



Zhou, D., Chen, X., Jin, Q., Kuang, Z. and Yang, H. (2017) 'An affinity analysis based CIM-to-PIM transformation', Multiagent and Grid Systems, 13 (3), pp. 269-286.

The final publication is available at IOS Press through <http://dx.doi.org/10.3233/MGS-170271> .

ResearchSPAce

<http://researchspace.bathspa.ac.uk/>

This pre-published version is made available in accordance with publisher policies.

Please cite only the published version using the reference above. Your access and use of this document is based on your acceptance of the ResearchSPAce Metadata and Data Policies, as well as applicable law:-

<https://researchspace.bathspa.ac.uk/policies.html>

Unless you accept the terms of these Policies in full, you do not have permission to download this document.

This cover sheet may not be removed from the document.

Please scroll down to view the document.

An Affinity Analysis Based CIM-to-PIM Transformation

Dongdai Zhou^{1,2,3} Xue Chen¹ Qiuzhi Jin¹ Zhejun Kuang¹ Hongji Yang⁴

¹College of Information and Software Engineering, Northeast Normal University, Changchun, 130117, China

²E-Learning Technological Innovation Center of Jilin Province, Changchun, 130117, China

³Engineering & Research Center of E-learning, Ministry of Education, Changchun, 130117, China

⁴Centre for Creative Computing, Bath Spa University, Bath, England, SN13 0BZ

Abstract: To tackle the problems such as the imperfection and inconsistency in software requirements in traditional Computation Independent Model (CIM) modelling, the low degree of automation as well as the imperfection in the description of Platform Independent Model (PIM) in CIM-to-PIM transforming, in this article, we propose a Business-Process-based CIM modelling method and a CIM-to-PIM transformation approach. Business Process Model is used to express CIM, and UML's Sequence Diagram, State Chart Diagram as well as Class Diagram are used to express PIM. Firstly, the users' requirements are obtained through business process models. We extract use cases from business processes and create use case specifications. A verification mechanism is also added for the use case specification. Secondly, we transform CIMs into PIMs automatically with use case specifications as the inputs as well as combining with use case based thinking, responsibility based thinking and affinity analysis. Finally, by comparing with the methods in other studies, we conclude that methods proposed in this article can ensure model integrity and increase the degree of model transformation automation.

Keywords: MDA; CIM Modelling; Business Process; CIM-to-PIM Transformation; Affinity Analysis

1 Introduction

In July 2001, OMG (Object Management Group) formally promulgated the Model Driven Architecture (MDA), which drove the software development by the establishment of CIM, PIM, and PSM, as well as the conversion between those models. In the following 10 years or so for MDA's development, researchers put much emphasis on PIM, PSM and the transformation from PIM to PSM. CIM-to-PIM, as the first transformation step of MDA, somehow, attracted less attention. As a result, it not only affects the automatic generation of MDA code, but also results in several problems, such as the loss of information in the process of model transformation, requirement inconsistency between the user and the system, and even the code is unable to be executed. For all these reasons, how to build a complete and reliable CIM model, and convert CIM into PIM is of great importance.

The rest of this paper is organised as follows. Section 2 introduces related work and points out the shortcomings of the existing work and the research content of our paper. Section 3 gives the PIM model and a presentation of a proposal for CIM-to-PIM transformation. Section 4 takes a library management system as an example to apply the above modelling as well as the transformation method. In Section 5, we discuss advantages of this paper in contrast with the existing studies in the literature. Finally, Section 6 concludes this paper.

2 Literature Review

At present, there are four aspects in the research of CIM-to-PIM transformation.

(1) Model transformation approach based on use case descriptions parser -- Li [20, 21] proposed 13 kinds of syntactic rules to describe use case event flow. He also provided an approach to parsing sentences described by these syntactic rules to extract different elements (that are used in model transformation) from the flow of events. The elements are sender, receiver, action, etc. Basing on Li's work, Xin, et. al. [4-6], proposed using use case diagrams to describe CIM; using activity diagrams, sequence diagrams and state diagrams to describe PIM; and eventually transforming PIM into CIM. Chen [60] proposed a relationship oriented approach to transforming use case specification into class diagrams. In the PIM layer, class diagram can reflect the structure of the PIM layer, which is the static information of the PIM layer, but cannot express the behaviour information of the PIM layer, which is the dynamic information. This will lead to the incompleteness of the PIM.

(2) Model transformation approach based on meta-model -- Generating systems capable of running from the business requirements faces one big problem, i.e., how to construct an effective CIM and automatically transform CIMs into PIMs. In order to solve these problems, Kherraf, Cao, et. al. [22, 23], proposed an approach to transforming from CIM to PIM based on patterns and archetypes. They used use case diagram and activity diagram to construct CIM, used component diagram to construct PIM, and used patterns and archetypes to simplify the transformation from CIM to PIM. Yin [24] proposed an

approach to transforming from CIM to PIM based on ICEMDA. Firstly, meta-models of CIM and PIM were created. Then, mapping rules between the CIM meta-model and the PIM meta-model were constructed. Finally, she found out corresponding mapping rules developed for converting CIMs to PIMs. The BO model refers to entity information used for the systems, collected through user's requirement. Data model is divided into two abstract levels of BO-R and E-R. BO-R describes dependencies between BO in PIM, and E-R describes the association and constraint between the entities. Cao, et al. [25] proposed an approach to transforming from CIM to PIM based on SOA. They separated model transformation from realisation technology and platforms. The approach mainly includes the following two parts. The first part gives a service-oriented way to model the requirement. The second part gives a model refinement mechanism and a set of refinement rules. The refinement mechanism and rules can transform the requirement model to a set of loosely coupled services and can match these services to their suitable components and interfaces.

(3) Model transformation approaches based on feature and component -- Wei, et al., [7] proposed a feature-oriented component-based approach to the CIM-to-PIM transformation. In this approach, using feature model represents the CIM requirement, and the CIM includes a set of features as well as their relationships, using component as the key element of PIM. One important characteristic of this approach is that it provides a method to decompose the n-to-n relations between features and components into two groups of 1-to-n relations. The other important characteristic is that this approach proposes a way to create components by clustering responsibilities which are operationalised from features. These two characteristics partially parse two basic problems related to the CIM-to-PIM transformation: one is the traceability problem between CIM and PIM, the other is the problem of CIM-based PIM construction.

(4) Model transformation approach based on QVT -- Rodríguez, et. al. [26, 27] proposed an approach to transforming from CIM to PIM based on QVT. QVT (Query/View/Transformation [28] was proposed by OMG, which is an important member of OMG meta-model family. In this approach, activity diagram is used to describe CIM; use case diagram and class diagram of analysis level are used to describe PIM. At last, certain QVT rules and model refinement rules are used to transform CIM into PIM.

Having analysed above studies we found that there are problems in the transformation from CIM to PIM:

(1) For the integrity part, semantic of PIM is not complete as it only contains static model or dynamic model after the transformation.

(2) A low degree of automation, over-reliance on the designer's experience no matter generating static model based on class diagram or dynamic model. Only few approaches automate the process partially. Furthermore, the generated class model is not enough to support the attributes, operations, and inheritance relationship of classes.

Accordingly, the research contents of this paper are as follows:

1. To establish PIM models correctly and completely, and
2. To Automate transformations from CIMs to PIMs. And during the transformation attributes, operations, and inheritance relationship of classes are well supported.

3 Proposed Method of Automatic Transformation from CIM to PIM

3.1 CIM modelling

CIM is the first step of model transformation of MDA, whose integrity directly affects the follow-on models and their transformations. In [60], we proposed a Business-Process-based CIM modelling approach, where Business-Process-based model is able to separate the analysis from the implementation of the system. It can also help dynamically analyse the business process from the perspective of the user and reduce the gap between the users' original needs and the final system. In order to ensure the integrity of CIM, we use Business-Process-based models to dynamically describe the business process of the system, and the use use-case models to describe each business function of the Business-Process-based model in details. Firstly, the user's requirements is obtained through business process models. We extract use cases from business processes and create use case specification. A verification mechanism is added for the use case specification.

CIM is used to describe the system requirements, and therefore a model such as a Use Case diagram that can express the system clearly and directly is needed. But relying simply on the use case diagram (consists of actor, use case, and the association between them) cannot give a detailed description for the specific function of the system. Therefore, use case specification is also needed. Currently, natural language is used to describe the use case specification. Unfortunately, information used for model transformation is hardly to be drawn from the natural language because of the language's defect. Therefore the description of use case specification needs normalised constraints. Li [20, 21] proposed 13 basic English sentence patterns that can satisfy the need of the description of use case specification, as shown in Table 1.

Table 1. 13 Basic English sentence structures

No.	Syntactic Structure
1	subject verb object
2	subject verb object (to) verb1 (object1)
3	subject verb object participle (object1)
4	subject verb object adjective

5	subject verb object object1
6	subject verb object conjunctive to verb1 (object1)
7	subject verb gerund (object)
8	subject verb object preposition object1
9	subject verb object object1
10	subject verb (for) complement
11	subject verb
12	subject be predicative
13	subject verb preposition object1

Thomas, et. al., also proposed that these 13 Basic English sentence patterns can describe the use case specification completely. So, we choose the above sentence patterns as the normalisation mechanism of the use case specification. Transformation from CIM to PIM can be automated by standardising the use case specification and extracting information from it.

3.2 PIM Modelling

There are different PIM modelling methods presently. Most of them use UML as the descriptive language of PIM meta-model for two reasons. One is that UML has become relatively mature after many years' development, and it is widely used in the process of software engineering modelling. The other is that UML2.0 provides a good support to MDA.

In the previous article [60], we used Class Diagram to present PIM models. However, a class model can only express the static information of PIM, and cannot express the dynamic information, which will result in missing information about PIM. Therefore, in this study, we add dynamic information to PIM. In UML 2.0, the dynamic behaviour view includes the state machine diagram, activity diagram and interaction diagram. The State Machine View describes the dynamic behaviour of the object time by modelling the lifetime of each class's object. A state machine is a narrow, deep view of behaviour, a compact view of a single object, through which it is difficult to understand the overall function of a system. Interactive views provide a more global view of object behaviour. In interactive view, Sequence Diagram is selected in this paper. Active view is a special form of state machine for computing and workflow modeling. In this paper, State Diagram is already used at the PIM, so activity diagram is no longer used. To sum up, in the PIM layer, we use three kinds of diagrams in UML to do the modelling, which are Class Diagram, Sequence Diagram and State Diagram. Class Diagram belongs to the static model and Sequence Diagram and State Diagram belong to dynamic models. The combination of static model and dynamic model makes the PIM more complete. The transformation process from CIM to PIM is shown in Fig. 1.

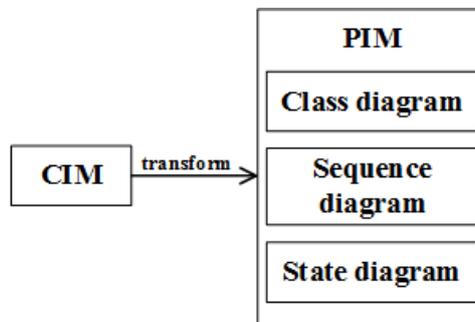


Fig. 1. Transformation process from CIM to PIM

3.3 Transformation from Use Case Models to Class Diagrams

Li, et. Al., [21, 23, 26, 31] proposed a number of Class Diagram based model transformation methods. Use Case driven method is the common choice while transforming from a Use Case Diagram to a Class Diagram. Firstly, robust analysis is used to add attributes for the initial class diagrams. Then sequence diagram is used to add methods for the class diagram. However, in this way, only separate classes can be generated. Relationships between classes like inheritance and dependency cannot be generated automatically. Now transformations to class diagram are lack of support for the relationships between classes, which will reduce the integrity and consistency of the model, and increase the risk of the system. Therefore, in the previous article [60], we proposed a relationship oriented model transformation method.

In the first place, we parse the use case specification event flow to generate a "Function-Argument" matrix. Secondly, the affinity analysis [32] is carried out to obtain the entity class. Thirdly, affinity analysis is used again to obtain business class. At last, we refine the aggregated classes and add relationships between the classes.

In existing studies, the transformation into a Class Diagram depends on experienced people. Only a few parts implemented semi-automation. Furthermore, the generated class model is not enough to support the attributes, operations, and inheritance relationship of classes. In our previous paper [60], a relationship oriented automatic transformation approach which describes

how Use Case specification event flow is transformed to Class Diagram is proposed. On the basis of the use case description template and the use case description semi-formal language function-argument matrix is extracted depends on relationship oriented thinking and class model is automatically aggregated by affinity analysis automation algorithm. In addition, during the transformation attributes, operations, and inheritance relationship of classes are well supported.

3.4 Transformation from Use Case Model to Sequence Diagrams

Xin and Jiang [6, 7] discussed that using Sequence Diagram (which is used to describe the dynamic interaction between objects) to represent the PIM. The sequence diagram is the dynamic representation of a system, and it also shows how objects communicate over times. Sequence Diagram expresses dynamically how a system implements business functions by the dynamic message interactions between objects, so that the model can describe the system more precise. Sequence Diagram consists of object, message, and other basic elements. "Message" is marked by the sender and the receiver of the message. They parse each event flow of the use case event flow to obtain message sequence of the sequence diagram, then assemble the message sequences in a specific pattern. Finally, the system's Sequence Diagram is generated.

3.4.1 Transformation Approach

We parse the use case specification event flow that has been normalised by the sentence patterns provided. The rules of transformation from use case event flow to sequence diagram is obtained by analysing the generated class diagrams as well as 13 English sentence patterns, as shown in Table 2. After resolving each event flow according to the rules, we can get message tuple consists of sender, receiver, message and message type. Message type includes calling message, returning message, etc. After obtaining the message tuple we need to adjust them according to the generated class. At last, sequence diagram is generated from message tuples.

Table 2. Sequence diagram event flow resolution rules

Syntactic Structure No.	Sender	Receiver	Message	Type
1	subject	subject	verb + object	call
2	object	object	verb1 + object1	call
3	object	object	participle + object1	call
4	subject	subject	be + adjective	call
5	subject	subject	set + object1	call
6	object	object	verb	call
7	object	object	verb	call
8 (the preposition is to)	subject	object1	verb + object	return
8 (other situation)	subject	subject	verb + object	call
9	subject	subject	verb + object	call
10	object	object	verb	call
11	object	object	verb	call
12	object	object	be + predicative	call
13	object	object	verb	call

3.4.2 Transformation Algorithms

We loop read each use case specification event flow, if an event starts with a "[" then read the value of the event flow sentence number and query for transformation rule table based on the value. Use case event flow is parsed to obtain sender, receiver, message, message type and other elements used by sequence diagram of the message tuple. We process the obtained message tuple by considering the following situations:

- If the sender itself is a class name and the "sendMessage" is included in the class, then the message tuple does not need to be changed;
- If the sender is not a class name and only one class includes "sendMessage", then the sender will be replaced with the name of the class which includes the "sendMessage";
- If the sender is not a class name and the "sendMessage" is included in some same classes, the sender will be replaced with the name of the class which includes the "sendMessage" according to "responsibility driven" thought.

The processed message tuple is put into the message sequence. We use message sequence input and generate the final sequence diagram according to the message sequences in the queue. As shown in Algorithm 1. When generating a sequence diagram, we should pay attention to the layout, location and other factors of the object, lifeline, activation, message and other basic components.

Algorithm 1: Transformation from Use Case Diagram to Sequence Diagram.

input:

EventFlowList of all the use case specification event flow

output:

Sequence Diagrams

```
1: foreach EventFlow in EventFlowList{
2:   foreach Event in EventFlow{
3:     if(checkFirstLetter(Event,[' '])){
4:       Value = getValue(Event);
5:       Sender = getSender(Event,Value);
6:       Receiver = getReceiver(Event,Value);
7:       Message = getMessage(Event,Value);
8:       MessageType=getMessageType(Event,Value);
9:       dealWith(Sender, Receiver, Message,
10:      MessageType);
11:      add(sequenceList, Sender, Receiver, Message,
12:      MessageType);
13:    }
14:  }
15:  generateSequenceDiagram(sequenceList);
16: }
```

3.5 Transformation from Use Case Model to State Diagrams

Xin discussed [6] that in the process of transformation from use case model to State Diagram the most important point is to consider how to parse the use case event flow, in order to ensure the one-to-one mapping between the functional requirements of the system described in the use case specification and the elements of the State Diagram such as state, transition and condition decision.

3.5.1 Transformation Approach

We need to transform each use case in the use case diagram into state diagram like the transformation from use case model to Sequence Diagram.

Firstly, the use case specification event flow is parsed. We make rules of transformation from use case model to state diagram by analysing the basic elements of state diagrams and the 13 English sentence patterns, as shown in Table 3. In the process of executing this operation, the flow direction of the decision branch path in different environment should be considered since the state diagram is different from the sequence diagram.

Table 3. State diagram event parsing rules

Syntactic Structure No.	State
1	verb + object
2	verb1 + object1
3	participle + object1
4	be + adjective
5	set + object1
6	verb
7	verb
8	verb + object
9	verb + object
10	verb
11	verb
12	be + predicative
13	verb

We loop read each use case specification event flow, if an event flow starts with "[" then obtain the value of the event flow sentence number and query for transformation rule table based on the value, as shown in Table 3. We extract state element from the event flow and put the element into the state List. If the event flow is "If", "Else", or "EndIf", these conditional statements will be put into the state list directly.

A new state diagram is generate based on the state list. We paint the states in order on the state diagram, new a stack if conditional statement (If) exists, push "If" and other states after "If" onto the stack. When the top element of the stack is "EndIf", we get the position (pos) of the nearest "If" to "EndIf", pop the states between the top and the pos off the stack, and then paint these states in order on the state diagram, as shown in Algorithm 2. When generating State Diagrams, we should pay attention to the layout if the elements on the state diagram to prevent dislocation of the elements.

In [6], there may be a bias in the order of States. In our study during the transformation process, the stack is used to ensure the correctness of the order of States, especially when the selection structure is encountered.

Algorithm 2 The transformation from use case model to state diagram.

Input:

EventFlowList

Output:

state diagram

```

1: foreach EventFlow in EventFlowList{
2:   foreach Event in EventFlow{
3:     if(checkFirstLetter(Event,'[')){
4:       Value = getValue(Event);
5:       State = getSender(Event,Value);
6:       add(stateList,State);
7:     }else{
8:       condition = getContent(Event);
9:       add(stateList,condition);
10:    }
11:  }
12: generateStateDiagram();
13: foreach State in stateList{
14:   if(State!="If"&&State!="Else"&&State!="EndIf")
15:   {
16:     drawState(State);
17:   }else{
18:     Stack stack;
19:     if(State!="EndIf")
20:     {
21:       stack.push();
22:     }
23:     if(State=="EndIf")
24:     {
25:       currentStateList=stack.pop();
26:       drawState(currentStateList);
27:     }
28:   }
29: }
30: }

```

4 Case Analysis

4.1 CIM Modelling

In the scheme of this paper, the business process model and the use case model are used to represent CIM. Therefore, the method is very suitable for the software systems based on business processes. A library management system is a typical and common business based system, which mainly deals with the book lending related business. Based on the above reasons, we take a simplified library management system as an example and analyse the above modelling as well as transformation method. First of all, the Level 0 business process modelling, the simplified library management system contains of "Borrow Management", "Book Management" and "Reader Management", as shown in Fig. 2.

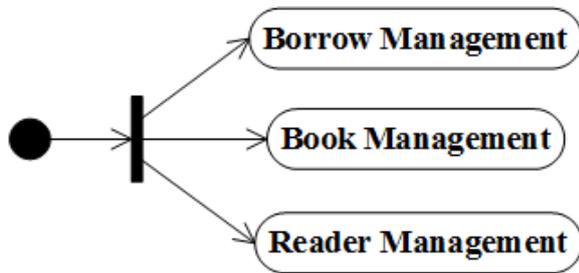


Fig. 2. Level0 business process modelling

In this paper, we dig deep into the “Borrow Management” activity only for the principle is similar. The main business of the business module is: (1) When readers search books by the system, there are two ways to choose: one is searching by book title and the other is by book author; (2) When readers find the book they want to borrow, they need to turn the book over to the librarian and hand borrowing procedure; (3) When the readers return the books, the librarian need to review the time limit of the books. If the time is overdue, readers should pay a fine. The Use Case Diagram of Borrow Management is shown in Fig. 3. The use case specification of Borrow Management is shown in Table 4. Each event of the use case specification is written according to the thirteen English sentences patterns used to describe use cases. The number of the subordinate sentence pattern is added before the sentence. So, it is convenient for the use case event flow analysis. Using these use case specification as input, the automatic transformation from CIM to PIM is carried out.

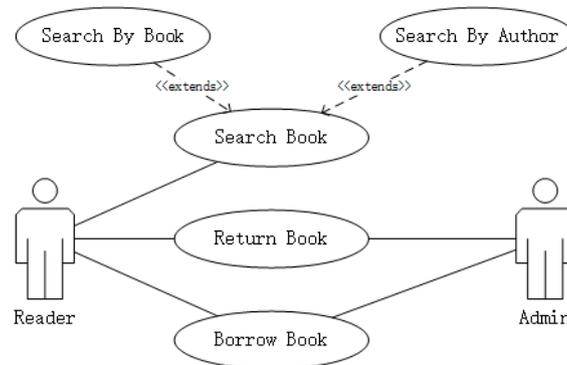


Fig. 3. Use case diagram of Borrow Management

Table 4. Use case specification of Borrow Management

<p>[8]Admin scan readerID of Reader. [8]Admin input readerID into System. [1]System check readerID. If [4]System find readerID invalid. [8]System throw errorMessage to Reader. Else [8]System show info like readerID, readerName, readerBorrowStatus of Reader. [1]System check readerBorrowStatus. If [4]System find readerBorrowStatus invalid. [8]System throw errorMessage to Reader. Else [8]Admin scan bookID of Book. [8]Admin input bookID into System. [1]System check bookID.</p>

```

If
    [4]System find bookID invalid.
    [8]System throw errorMessage to Reader.
Else
    [8]System show info like bookID, bookTitle, bookAuthor,
bookStatus of Book.
    [1]System check bookStatus.
    If
        [4]System find bookStatus invalid.
        [8]System throw errorMessage to Reader.
    Else
        [8]System update bookStatus of Book.
        [1]Reader borrow Book.
    EndIf
EndIf
EndIf
EndIf
EndIf

```

4.2 Transformation from CIM to PIM

We transform CIM into PIM using use case specification as input according to the transformation rule mentioned in Section 3. The transformation from use case specification to class model has been shown in our previous work [60]. In this paper, we will show how use case specification is transformed into Sequence Diagrams and State Diagrams.

4.2.1 Transformation from use case specification to sequence diagram

When a use case model is transformed into a Sequence Diagram, the first case is to loop read use case event stream and get the number of sentences for each event. Then, transformation rule function is called with the number of sentences and the sentence after tagging as parameters. This function extracts the corresponding basic elements from the sentence according to the different sentence numbers, and the specific internal execution algorithm is shown in Algorithm 1. So far, the event flow analysis step of transformation from use case specification to sequence diagram has been completed, and the obtained message sequence is shown in Table 5. The sequence diagram of “Borrow Book” is shown in Fig. 4.

Table 5. Sequence diagram message sequence of “Borrow Book”

NO.	Sender	Receiver	Message	MessageType
1	Admin	Admin	scanReaderID	call
2	Admin	System	inputReaderID	return
3	System	System	checkReaderID	call
4	System	System	beInvalid	call
5	System	Reader	throwErrorMessage	return
6	System	System	showInfo	call
7	System	System	checkReaderBorrowStatus	call
8	System	System	beInvalid	call
9	System	Reader	throwErrorMessage	return
10	Admin	Admin	scanBookID	call
11	Admin	System	inputBookID	return
12	System	System	checkBookID	call
13	System	System	beInvalid	call
14	System	Reader	throwErrorMessage	return
15	System	System	showInfo	call
16	System	System	checkBookStatus	call
17	System	System	beInvalid	call
18	System	Reader	throwErrorMessage	return
19	System	System	updateBookStatus	call

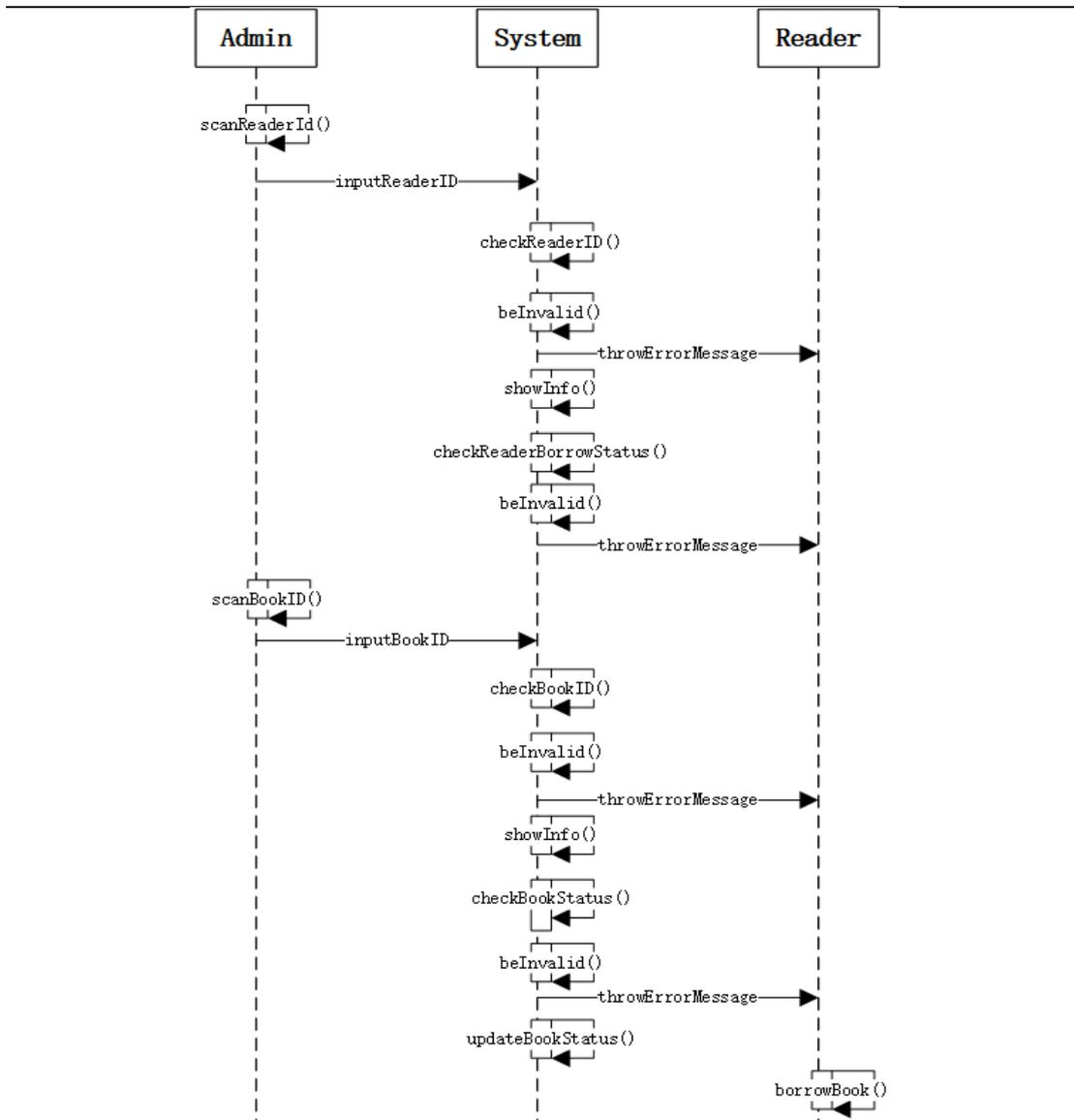


Fig. 4 sequence diagram of "Borrow Book"

4.2.2 Transformation from use case specification to state diagram

When the use case model is transformed into a state diagram, the event stream parsing function is called to process the event stream. Firstly, we loop read each event, mark the word and get the sentence number, but the conditional identification words such as If, Else and EndIf do not need to be marked.

Then, the generating sequence function is called with the input of the sentence number and the marked event. According to the rules of transformation from use case specification to State Diagram, the feature words that can be mapped to the basic elements in the State Diagram are extracted from the event stream, and the list of States is formed. The conditional identifier is directly added to the list of states without processing. Similarly, in the case of "Borrow Book", its status list is shown in Table 6. The state diagram of "Search By Author" is shown in Fig. 5.

Table 6 State list of "Borrow Book"

NO.	State	NO.	State	NO.	State
1	scanReaderID	12	throwErrorMessage	23	If

2	inputReaderID	13	Else	24	beInvalid
3	checkReaderID	14	scanBookID	25	throwErrorMessage
4	If	15	inputBookID	26	Else
5	beInvalid	16	checkBookID	27	updateBookStatus
6	throwErrorMessage	17	If	28	BorrowBook
7	Else	18	beInvalid	29	EndIf
8	showInfo	19	throwErrorMessage	30	EndIf
9	checkReaderBorrowStatus	20	Else	31	EndIf
10	If	21	showInfo	32	EndIf
11	beInvalid	22	checkBookStatus		

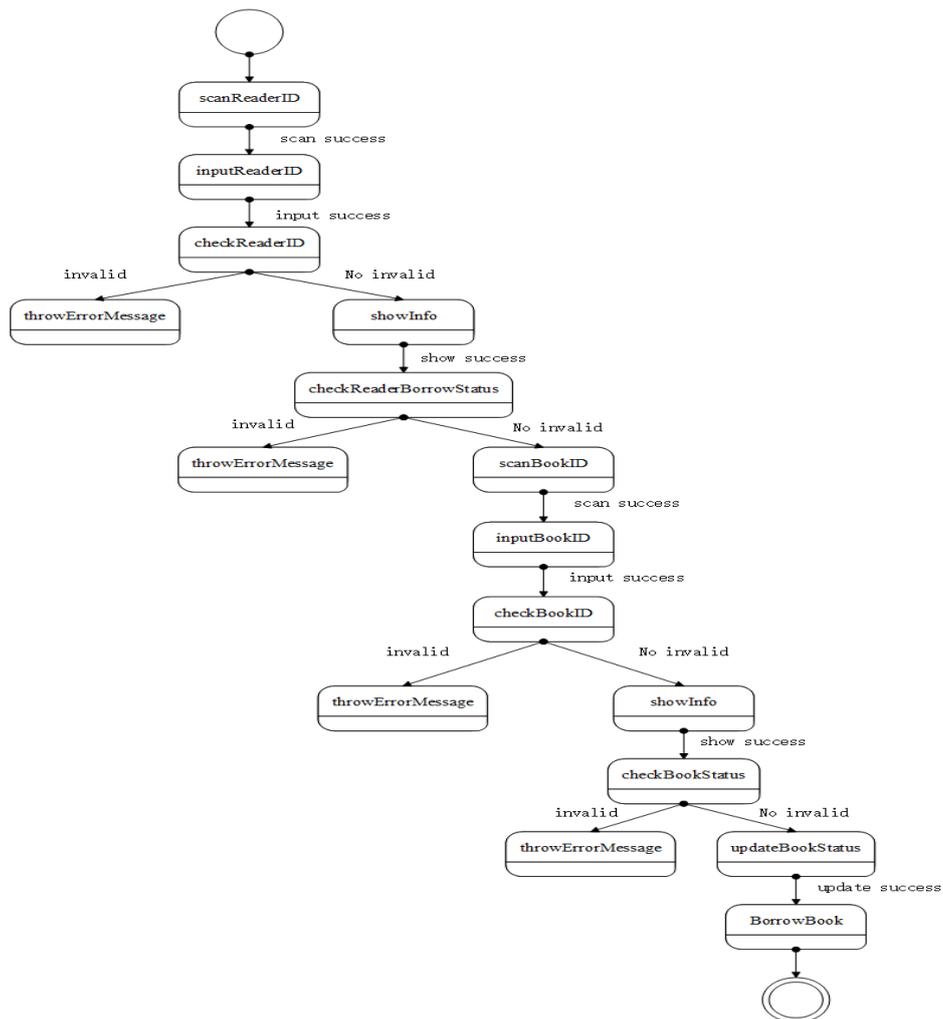


Fig. 5. State diagram of "Search By Author"

4.3 Implementation of A Model Transformation Tool

The code structure of model transformation tool is shown in Fig. 6. "Src" directory stores the core code of the program, including the classes of transformation from use case model to class diagram as well as the transformation of sequence diagram and state diagram, the helper classes and graphical user interface classes. The open source speech tagging tool kit released by the Stanford NLP (Natural Language Processing) Group is used to parse the use case specification event flow. The tool kit is placed in "lib" directory and "models" directory stores the models that have been trained and are used by the tool kit. "UseCaseSpec" directory stores the use case specifications used as the input of the program.

Program interface screenshots are shown in Fig. 7. The Class Diagram of the system exists in the form of XMI document which is of good portability and can be opened in any model tools support XMI. This measure has a good foundation for code automation generation.

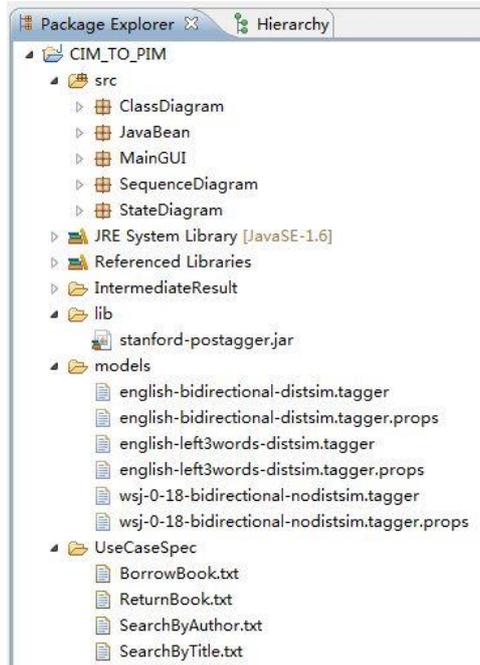


Fig. 6. Code structure

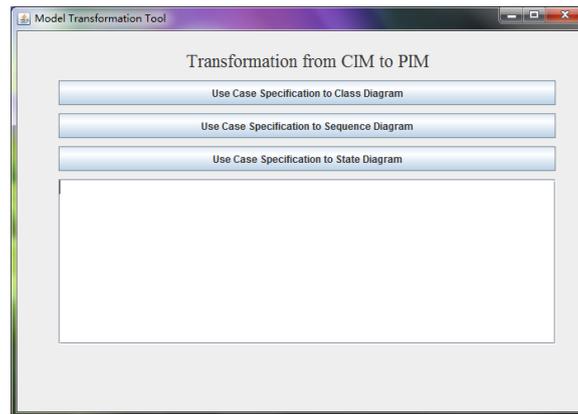


Fig. 7. Program interface screenshots

5 Conclusions

Compared with the related studies, the advantages of our paper are as follows. A relationship oriented CIM-to-PIM automatic transformation approach is proposed. Class Diagram and Sequence Diagram are used to describe PIM. The class diagram describes the static part of the PIM, and the Sequence Diagram and State Diagram describe the dynamic part. The combination of class diagram, sequence diagram and state diagram ensures the integrity of the PIM model. In addition, a relationship oriented automatic transformation approach which describes how use case specification event flow is transformed to class diagram is illustrated. On the basis of the use case description templates and the use case description semi-formal language function-argument matrix is extracted depends on relationship oriented thinking and class model is automatically aggregated by affinity analysis automation algorithm. In addition, during the transformation attributes, operations, and inheritance relationship of classes are well supported.

The comparison of our work with related studies in CIM-to-PIM transformation is shown in Table 7.

Table 7. Comparison of our work with related ones in CIM-to-PIM transformation

CIM-to-PIM transformation approach	Integrity	Automated
Use case event flow based approach	N	Y

meta-model based approach	N	N
feature and component based approach	N	N
QVT (Query/View/Transformation) based approach	N	N
<i>Relationship oriented approach (our paper)</i>	<i>Y</i>	<i>Y</i>

In this article, we proposed a relationship-oriented transformation approach from CIM to PIM, attempting to resolve the problems such as the low degree of automation and the non-support for relationships between classes.

We plan to continue this work in the future. Firstly, CIM modeling method discussed in this paper involves the conversion of CIM's internal models. The transformation from business process models to the Use Case model is performed manually. We will pay more attention to this part by improving the CIM-to-CIM conversion method in order to eliminate the risk brought by the artificial intervention. Secondly, the approach we proposed does not yet support describing the Use Case specification in Chinese. In the process of software development, it is difficult for the developers in the Chinese language system to describe the use cases in English. At present, there are several studies of Chinese sentence analysis which includes the analysis of Chinese grammar and semantics, the Chinese word segmentation, etc. These studies will be referenced in the future.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is partially supported by research and application demonstration of training service support technology for primary and secondary school teachers which belongs to project of national science and technology supporting plan (2014BAH22F00).

References

- [1] Miller J. and Mukerji J., MDA Guide, Version 1.0.1[J], OMG, 2003, 20(3):13.
- [2] Njonko P. B. and Abed W. E., From natural language business requirements to executable models via SBVR, International Conference on Systems and Informatics, 2012:2453 - 2457.
- [3] Debnath N., Leonardi M. C., Ridao M, et al., An ATL transformation from natural language requirements models to business models of a MDA project, IEEE 11th International Conference on ITS Telecommunications (ITST), 2011:633 - 639.
- [4] Huang J., Zhang L., and Gu J., Object-oriented three-layer modelling in MDA. Journal of Computer Applications, 2010, 30(2):385-389.
- [5] Zhang Z. and Zhang G., Software Requirement Modeling Research Based on Use Case, Microcomputer Development, 2004, 14(7):30-32.
- [6] Xin Y., Research on UML Model Transformation based on MDA, Xidian University, China, 2009.
- [7] Jiang L., MDA-based Transformation between UML Models. Xidian University, China, 2009.
- [8] Liu T., Research on Model Transformation from CIM to PIM in MDA, Xidian University, China, 2010.
- [9] Wei Z, Hong M, Zhao H, et al., Transformation from CIM to PIM: A Feature-Oriented Component-Based Approach, Lecture Notes in Computer Science, 2005:248-263.
- [10] Kuang Z. and Wei C., Based on UCM Demand Modeling Techniques, Science and Technology Information, 2013, (15):230-231.
- [11] Fuxman A., Liu L., Mylopoulos J, et al., Specifying and analyzing early requirements in Tropos, Requirements Engineering, 2004, 9(2):132-150.
- [12] Yu E., Towards modelling and reasoning support for early-phase requirements engineering, Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997, 1997:226--235.
- [13] Zheng Y. Research on Requirements Modeling Based on Scenarios, Zhejiang Normal University, China, 2009.
- [14] Jin H. and Huang Q., Requirement formal modeling and quality characteristics verification based on scene behavior, Application Research of Computers, 2016, 05:1-9.
- [15] Cheng M., Huang B., Wu G., Yao Y. and Yuan M., Software Behavior and Multi-Viewpoints Oriented Requirement Modeling Methodology, Journal of Wuhan Univ (Nat.Sci.Ed.), 2014,03:211-218.
- [16] Yao Y., Wu D., Wu H. and Wan L., Design and realization of new software requirements description language, Computer Engineering and Applications, 2009, 45(21):185-88.
- [17] Valles-Barajas F. A., Formal Model for The Requirements Diagrams Of Sysml, IEEE Latin America Transactions, 2010, 8(3):259 - 268.
- [18] Osis J., Asnina E. and Grave A., Computation Independent Modeling within the MDA, IEEE Computer Society, 2007:22-34.
- [19] Karakostas B., Panagiotakis D., and Fakas G., Workflow Requirements Modelling Using XML[J]. Requirements Engineering, 2002, 7(3):124-138.
- [20] Li L. A. Semi-Automatic Approach to Translating Use Cases to Sequence Diagrams, International Conference on Technology of Object-Oriented Languages. IEEE Computer Society, 1999:184.
- [21] Li L., Translating Use Cases to Sequence Diagrams, Proceedings of 15th IEEE international conference on Automated software engineering, IEEE Computer Society, 2000:293.
- [22] Kherraf S, Lefebvre, R. and Suryan W., Transformation from CIM to PIM Using Patterns and Archetypes, 2014 23rd Australian Software Engineering Conference. IEEE, 2008:338-346.
- [23] Cao X., Miao H. and Sun J., An Approach to Transforming from CIM to PIM Using Pattern, Computer Science, 2007, 34(6):265-269.
- [24] Yin J., Research on MDA-CIM modeling and its transformation to PIM, Harbin Institute of Technology, China, 2007.

- [25] Cao X., Miao H. K., and Xu Q. G., Modeling and Refining the Service-Oriented Requirement, 2012 Sixth International Symposium on Theoretical Aspects of Software Engineering. IEEE, 2008:159-165.
- [26] Rodríguez A., Fernández-Medina E. and Piattini M., CIM to PIM Transformation: A Reality[M]// Research and Practical Issues of Enterprise Information Systems II. Springer US, 2008:1239-1249.
- [27] Rodríguez A, Fernández-Medina E. and Piattini M. Towards CIM to PIM Transformation: From Secure Business Processes Defined in BPMN to Use-Cases., *Business Process Management*, 2007:408-415.
- [28] Omg Q. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Final Adopted Specification, 2008.
- [29] White S. A. Introduction to BPMN, IBM Cooperation, 2004.
- [30] Phalp K. T., Vincent J. and Cox K., Improving the quality of use case descriptions: empirical assessment of writing guidelines, *Software Quality Control*, 2007, 15(4):383-399.
- [31] Dennis A. R., Wixom B. H., Tegarden D. Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach[J]. *International Journal of Control*, 2007, v.48 n.5(5):1807-1818.
- [32] Martin J., *Information Engineering Book II: Planning and Analysis[M]*. Englewood cliffs N.J.:Prentice Hall,1989:160,447.
- [33] Rosenberg D. and Stephens M., *Use Case Driven Object Modeling with UML: Theory and Practice*, Apress, 2007.
- [34] Michael B. and James R., *Object-Oriented Modeling and Design with UML*, Beijing Posts and Telecom Press, China, 2006.
- [35] Njonko P. B. F. and Abed W. E., From natural language business requirements to executable models via SBVR, *International Conference on Systems and Informatics*, 2012:2453 - 2457.
- [36] Debnath N., Leonardi M. C., Ridao M., et al., An ATL transformation from natural language requirements models to business models of a MDA project, 11th IEEE International Conference on ITS Telecommunications (ITST), 2011:633 - 639.
- [37] Zou S., Peng Y., Guo Z., Liu C., Zhou T., Wei L. and Gu A. Case Study of Acquiring Formal Software Requirement, *Modern ElectronicsTechnique*,2009, 32(12):45-48.
- [38] Gao R, Miao H. and Chen Y., An Approach to Capturing Formal Functional Requirements, *Computer Applications and Software*, 2004,21(10):4-6.
- [39] Tang Y., *Research of Use Case Modeling Based on Petri Net*, Southwest University, China, 2006.
- [40] Abbott R. J., Program design by informal English descriptions, *Communications of the ACM*, 1983, 26 (11) , 882–894.
- [41] Chen P., Entity-relationship modeling: historical events, future trends and lessons learned, *Software Pioneers: Contributions to Software Engineering*, Springer-Verlag, New York, USA, pp. 2002:296–310.
- [42] Chen P., English sentence and entity-relationship diagrams, *Information Sciences*, 1983, 29 (2–3), 127–149.
- [43] Beck K. and Cunningham, W. A., Laboratory for teaching object-oriented thinking, OOPSLA: Conference on Object-Oriented Programming Systems, Languages and Applications , New York, NY, USA, ACM , 1989.
- [44] Gomez F, Segami C. and Delaune C. A., System for the semiautomatic generation of E-R models from natural language specifications, *Data and Knowledge Engineering*, 1999, 29 (1), 57–81.
- [45] Wahono R. S. and Far, B. H., OoExpert: distributed expert system for automatic object-oriented software design, 13th Annual Conference of Japanese Society for Artificial Intelligence, 1999:456–457.
- [46] Mich L. and Garigliano, R., NI-oops: a requirements analysis tool based on natural language processing, Third International Conference on Data Mining Methods and Databases for Engineering, Southampton, UK, 2002, WIT Press, pp. 321–330.
- [47] Meziane F. and Vadera S., Obtaining E-R diagrams semi-automatically from natural language specifications, Sixth International Conference on Enterprise Information Systems, Universidade Portucalense, Porto, Portugal, 14–17, 2004, pp.2004:638–642.
- [48] Harmain, H. M. and Gaizauskas R., CM-Builder: a natural language-based CASE tool for object-oriented analysis., *Automated Software Engineering*, 2003, 10(2),157–181.
- [49] Hartmann S. and Link S., English sentence structures and EER modeling, APCCM '07: Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling, Darlinghurst. Australian Computer Society, 2007: 27–35.
- [50] Giganto R., Generating class models through controlled requirements, NZCSRSC-08, New Zealand Computer Science Research Student Conference Christchurch, New Zealand, 2008.
- [51] Al-Safadi L. A. E., Natural language processing for conceptual modeling. *International Journal of Digital Content Technology and its Applications*, 2009, 3(3), 47–59.
- [52] Elbendak M., Vickers P. and Rossiter N., Parsed use case descriptions as a basis for object-oriented class model generation, *The Journal of Systems and Software*, 2011, 84(2011),1209-1223.
- [53] Sharma A. and Sood M., Incorporating MDA to Design Business Intelligence Services by using SaaS Model of Cloud Computing, *International Journal of Computer Applications*, 2014, 106(16):1-8.
- [54] Atkinson C., Bostan P. and Draheim D., Foundational MDA Patterns for Service-Oriented Computing, *Journal of Object Technology*, 2015, 14(1).
- [55] Curcin V. and Woodcock T., Poots A. J., et al., Model-driven approach to data collection and reporting for quality improvement, *Journal of Biomedical Informatics*, 2014, 52:151-162.
- [56] Buchmann T., Dotor A. and Westfechtel B. MOD2-SCM: A model-driven product line for software configuration management systems, *Information and Software Technology*, 2013, 55(3):630-650.
- [57] Wang Z. and Chalmers K. Evolution Feature Oriented Model Driven Product Line Engineering Approach for Synergistic and Dynamic Service Evolution in Clouds: Pattern Data Structure, 7th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), IEEE Computer Society, 2013:471-474.
- [58] Xiao P., *An MDA Based Modeling and Implementation for Web App*, Zhejiang University, China, 2013.
- [59] Zhang C., *Research and Implementation of The Model-Driven Development Methods for Automotive Electronic OS*, University of Electronic Science And Technology Of China, 2013.
- [60] Chen X, Zhou D, and Wu L., Theory of automatic transformation from Computation Independent Model to Platform Independent Model for

MDA, Engineering Technology and Applications, 2014.

Dongdai Zhou graduated from Changchun University of Science and Technology (CUST), China in 1992. He received the M. Sc. Degree from CUST in 1997 and the Ph. D. degree from the Jilin University, China, in 2001. He is currently a professor at School of Software, Northeast Normal University, China. His research interests include software architecture and software code auto-generation, especially the architecture design and codeless development of e-learning software.

E-mail: ddzhou@nenu.edu.cn

Xue Chen graduated from School of Software, Northeast Normal University (NENU), China in 2012. She received M. Sc. Degree from NENU in 2015. She has been engaged in software engineering related work. Currently, she is a senior test engineer at Baidu Online Network Technology (Beijing) Co., Ltd. AI test department. Her research subject is software architecture.

E-mail: chenx206@nenu.edu.cn

Qiuzhi Jin graduated from School of Software, Northeast Normal University (NENU), China in 2014. She received the M. Sc. Degree from NENU in 2017. She has been engaged in software development related work. Currently, she is a software development engineer at Hai Kang Wei Shi Co., Ltd. Her research interests include software architecture and software product line.

E-mail: jinqz276@nenu.edu.cn

Zhejun Kuang received his B.E degree in computer science and information system in 2008 from Massey University of New Zealand. He obtained his M.Sc. degree and Ph.D., both from the College of Computer Science and Technology, Jilin University, China, in 2011 and 2014, respectively. He is currently a postdoctoral researcher at School of Software, Northeast Normal University, China. His interests are in cyber physical system, internet of things and data mining.

E-mail: kuangzj100@nenu.edu.cn

Hongji Yang received the B. Sc. and M. Phil. Degrees from Jilin University in 1982 and 1985, respectively, and the Ph. D. degree from Durham University, UK in 1994. He was a lecturer at Jilin University, China from 1985, and a lecturer and professor at De Montfort University, UK from 1993 and since 2013 he is a professor in the Centre for Creative Computing at Bath Spa University, UK. He has published about 400 refereed journal and conference papers. His research interests include software engineering, creative computing, web and distributed computing. He became an IEEE Computer Society Golden Core member since 2010. Also, he is a member of EPSRC Peer Review College since 2003. He is the editor in chief of the International Journal of Creative Computing. E-mail: h.yang@bathspa.ac.uk