, J. Sztipanovits, S. Abdelwahed, and E. Jackson, "Semantic anchoring with model transformations," pp. 115–129, 2005.

[10] M. Wimmer, M. Strommer, *et al*., "Towards model transformation generation by-example," in Proceedings of HICSS-40 Hawaii International Conference on System Sciences, Hawaii, USA, 2007.

[11] A. Agrawal *et al*., "MILAN: A model based integrated simulation framework for design of embedded systems," ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems, Snowbird, Utah, June 2001.

***J. Software Engineering & Applications*, 2009, 2: 122-126**
doi:10.4236/jsea.2009.22018 Published Online July 2009 (www.SciRP.org/journal/jsea)

Transliterated Word Identification and Application to Query Translation Mining

Jing Zhang[1], Lei Guo[2], Meiling Zhou[2], Jianmin Yao[2]

[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China; [2]School of Computer Science and Technology, Soochow University, Suzhou, China.
Email: zhjing@scut.edu.cn, jyao@suda.edu.cn

Received February 12th, 2009; revised April 28th, 2009; accepted May 4th, 2009.

ABSTRACT

Query translation mining is a key technique in cross-language information retrieval and machine translation knowledge acquisition. For better performance, the queries are classified into transliterated words and non-transliterated words based on transliterated word identification model, and are further channeled to different mining processes. This paper is a pilot study on query classification for better translation mining performance, which is based on supervised classification and linguistic heuristics. The person name identification gets a precision of over 97%. Transliterated word translation mining shows satisfactory performance.

Keywords: *Transliteration, Query Classification, Supervised Learning, Translation Mining*

1. Introduction

For both cross-language information retrieval and machine translation knowledge acquisition, translation mining for out-of-vocabulary words is an important module which can help translate named-entities, organization and location names, book and movie titles, technical terms, and newly-coined words that are not included in the dictionary.

The web is a rich mineral for translation mining based on co-occurrence statistics. Related researches [1,2,3] use web search engines to get the web snippets for translation and query co-occurrence. Researches [4,5] make study on how to obtain the effective web pages that include both the query and the translation. Besides the co-occurrence statistics, natural language processing techniques such as word alignment is also utilized in recent research work. This paper is a further endeavor to query classification for higher translation mining accuracy.

In past researches, all query terms go through the same process for translation mining, which omits the difference between transliterated words and non-transliterated words. But in fact, general words such as “翻译模型” (translation model) and transliterated words such as “巴拉克 • 胡赛因 • 奥巴马” (Barack Hussein Obama) should follow some different mining channels so that the results can achieve higher accuracy. This paper makes an endeavor to an automatic query classification method so that transliterated words can be separated from general query terms.

In present researches on query translation mining, transliterated words are not separated from non-transliterated words. This method leads to a compromised solution in the modeling. [1,6] propose a solution applicable to both transliterated and non-transliterated words, which improved the performance on the whole, but lowered the performance for specialized words.

In transliteration study, [6,7,8] select the most probable translation from series of candidate transliterations. Other transliteration models include [9,10]. Because it’s not feasible to identify whether the word is transliterated or not, the work cannot be combined in natural language processing systems up to now.

A method is proposed in this paper to decide whether the query word is a transliterated word or not, which utilizes a unigram-based transliteration statistics plus some heuristic rules. The experiment shows a precision over 97%.

The next section of the paper describes the unigram-based transliteration identification modeling based on a supervised-learning process. The second section describes the experimental results and analysis of the solution. The last section concludes the method and describes future works in the field.

2. A Unigram-Based Transliteration Identification Model

From observation, we can see that the wording of the Chinese transliterated words follows some academic traditions and has good characteristics for statistical study and supervised learning. Two models are proposed from two perspectives and then integrated for better performance on transliterated word identification.

2.1 Transliteration Features of Chinese Characters

Followed are some related concepts for further modeling of the transliteration identification.

Definition 1. Transliteration characters. The Chinese characters that are used in transliterations are called transliteration characters. A national standard for name transliteration exists in China as in [11], which specifies the rules for character selection in name transliteration.

For example, the character “斯” is a transliteration character. While another character, “撕”, is not seen in transliterated words, so it’s not a transliteration character.

Definition 2. Transliteration probability of a Chinese character. The transliteration probability of a character refers to the probability that the character occurs in transliteration words. The definition is as follows:

$$TP(c) = \frac{the\ number\ of\ c\ in\ transliterated\ words}{the\ number\ of\ all\ the\ transliterated\ character} \tag{1}$$

where, c is a Chinese character. For example, the character “斯” has been seen in transliterated words n_1 times in the running corpus, and the number of transliterated characters is n_2 in the corpus. So $TP(\text{斯})=\frac{n_1}{n_2}$. We use the person name corpus as the training corpus in our experiment.

Based on the above definitions, two models to identify the transliteration words are proposed in the next section.

2.2 Models and the Algorithm for Transliteration Word Identification

Model 1: Counting the number of transliteration characters. Compare the transliterated words like “斯坦福” (Stanford), “克林顿” (Clinton), with non-transliterated words like “星期天” (Sunday), “出版社” (press), we can see that some Chinese characters are frequently seen in transliterated words, e.g. “克” and “斯”, which are called transliteration characters. The word “斯坦福” (Stanford) contains 3 transliteration characters, while the word “星期天” (Sunday) contains no transliteration characters. Based on this observation, the first transliteration word identification model is proposed based on the percentage of transliteration characters in the word, as follows:

$$\text{PTC}(w)=\frac{\text{number of transliteration characters in the word } w}{\text{number of characters in the word } w} \tag{2}$$

Based on supervised learning decision, when the PTC (w) is above a threshold, we can decide that the word is a transliterated word. An empirical threshold θ_1 in the following experiment is set at 50.001%.

Model 2: Averaging the transliteration probability of characters. A second model is built based on the average of transliteration probability of all characters in the given Chinese word, defined as follows:

$$A\text{TC}(w)=\frac{\sum_{for\ all\ characters\ c_i\ in\ w} TP(c_i)}{number\ of\ characters\ in\ w} \tag{3}$$

The definition of $TP(c_i)$ is given in Equation (1). Similarly, there exists a threshold θ_2 for $ATC(w)$ to decide whether the word is a transliteration or not. The value of the threshold θ_2 is decided by experiments through training, which is $3e^{-5}$ in the following experiment.

Linguistic heuristics: According to observation, some heuristics are applied to enhance the overall performance on basis of combing the two models. The first heuristic is that if the word contains only 1 or 2 characters, Model 1 should be used. For example, the word “卓娅” contains 2 characters, and we should used model.

If the word contains more than 2 characters, use Model 1 first. If Model 1 returns true, then w is a transliterated word. If Model 1 returns false, use Model 2 to make a second identification.

Based on the two models above and the observations, the transliteration word identification process is proposed as follows:

```
PROCEDURE: Transliteration word identification
BEGIN PROCEDURE
IF the length of the word w is less than 2
    THEN use model 1,
         IF $PTC(w) > \theta_1$, RETURN TRUE
         ELSE RETURN FALSE
ELSE IF the length of word w is more than 2
    THEN use model 1 first,
         IF $PTC(w) > \theta_1$, RETURN TRUE
         ELSE IF $PTC(w) > \theta_1$,
              THEN use model 2 secondly
                   IF $ATC(w) > \theta_2$, RETURN TRUE
                   ELSE RETURN FALSE.
END PROCEDURE
```

3. Query Translation Mining from Search Engine Snippets

Taking translation mining system flow is as follows: First, the source Chinese query is sent to a search engine to retrieve Chinese documents. Second, the relevant topic words, which are hint words for the subject or topic of the query, are extracted from the returned snippets. Third, the source query together with the translations of the topic words are sent to search engine again to obtain relevant bilingual web snippets. The next step is extracting valid terms from the returned bilingual snippets and the final step is ranking the candidate terms to get the final translation(s).

Briefly, this system consists of three main parts:

1) Bilingual snippets collection. Retrieve the bilingual snippets that contain the source term in Chinese and translation in English from a search engine and download as the bilingual resource. Effective techniques to obtain higher relevant snippets are the basis of translation extraction.

2) Candidate term extraction. Extract valid lexical units and multi-lexical units from the returned snippet set in Step 1. It is not a straightforward work. Firstly, Chinese texts have no spaces between characters and one snippet with 2 or 3 sentences usually is relatively small compared to authoritative corpora. Secondly, the snippets generally contain OOV terms. Thus term extraction from returned snippets needs specific technical study.

3) Appropriate translation selection. Rank and sort out candidate translations generated in Step 2. The candidate set may be very large, so the most proper translations should be selected from this set.

3.1 Transliteration Model

Proper names, such as person names, place names, etc., compose a large part of OOV terms. Many proper names are translated based on phonetic pronunciations, which we call transliteration. There has been some related work on extracting term translations based on transliteration techniques as in [6,12,13,14,15]. They converted an English name into a phonetic representation, and then transformed the representation sequence into Chinese pin-yin (phonetic sequence) symbols. At last, they translated the pin-yin sequence to a Chinese character sequence. Our transliteration model differs in two aspects. First, the problem is a sort of matching problem. We already have the Chinese candidates and thus do not need to generate the Chinese transliterations. The other difference is, to avoid the double errors from English phonetic representation to pin-yin and from pin-yin to characters, we use a similar idea as in [16,17] to segment an English name into a sequence of syllables, compute the probability between an English syllable and a Chinese character to estimate the possibility. The aim is computing the phonetic similarity for selecting the right translation. First of all, we segment the English term into a sequence of syllables based on heuristic rules and then compute the transliteration cost use the following equation.

$$Trl(s,t) = \frac{P(s,t)}{D(s,t)} \tag{4}$$

where P(s,t) is the co-occurrence probability of s and t which is defined as:

$$P(s,t) \approx \prod_{i=1}^{\min(m,n)} (1-\gamma_1)\, prob(e_i, c_i) \tag{5}$$

where γ_1 is the smoothing weight. $prob(e_i,c_i)$ is the probability between an English syllable e_i and a Chinese character c_i and is computed based dynamic programming from the training corpus contains 37665 proper name pairs. $D(s,t)$ is the number of syllable difference between an English term s and a Chinese candidate t, which is defined as:

$$D(s,t) = \varepsilon + |\, m-n \,| \tag{6}$$

Here ε is a decaying parameter, m is the total number of English term syllables and n is the total number of Chinese characters.

In order to improve incorrect transliteration mapping between English syllables and Chinese characters, we combine the forward mapping and backward mapping. The final transliteration cost is defined as:

$$Trl(s,t) = \frac{Trl_F(s,t) + Trl_B(s,t)}{2} \tag{7}$$

where $Trl_F(s,t)$ is the forward transliteration value and $Trl_B(s,t)$ is the backward transliteration value.

4. The Experiment Setup and Result Analysis

Experiments are carried out on transliterated word identification and translation mining. The experiment data and setup is described in this section. And an experiment result analysis is made for further study and its application in query translation mining process.

4.1 Experiment on Transliterated Word Identification

The person name dictionary published by the Xinhua News Agency is used as the training corpus, which contains 37669 person names by transliteration. There are 119329 Chinese characters in the corpus, and 376 different characters.

To test the performance, another different dictionary is used as the test corpus, which contains 106191 transliterated person names.

Because all the words in the dictionaries are person names, some non-transliterated words from a common verbal dictionary are mixed into the test corpus, which contains 12205 items.

The transliteration probability of each character to be part of a transliteration word is first calculated based on the person names from the Xinhua News agency. For each Chinese character t, if it is not in the person name dictionary, the transliteration probability is zero. Or the probability can be calculated based on the following equation,

$$TP(w) = \frac{c(t)}{119329} \qquad (8)$$

In which, count(t) is the count of times that the character t occurs in the person name dictionary. 119329 is the total number of Chinese characters in the person names dictionary.

Then, the Model 2 is utilized to calculate the average character probability of each word in the person name dictionary. The minimum value of the probability is set as the threshold value θ_2. Here precision is calculated for evaluation as following.

$$precision = \frac{\#of\ correctly\ classified\ words}{\#of\ words\ for\ classification} \qquad (9)$$

At last, the transliteration word identification process is applied based on Model 1, Model 2 and the combination. The performance of the process on the test corpus is shown in the table followed.

4.2 Experiment on Translation Mining from Search Engine Snippets

100 person names are selected randomly from a foreign name dictionary as the test suite. The top N coverage rate, which refers to the ratio of the names whose correct translation is included in the top N mining results, is evaluated. The experiment result is shown in Table 2.

Based on the result above, a comparison of the translation mining is compared with the famous BabelFish translation system. The result is as in Table 3.

False results are underlined in the table. The result shows that our system outperforms BabelFish in transliterated word translation. Result analysis shows that transliteration characters feature is a good feature for transliterated word identification, which can serve as a basis for transliteration word identification. The precision of transliterated word identification becomes low when the precision of non-transliterated word become high. That is because we choose some characters that distinguish between transliterated word and non-transliterated word as transliterated characters. Our approach has failed when we encounter the word such as “巧克力” (chocolate). These words are not transliterated by using standard transliterated characters.

Table 1. Transliterated and non-transliterated word identification performance

Model	classification	Sum	# words classified true	# words classified false	Precision
Model 1	Transliterated word	106191	104197	1994	98.122%
	non-transliterated words	12205	280	11925	97.706%
Model 2	Transliterated word	106191	105939	252	99.763%
	non-transliterated words	12205	1585	10620	87.014%
Model 1& model 2 combined	Transliterated word	106191	104574	1617	98.477%
	non-transliterated words	12205	340	11865	97.212%

Table 2. Top N inclusion rates of the name translation mining from different number of snippets

# returned snippets	Model	Top1	Top3	Top5	Top10
50	Transliteration	52.1%	68.9%	72.5%	78.3%
	+ frequency and distance	58.1%	77.5%	80.6%	89.1%
100	Transliteration	54.3%	71%	76.8%	81%
	+ frequency and distance	73.6%	87.6%	89.1%	94.6%
150	Transliteration	54.4%	70.7%	76.8%	80.8%
	+ frequency and distance	73.6	82.2%	87.6%	91.5%

Table 3. Comparison of translation mining and machine translation of person names

NO.	Name	Mined transliteration	BabelFish machine translation
1	格兰芬多	GRYFFINDOR	the standard sweet orchid smell are many
2	斯莱特林	Slytherin	Slye tring
3	拉文克劳	revenclaw	Lavin crowe
4	赫奇帕奇	HUFFLEPUFF	Hetch Patch
5	韦斯莱	George Weasley	Wei slye
6	格兰杰	granger	Grainger
7	阿不思	albus	Arab league does not think
8	吉德罗	Dumbledore	Lucky Deluo
9	克林顿	Clinton	Clinton
10	布什	Bush	Bush

5. Conclusions

Translation mining is a key process for lexicon acquisition in cross-language information retrieval, machine translation, etc. For better translation mining performance, a supervised transliteration person name identification process is introduced, which helps classify the types of query lexicon. Concepts of transliteration characters and transliteration probability of a character are proposed. Based on the two concepts, two models to identify a transliteration person name are proposed for a supervised classification algorithm. Experiment results show that our method is highly effective.

REFERENCE

[1] F. Huang and Y. Zhang, “Ming key phrase translations from web corpora,” Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp. 483–490 ACL, 2005.

[2] P. J. Cheng, J. W. Teng, R. C. Chen, J. H. Wang, W. H. Lu, and L. F. Chien, “Translating unknown queries with web corpora for cross-language information retrieval,” in the Proceedings of 27th ACM SIGIR, ACM Press, pp. 146–153, 2004.

[3] C. Y. Lu, Y. Xu, and S. Geva, “Web-based query translation for English-Chinese CLIR,” Computational Linguistics and Chinese Language Processing, Vol. 13, No. 1, pp. 61–90, 2008.

[4] M. Nagata, T. Saito, and K. Suzuki, “Using the web as a bilingual dictionary,” Proceedings of ACL 2001 Workshop Data-Driven Methods in Machine Translation, pp. 95–102. 2001.

[5] W. H. Lu, L. F. Chien, and H. J. Lee, “Translation of web queries using anchor text mining,” ACM Transactions on Asian Language Information Processing (TALIP), Vol. 1, No. 2, pp. 159–172, 2002.

[6] S. Li and H. T. Ng, “Mining new word translations from comparable corpora,” COLING 2004 ACL, 2004.

[7] M. L. Zhou and J. M. Yao, “Mining named entity transliterations from comparable corpora,” Proceedings of 7th International Conference on Chinese Computing, 2007.

[8] J. Li, “Researching and implementing of English-Chinese transliteration method based on text,” Master’s degree thesis, Harbin Institute of Technology, 2005.

[9] W. Gao, “Phoneme-based statistical transliteration of foreign names for OOV problem [D],” The Chinese University of Hong Kong, 2004.

[10] P. Virga and S. Khudanpur, “Transliteration of proper names in cross-lingual information retrieval[A],” in Proceedings of the ACI Workshop on Multilingual Named Entity Recognition [C], 2003.

[11] Xinhua News Agency, “Translation name office dictionary of world-wide person name translations,” China Translation and Publishing Corporation, 1993.

[12] W. H. Lin and H. H. Chen, “Backward machine transliteration by learning phonetic similarity,” in Proceedings of CONLL, Taipei, Taiwan, pp. 139–145, 2002.

[13] T. Lin, C. C. Wu, and J. S. Chang, “Word-transliteration alignment,” in Proceedings of ROCLING XV, Hsinchu, Taiwan, pp. 1–16, 2003.

[14] W. Gao, K. F. Wong, and W. Lam, “Phoneme-based transliteration of foreign name for OOV problem,” in Proceedings of the first International Joint Conference on Natural Language Processing (IJCNLP), Hainan Island, China, pp. 274–381, 2004.

[15] W. Lam, R. Z. Huang, and P. S. Cheung, “Learning phonetic similarity for matching named entity translations and mining new translations,” in Proceedings of 27th International ACM SIGIR Conference on Research and Development in Information Retrieval, the University of Sheffield, UK, pp. 281–288, 2004.

[16] S. Wan and C. M. Verspoor, “Automatic English-Chinese name transliteration for development of multilingual resources,” in Proceedings of 36th Annual Meeting of the Association for Computational Linguistics, Montreal, Quebec, Canada, pp. 1352–1357, 1998.

[17] W. H. Lu, J. H. Lin, and Y. S. Chang, “Improving translation of queries with infrequent unknown abbreviations and proper names,” Computational Linguistics and Chinese Language Processing, Vol. 13, No. 1, pp. 91–120, 2008.

J. Software Engineering & Applications, **2009, 2: 127-135**
doi:10.4236/jsea.2009.22019 Published Online July 2009 (www.SciRP.org/journal/jsea)

A Performance-Driven Approach for Restructuring Distributed Object-Oriented Software

Amal Abd El-Raouf[1], Tahany Fergany[2], Reda Ammar[3], Safwat Hamad[4]

[1]Computer Science Department, Southern CT State University, New Haven, CT, USA; [2]Computer Science Department, University of New Haven, CT, New Haven, USA; [3]Computer Science and Engineering Department, University of Connecticut, CT, Storrs, USA; [4]Computer & Information Sciences Department, Ain Shams University, Abbassia, Cairo, Egypt.
Email: abdelraoufa1@southernct.edu, {tfergany, reda}@engr.uconn.edu

Received March 1st, 2009; revised May 6th, 2009; accepted May 29th, 2009.

ABSTRACT

Object oriented techniques make applications substantially easier to build by providing a high-level platform for application development. There have been a large number of projects based on the Distributed Object Oriented approach for solving complex problems in various scientific fields. One important aspect of Distributed Object Oriented systems is the efficient distribution of software classes among different processors. The initial design of the Distributed Object Oriented application does not necessarily have the best class distribution and may require to be restructured. In this paper, we propose a methodology for efficiently restructuring the Distributed Object Oriented software systems to get better performance. We use Distributed Object-Oriented performance (DOOP) model as guidance for our restructuring methodology. The proposed methodology consists of two phases. The first phase introduces a recursive graph clustering technique to partition the OO system into subsystems with low coupling. The second phase is concerned with mapping the generated partitions to the set of available machines in the target distributed architecture.

Keywords: *Performance Modeling, Distributed Object Oriented, Restructuring Methodology*

1. Introduction

The software restructuring techniques present solutions for the hardware/software mismatch problem in which the software structure does not match the available hardware organization [1,2,3,4]. There are two approaches to solve such a problem; either to configure the hardware to match the software components (hardware reconfiguration), and/or to reconfigure the software structure to match the available hardware by reorganizing its components (software restructuring). The first approach is impractical especially in complex programs containing many interacting modules (or subtasks). The second approach is more practical especially in computing environments that contain large number of users. It provides an optimal way to use the available system capabilities, reduces the overall computational cost, and improves the overall system performance.

The basic idea of distributed software restructuring techniques as described in [2] is to select the best alternative structure(s) for a constrained computing environment while reducing the overall resources need. These structures can be created through; granularity definition (merging of modules), alternative modules ordering, loop decomposition, or multiple servers support. It has been shown that performing software restructuring ahead of the partitioning, allocation and scheduling phases improved the results obtained from these phases and reduced the overall resources cost. Unfortunately this technique is not applicable for Distributed Object Oriented (DOO) systems.

In DOO systems, a program is organized as a set of interacting objects, each of which has its own private state rather than a set of functions that share a global state. Classes represent abstraction that makes adapting software easier and thus lower the cost of reuse, maintenance and enhancement. OO paradigm is based on several concepts such as encapsulation, inheritance, polymorphism, and dynamic binding [5,6]. Although these features contribute to the reusability and extensibility of systems, they produce complex dependencies among classes. The most interesting feature of OO approach is that objects may be distributed and executed either sequentially or in parallel [7]. Distributed objects are one of the most popular programming paradigms in today's computing environments [8,9], naturally extending the sequential message-passing-oriented paradigm of objects.

Designing a distributed Object Oriented application depends on how performance-critical the application is. An important phase is to tune performance by “concretizing” object locations and communication methods [10]. At this stage, it may be necessary to use tools to allow analysis of the communication patterns among objects in order to take the right allocation decision.

The principal goal of this research is to develop new methodology for efficiently restructuring the DOO software to fully exploit the system resources and get better performance.

This paper is organized as follows: the second section states the problem definition, including our objective and assumptions. Section 3 describes the analytical DOOP model its basic components and how it can be used to measure the communication cost between different classes. Section 4 presents the restructuring scheme and the algorithm used in the first phase and the different approaches proposed for the second phase of the restructuring process. Then a comparison and analysis of generated results are included within Section 5. Finally, Section 6 draws our conclusions.

2. Problem Definition

In this paper, we consider restructuring DOO applications for mapping on a distributed system that consists of a collection of fully connected homogenous processors in order to attain better performance. This process is achieved in two phases illustrated in Figure 1. The first phase is concerned with identifying clusters of a dense community of classes within the DOO system. This helps in decomposing the system into subsystems that have low coupling and are suitable for distribution. The inter-class communication is modeled as a class dependency graph. In the second phase, we investigate the possibilities of grouping the generated clusters and mapping them to the nodes of the distributed system. The main objective is to propose a group of sub-systems, each has maximum communication cost among the inner-classes and the communication cost among the sub-systems is minimized. Then, the proposed group of sub-systems is mapped to the available nodes in the distributed system.

3. Distributed Object-Oriented Performance Model

Performance analysis is a very important phase during the software development process. It will ensure that the system satisfies its performance objectives. Performance analysis will not only evaluate the resources utilization but also will allow comparing different design alternatives, detecting bottlenecks, maintenance, re-use and identify the tradeoffs.

Most OO approaches studying the performance of OO systems are based on either the system measurements after its development or mapping it to a conventional performance model and hence no way to preserve the OO features during the analysis phase.

In [11], the Distributed Object Oriented Performance (DOOP) model was introduced. The DOOP model analyzes and evaluates the overall time cost considering the

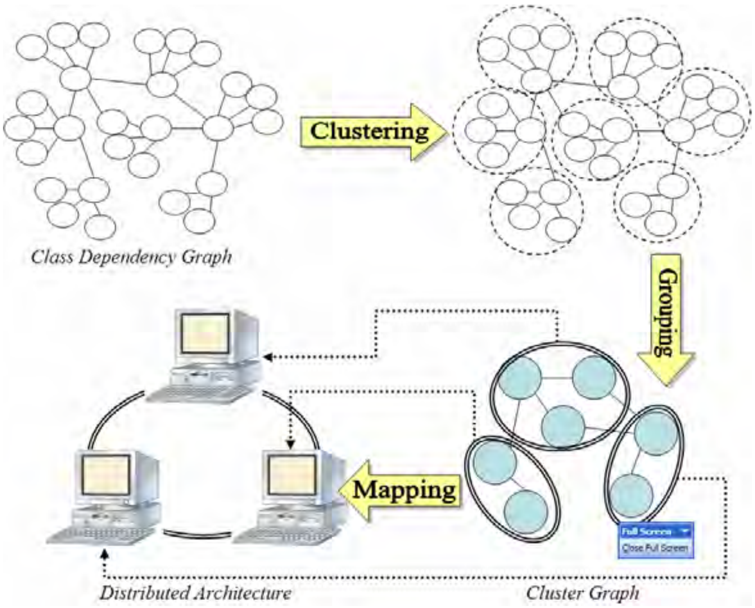


Figure 1. Two-phase process for restructuring DOO software

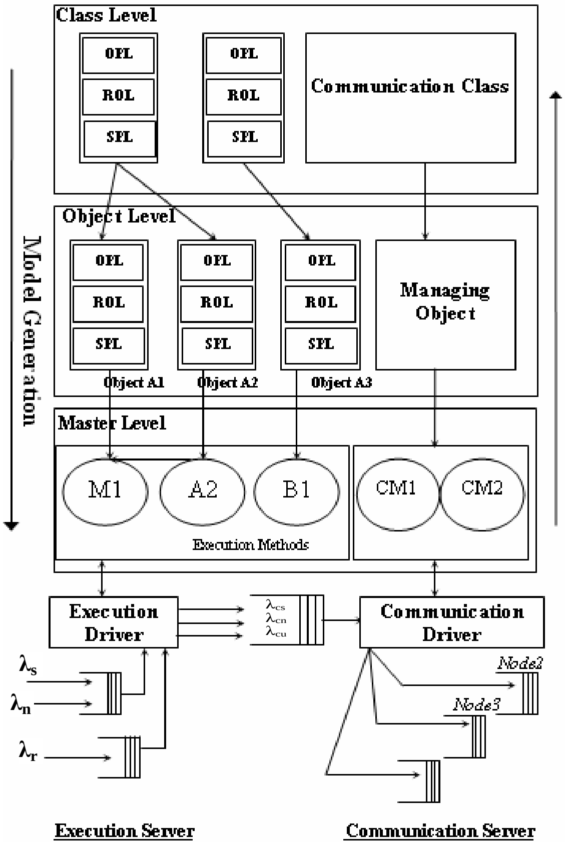


Figure 2. The DOOP model node structure

communication overheads while preserving the features, properties and relationships between objects. According to the model, each node in a DOO system will be represented as shown in Figure 2.

The performance model consists of two main parts: the execution server and the communication server. The execution server represents and evaluates the execution cost of the software modules that reside on the target node. The communication server provides the analysis of the communication activities (including objects updating) as well as the evaluation of the communication cost.

In our restructuring scheme, we utilize the DOOP model in the evaluation process of the communication activities among classes as shown below.

Assume that the overall arrival rate to the communication queue λ_{ck} is given by:

$$\lambda_{ck} = \lambda_{cs} + \lambda_{cn} + \lambda_{cu} \qquad (1)$$

where λ_{cs}, λ_{cn} and λ_{cu} represent the communication arrival due to External User Request (EUR), Remote Request (RR), and updating objects' data on other nodes, respectively.

$$\lambda_{cs} = \beta_s \lambda_s\,,\ \lambda_{cn} = \beta_n \lambda_n\,,\ \lambda_{cu} = \sum_{i=1}^{N} \lambda_{Ui}$$

where, β_s and β_n are the message multipliers for EUR and RR. Let λ_{cui} be the arrival rate corresponding to object i data updating.

Since the updating process to an object i occurs due to processing EUR or RR, P_{i1} defined to be the probability that object i is updated due to EUR, P_{i2} is the probability that object i is modified due to RR. λ_{cui} can be expressed as:

$$\lambda_{cui} = P_{i1}\lambda_s + P_{i2}\lambda_n$$

Hence, the expected communication service time for each class will be:

$$t_{cs} = \frac{m_s}{R} \quad t_{cn} = \frac{m_n}{R} \quad t_{ui} = \frac{m_{ui}}{R}$$

where t_{cs}, t_{cn} and t_{ui} are the expected communication service time for EUR, RR and for update requests from object i. While m_s, m_n and m_{ui} are the expected message sizes of EUR, RR and of sending object i updating data. R is the communication channel capacity. Furthermore, the average communication service time for node (k) will be:

$$t_{ck} = P_{cs}t_{cs} + P_{cn}t_{cn} + \sum_{i=1}^{N} P_{ui}t_{ui} \qquad (2)$$

$$p_{cs} = \frac{\lambda_{cs}}{\lambda_{ck}} \qquad p_{cn} = \frac{\lambda_{cn}}{\lambda_{ck}} \qquad p_{ui} = \frac{\lambda_{ui}}{\lambda_{ck}}$$

where P_{cs}, and P_{cn} are the probabilities of activating communication service by the external user requests and by remote request respectively. P_{ui} is the probability of sending object i's data update to other nodes.

4. Restructuring Scheme

Our restructuring scheme starts with using the DOOP model to map the distribute Object Oriented application into a Class Dependency Graph (CDG) illustrated in the following subsection. Then this CDG is restructured using our two-phase methodology. The first phase is based on a recursive use of a spectral graph bi-partitioning algorithm to identify dense communities of classes. The second phase is concerned with mapping the generated partitions to the set of available machines in the target distributed architecture. We will describe two proposed approaches: the Cluster Grouping Approach and the proposed Double K-Clustering Approach. The details of each phase are described in the following subsections.

4.1 The Class Dependency Graph (CDG)

If we assumed that each individual class is initially allocated to a separate node in the distributed system, then the above DOOP model can be used as a powerful analy-

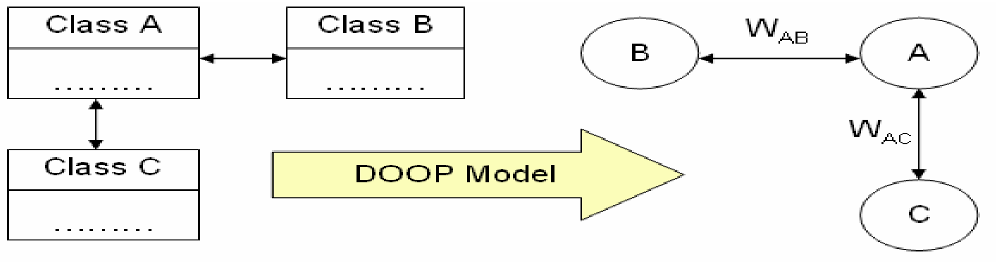


Figure 3. A graph dependency graph for interclass communication

```
ALGORITHM      GraphCluster(C, Clusters, CIndx)

INPUT:         C = Adjacency matrix (weights matrix).
               Clusters = a vector indicating the cluster no.
                 of each node in the CDG graph.
               CIndx = the cluster index representing the
                 subgraph needed to be bi-partitioned.
OUTPUT:        NewClus = a vector indicating which node in
                 the graph belongs to.

(1)  Let CurrC be the extracted Adjacency matrix of the graph
       indicated by CIndex.
(2)  Partition the CurrC into two subgraphs
       Indx =GraphBipart(CurrC)
(3)  Create vector G1 and G2 to hold the indices of the first and
       the second subgraphs respectively.
(4)  IF ( NOT WellPartitioned(CurrC, G1) )   OR
        ( NOT WellPartitioned(CurrC, G2) ) THEN
         return Clusters
     END IF
(5)  Update NewClus with new portioning in G1 & G2.
(6)  Recursively bipartition the produced subgraphs G1 and G2
       NewClus=GraphCluster(C, NewClus, CIndx*10+1)
       NewClus =GraphCluster(C, NewClus, CIndx*10+2)
```

Figure 4. A recursive graph clustering algorithm

tical tool to accurately evaluate the communication activities among classes in a distributed OO system. The calculated values will be used to generate the Class Dependency Graph (CDG) of the given OO application.

The class dependency graph CDG as shown in Figure 3 is a graph representation for interclass communications. In CDG, each class is represented as a vertex and an edge between class A and B represents a communication activity that exists between these two classes due to either data transfer or classes' dependency. The weight of the edge WAB represents the cost of that communication activity between class (A) and class (B). If no data communication or relationship dependency has been recognized between two classes, no edge will connect them in the CDG.

4.2 First Phase: Clustering System Classes

In this section, we describe a clustering technique that is considered the first primary phase of the restructuring approach to be proposed in this paper. After applying this step, the object oriented system is decomposed into subsystems that have low coupling and are suitable for distribution. The technique is based on a recursive use of a spectral graph bi-partitioning algorithm to identify dense communities of classes. At each recursive call, the CDG is partitioned into two sub graphs each of which will be further bi-partitioned as long as the partitioning results in clusters that are denser than their parents. Hence, the stopping condition depends on how good the produced partition is. A sub-graph is considered a Well-Partitioned if the summation of the weight of internal edges (those between the vertices within a sub-graph) exceeds those of external edges (those between the vertices of the sub-graph and all the other vertices in other sub-graphs). The iteration stops when at least one of the produced sub-graphs is badly partitioned (the summation of the weight of external edges exceeds those of internal edges). In this case, the bi-partitioning step is considered obsolete and the algorithm will backtrack to the clusters generated in the previous step. At the end, the identified sub-graphs are the suggested clusters of the system. Figure 4 shows a detailed step by step description of the clustering Algorithm as described in [12].

The mathematical derivation of the spectral factorization algorithm used in our clustering approach was introduced in [13]. It is originally solving the l-bounded Graph Partitioning (l-GP) problem. However, we have adapted the solution methodology to fit within our bi-partitioning approach.

4.3 Second Phase: Mapping Classes to Nodes

In this phase, the restructuring process is accomplished by mapping the set of DOO application clusters, resulted from the first phase to the different network nodes in order to attain better performance. To achieve this goal, we perform the mapping process taking into consideration our objective of minimizing the effect of class dependency and data communication. It is assumed that the target distributed system consists of a set of homogeneous processors that are fully connected via a communication network.

The mapping process has two cases. The first case appears when the number of candidate clusters are less than or equal to the number of the available nodes. In this case the mapping process will be done simply by assigning each cluster to one of the available nodes. The problem occurs in the second case, when the number of the generated clusters exceeds the number of available nodes. This is a more realistic view since there will be always huge software systems with large number of classes and limited hardware resources.

In the following subsections, we are going to introduce two different approaches to be used in performing the grouping and mapping steps of the second phase: the Cluster Grouping Approach and the Double K-Clustering Approach. In Section 5, the results would be compared with the direct K-Partitioning Approach listed in Figure 4, where the graph is partitioned into a pre-specified number equals exactly to the number of nodes in the target distributed system. This is a one step allocation process that also maintains the criterion of minimizing inter-cluster communication cost.

4.3.1 The Cluster Grouping Approach:

In this approach, we use the clusters (sub-systems) generated at the first phase as the main candidates for distribution. The technique is based on merging clusters into groups in a way that keeps the communication costs among them minimized. As a result a cluster graph is formed, where the nodes represent clusters and the edges will capture the possible communication links that may exist among clusters. Then, the K-Partitioning algorithm is used to perform cluster grouping such that the number of the resultant groups is equal to the number of available nodes. The result will be groups of clusters that have minimal communication costs among them. Finally, those groups are assigned to the different available nodes in the distributed environment.

4.3.2 The Double K-Clustering Approach:

The K-Partitioning algorithm can not predict the number of system modules or clusters as the recursive Bi-Partitioning algorithm does. Instead, the number of required clusters must be given as an input parameter which is the k value. Hence, we will make use of the advantages provided by both algorithms: the recursive Bi-Partitioning and the K-Partitioning. So, the first phase described in Section 3 is considered as a pre-phase to estimate the number of the existing sub-systems within the distributed application. However, we will use the K-Partitioning algorithm twice. In the first time, the original class dependency graph CDG is clustered according to the number suggested at the pre-phase. In the second time the K-Partitioning algorithm is used again in the same way as in the cluster grouping approach illustrated above such that the number of the resultant groups is equal to the number of available nodes. Then, the resultant clusters are mapped to the available nodes.

5. Simulation and Results

In the section, we provide an analysis of both the clustering and the mapping phases described above. This illustration is done through a case study which describes the detailed steps of the two-phase restructuring process and a set of simulation experiments that were performed to validate our results.

5.1 Case Study

We have developed a performance-driven restructuring simulator using Matlab 7.0.4. A Graphical User Interface (GUI) utility has been implemented to show the system status after each step while the algorithm proceeds. The simulator has a friendly user interface that allows the user to specify the nodes and edges in the systems, and then it will be able to generate the class dependency graph (CDG).

We conducted an experiment using an OO system that consists of 28 classes. The CDG that was generated by the simulator for this system is given in Figure 5. In this section we provide a step-by-step description of applying the proposed restructuring techniques illustrated above. Furthermore, we are going to analyze and compare the results in each case. Figure 6 shows the resultant system clusters generated by the proposed bi-partitioning algorithm after the first phase. We can see that the proposed first phase algorithm has created 7 clusters each of which

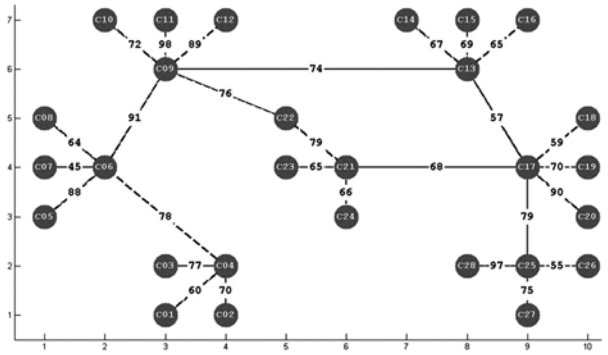


Figure 5. The generated CDG

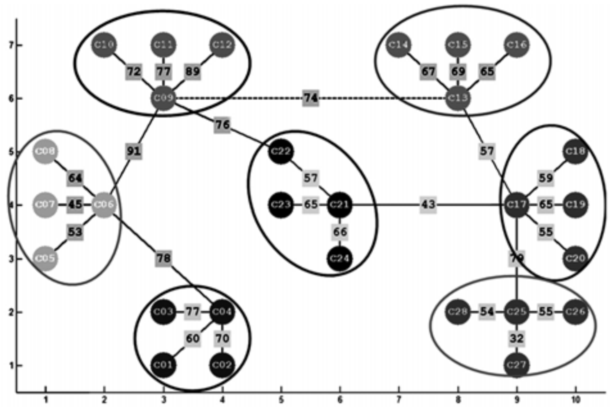


Figure 6. The resultant clusters generated by the restructuring scheme first phase

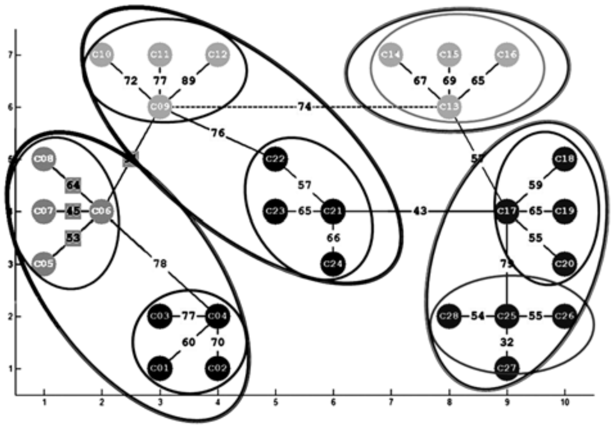


Figure 7. Mapping the DOO system to a 4-node environment using the direct partitioning approach

is marked up with a different color and surrounded by an oval for further highlight.

Now, let us assume that the target distributed environment consists of 4 homogenous nodes that are fully connected. We need to apply any one of the second phase approaches to map the classes to the distributed nodes. First we applied the direct approach that partitioned the original CDG into 4 clusters using the k-partitioning algorithm. The resultant clusters are depicted in Figure 7. In Figure 8 the Cluster Grouping approach is applied to merge the clusters in Figure 8 generating 4 large clusters. However, applying the Double-K clustering approach results in a completely different group of clusters as shown in Figure 9. It started with generating 7 clusters then they were grouped into 4 clusters. The first one includes {C1, C2, C3, C4, C5, C6, C7, C8}, the second one {C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C22, C25, C26, C28}, the third {C21, C23, C24}, and the last one includes only C27.

Notice that each one of the approaches resulted in a different grouping option, each of which has a different communication cost. The Direct Partitioning Approach results in a communication cost equals to 267 time unit. While the Cluster Grouping resulted in a cost of 265 time units, the Double-K produced a grouping of cost 223 time units.

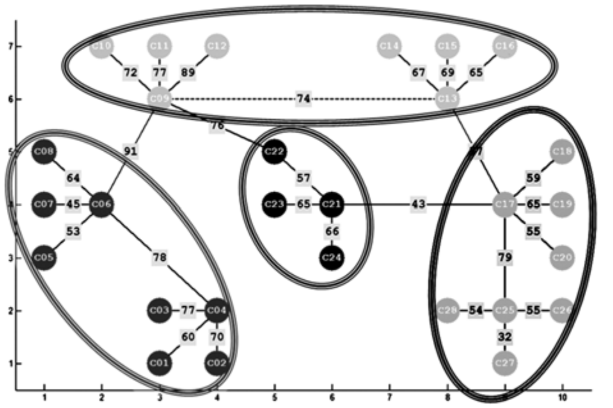


Figure 8. Mapping the DOO system to a 4-node environment using the cluster grouping approach

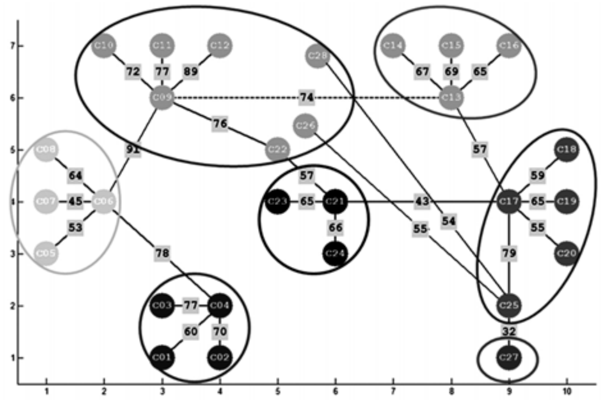


Figure 9. Mapping the DOO system to a 4-node environment using the double-K approach

5.2 Numerical Results

We conducted a number of experiments using a set of DOO applications, whose class dependencies were randomly generated. The generated matrices are assumed to represent the adjacency matrices of the CDG for the systems under inspection. The Adjacency matrices are generated by Andrew Wathen method proposed in [14]. The resultant matrices are random, sparse, symmetric, and positive definite.

In Figure 10, the X-axis represents the number of nodes in the target distributed environment and the Y-axis represents the communication cost in units of time that measured between classes located in different nodes. The decision of allocating a group of classes to a specific node is made by one of three algorithms. Two of them are the algorithms proposed above: the Cluster Grouping Approach and the Double K Approach. The third one is the well-known K-Partitioning approach.

Each data point in the comparison chart is an average of a set of many trials. In each trial, the adjacency matrix of the CDG is generated randomly having 107 classes. For each generated random matrix, the Bi-Partitioning algorithm was used to verify that it will result in exactly 11 clusters, otherwise the matrix is neglected and another one is generated.

The performance comparison depicted in Figure 10 shows that the Double-K Clustering Approach provides the best performance over the other algorithms since it gives the minimum interclass communication cost for various numbers of nodes (machines). Then comes the cluster grouping approach and at last comes the K-Partitioning approach. However, when the number of generated clusters equals to the number of machines or nodes, the proposed Double-K Clustering Approach gives typically the same results as that of the K-Partitioning algorithm. This is a logical finding since in this case the second clustering step in the Double-K Clustering approach is eliminated reducing the approach to the original K-Partitioning Algorithm.

In order to confirm the correctness of the proposed methodologies, we have conducted a number of experiments using various sets of classes and their associated clusters. These classes were generated randomly and the results were averaged just like the experiment illustrated above. Table 1 and Table 2 present the communication costs computed for each partition generated by the three approaches when applied on different sets of classes. Each table represents a simulation evaluation targeting a distributed system architecture composed of a predetermined number of machines. Figure 11 and Figure 12 show the results when using 4 and 6 machines respectively.

The simulation results came to be consistent with the results discussed in the case study presented above. That is, while changing the number of classes as well as the structure of the underlying distributed architecture, it is still the same remarks. The proposed two approaches Cluster Grouping and Double K-Clustering outperform the Direct Partitioning Approach as long as the number of nodes is less than the number of generated clusters.

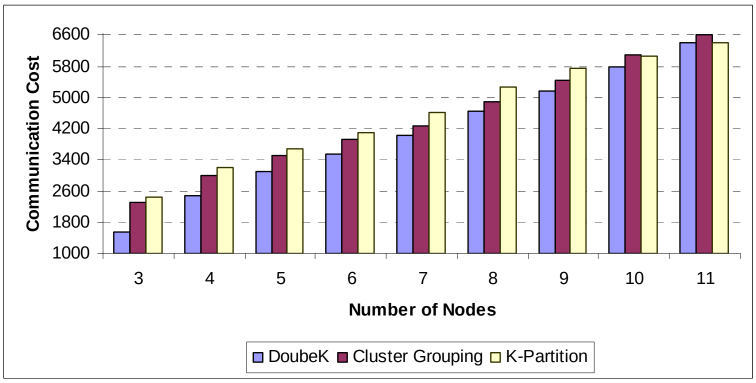


Figure 10. Interclass communication cost measured after applying different restructuring approaches

Table 1. Simulation results considering a distributed architecture consisting of four nodes

No. of Classes	No. of Clusters	Communication Cost		
		Double K	Cluster grouping	K-Partitioning
51	6	1680	1899	2045
62	7	2124	2160	2632
79	9	2097	2185	2600
107	11	2483	3009	3196
153	15	3143	3784	4090

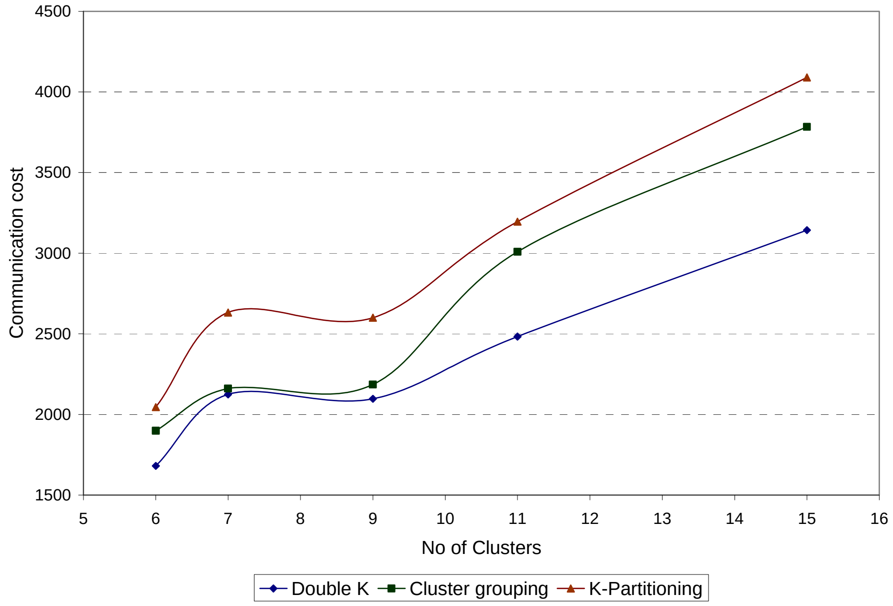


Figure 11. The simulation result of mapping different DOO systems with different number of classes on four nodes

Table 2. Simulation results considering a distributed architecture consisting of six nodes

No. of Classes	No. of Clusters	Communication Cost		
		Double K	Cluster grouping	K-Partitioning
51	6	2908	2911	2908
62	7	3079	3204	3453
79	9	3069	3205	3608
107	11	3565	3924	4117
153	15	4562	4978	5399

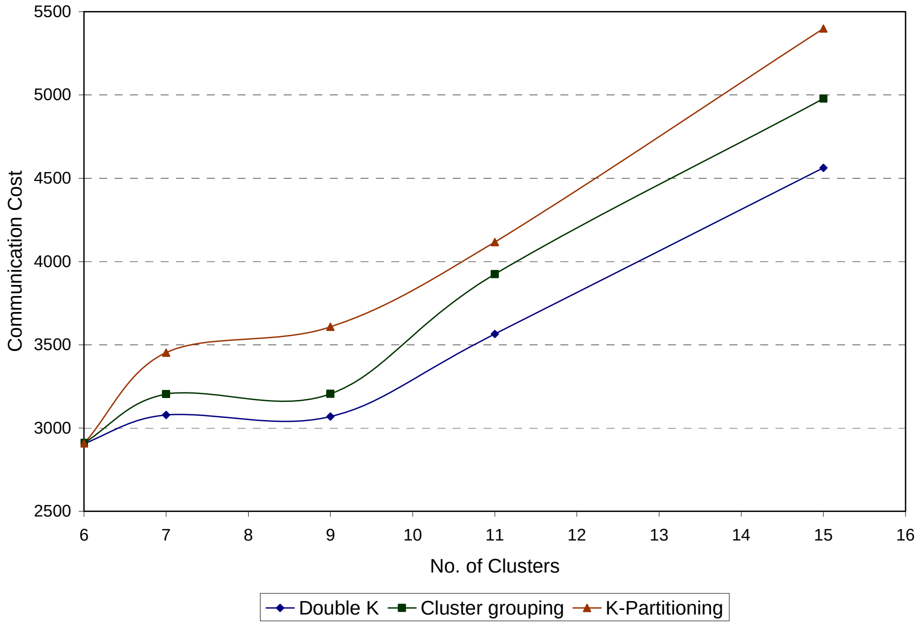


Figure 12. The simulation result of mapping different DOO systems with different number of classes on six nodes

6. Conclusions

In this paper, we proposed a restructuring approach for DOO applications into a distributed system consisting of a collection of fully connected homogenous processors. The restructuring process was performed in two phases: the clustering phase and the mapping phase. The first phase is based on the theory of spectral graph bi-partitioning, where the Distributed Object Oriented performance model was used efficiently to evaluate the communication costs between different classes. In the second phase, the identified subsystems were assigned to different machines in the target distributed environment. This is done through two proposed algorithms: cluster grouping approach and the Double-K Clustering approach. A Comparison was made between the proposed approaches and the k-partitioning algorithm. The results showed that the Double-K Clustering Approach provides the best performance in terms of minimizing the communication cost among classes located on different nodes (machines).

REFERENCES

[1] A. Raouf, R. Ammar, and T. Fergany, "Object oriented performance modeling and restructuring on a pipeline architecture," The Journal of Computational Methods in Science and Engineering, JCMSE, IOS Press, Vol. 6, pp. 59–71, 2006.

[2] T. A. Fergany, "Software restructuring in performance critical distributed real-time systems," Ph. D. Thesis, University of Connecticut, USA, 1991.

[3] T. A. Fergany, H. Sholl, and R. A. Ammar, "SRS: A tool for software restructuring in real-time distributed environment," in the Proceedings of the 4th International Conference on Parallel and Distributed Computing and Systems, October 1991.

[4] H. Sholl and T. A. Fergany, "Performance-requirements-based loop restructuring for real-time distributed systems," in the Proceedings of the International Conference on Mini and Microcomputers, From Micro to Supercomputers, Florida, December 1988.

[5] B. Meyer, "Object-oriented software construction," Prentice-Hall International (UK), Ltd, 1988.

[6] Ostereich, "Developing software with UML: OO analysis and design in practice," Addison Wesley, June 2002.

[7] J. K. Lee and D. Gannon, "Object oriented parallel programming experiments and results," in the Proceedings of Supercomputing 91, IEEE Computer Society Press, Los Alamitos, Calif, pp. 273–282, 1991.

[8] Sun Microsystems Inc. Java home page, http://www.javasoft.com.

[9] J. Waldo, G. Wyant, A. Wollrath, and S. Kendall, "A note on distributed computing," Sun Microsystems Laboratories, Technical Report-94-29, November 1994.

[10] I. Sommerville, "Software Engineering," 8th Edition, Addison-Wesley Publishers Ltd, New York, 2007.

[11] A. A. El-Raouf, "Performance modeling and analysis of object oriented software systems," PhD Dissertation, Department of Computer Science & Engineering, University of Connecticut, 2005.

[12] S. Hamad, R. Ammar, A. Raouf, and M. Khalifa, "A performance-driven clustering approach to minimize coupling in a DOO system," the 20th International Conference on Parallel and Distributed Computing Systems, Las Vegas, Nevada, pp. 24–26, September 2007.

[13] J. P. Hespanha, "An efficient MATLAB algorithm for graph partitioning," Technical Report, Department of Electrical & Computer Engineering, University of California, USA, October 2004.

[14] A. J. Wathen, "Realistic eigenvalue bounds for the galerkin mass matrix," The Journal of Numerical Analysis, Vol. 7, pp. 449–457, 1987.

Call for Papers

International Conference on Computational Intelligence and Software Engineering (CiSE)

December 11~13, 2009 Wuhan, China

http:// www.ciseng.org

The International Conference on Computational Intelligence and Software Engineering (CiSE) will be held on December 11~13, 2009 in Wuhan, China. This conference will cover issues in **computational intelligence, information security, multimedia and graphics technologies, and software engineering.** The conference proceedings will be published by IEEE, and all papers accepted will be included in IEEE Xplore and indexed by EI Compendex.

TOPICS

- Artificial Intelligence
- Automated Reasoning
- Autonomous Systems
- Cloud Computing
- Cluster Computing
- Cognitive Science
- Communication Networks and Protocols
- Computational Biology
- Computational Chemistry
- Computational Neuroscience
- Computational Physics
- Computer Graphics
- Data Modeling
- Database Mining
- Database Technology
- Evolvable Hardware
- Expert Systems
- Fuzzy Systems
- Geographical Information System
- Grid Computing
- Hardware Implementation
- Human-computer Interaction
- Hybrid Systems
- Image Processing
- Image Understanding
- Machine Learning
- Multi-Agent Systems
- Natural Neural Systems
- Neural Genetic Systems
- Neural-Fuzzy Systems
- Numerical Algorithms
- Operating Systems
- Pattern Recognition
- Programming Methodology
- Project Management
- Real Time Control
- Requirement Analysis
- Simulation and Modeling
- Software Engineering
- Software Maintenance
- Software Standards and Design
- Software Testing and Quality Control
- Symbolic Mathematics
- Technological Forecasting
- Visualization
- Web-based Services

IMPORTANT DATES

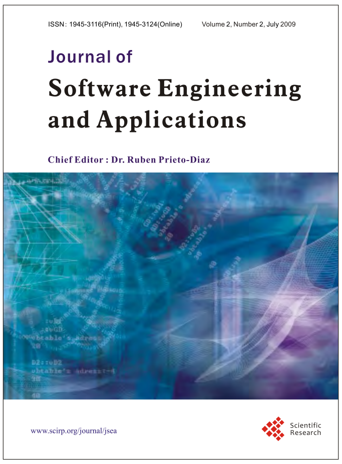
♦ Paper submission due:	Jun. 20, 2009
♦ Acceptance notification:	Aug. 20, 2009
♦ Conference:	Dec. 11–13, 2009

CONTACT INFORMATION

Website: http://www.ciseng.org

E-mail: info@ciseng.org

Call For Papers


ISSN: 1945-3116(Print), 1945-3124(Online)
Volume 2, Number 2, July 2009
Journal of
Software Engineering and Applications
Chief Editor : Dr. Ruben Prieto-Diaz
www.scirp.org/journal/jsea
Scientific Research

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 2 **July 2009**

J. Software Engineering and Applications. 2009; 67-135.

9771945311006 03