Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

Specification of e-business process model for PayPal online payment process using Reo

Min Xie

REPORT SEN-E0510 SEPTEMBER 2005

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

Specification of e-business process model for PayPal online payment process using Reo

ABSTRACT
E-business process modeling allows business analysts to better understand and analyze the business processes, and eventually to use software systems to automate (parts of) these business processes to achieve higher profit. To support e-business process modeling, many business process modeling languages have been used as tools. However, many existing business process modeling languages lack (a) formal semantics, (b) formal computational model, and (c) an integrated view of the business process being modeled. In this paper, we assess the effectiveness of the Reo coordination language as a business process modeling language. We present a specification of PayPal online payment business process model using Reo and evaluate Reo according to the criteria of e-business process modeling with respect to (a) language expressiveness, (b) visual notation and language semantics, (c) analysis and reasoning, (d) simulation and execution.

1998 ACM Computing Classification System: J.4, J.1, D.1.7, D.2.1
Keywords and Phrases: Reo, coordination languages, business process modeling, online payment
Note: The author carried out this work at CWI as an assignment for a Masters degree in ICT in Business from Leiden University.

SPECIFICATION OF E-BUSINESS PROCESS MODEL FOR PAYPAL ONLINE PAYMENT PROCESS USING REO

by

Min Xie

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

Master of Science in ICT in Business

Leiden University

2005

Approved by _____
Chairman of Supervisory Committee

_____

_____

_____

Programme Authorised
to Offer Degree _____

Date _____

Abstract

E-business process modeling allows business analysts to better understand and analyze the business processes, and eventually to use software systems to automate (parts of) these business processes to achieve higher profit. To support e-business process modeling, many business process modeling languages have been used as tools. However, many existing business process modeling languages lack (a) formal semantics, (b) formal computational model, and (c) an integrated view of the business process being modeled. In this paper, we assess the effectiveness of the Reo coordination language as a business process modeling language. We present a specification of PayPal's online payment business process model using Reo and evaluate Reo according to the criteria of e-business process modeling with respect to (a) language expressiveness, (b) visual notation and language semantics, (c) analysis and reasoning, (d) simulation and execution.

Acknowledgements

First of all I would like to thank my parents for making all this possible for me.

Thanks to my supervisor Prof. Farhad Arbab for giving me the opportunity to work on this topic.

Special thanks to my supervisor Dr. Nikolay Diakov for the advice and support for my thesis. I am very grateful for the comments and time given by him, which have greatly improved and clarified this work.

Finally, but not least, thanks to my classmate Yongzhi Li for the discussion of Reo which helps clarify my ideas.

Contents

Chapter 1
Introduction

1.1 Background

E-business process modeling aims to capture the features and the underlying structure of the business processes in an organization or among organizations. It allows business analysts to better understand and analyze the business processes, and eventually to use software systems to automate (parts of) these business processes to achieve higher profit.

To support e-business process modeling, new business process modeling languages have been developed, such as Business Process Modeling Notation (BPMN) [2]. Traditional software development modeling languages, such as the Unified Modeling Language (UML), have also been pushed into e-business process modeling chores [2], mainly because of their extensive tool support. They are applied to model an e-business process by specifying an e-business system in the context of the business process. In this way, the logic and the rules of the business process is reflected in that system, so that the system (or parts of it) can be implemented using a software solution.

1.2 Problem statement

Existing business process modeling languages have several problems:

- Lack of formal semantics, which leads to the ambiguity in a business process model. [25][26][27]
- Lack of a formal computational model so that it is difficult to simulate or run the business process model.[28][29]

- Software modeling languages used as business process modeling languages often lack an integrated view, in which all diagrams fit together, and thus leads to traceability problem. [4][17][18]

In this thesis work, we will focus on accessing Reo as a suitable business process modeling language for addressing these problems.

1.3 Reo

Reo is a connector-oriented coordination language. It allows the compositional construction of coordinators called “connectors” that can coordinate complex process, especially for concurrent processes through their composition, cooperation and communication. Reo offers the following:

1. A comprehensive visual notation;
2. Formal computational model that defines the rules for implementing or simulating Reo connectors [5].
3. Formal semantics based on a coinductive calculus of flow[6][7][8] and (alternatively) on constraint automata[8];
4. A serialization of its visual notation in XML validated by XML Schema, for interoperability among design and analysis tools.

1.4 Goal

In this thesis work, our goal is to investigate the effectiveness of Reo in e-business process modeling.

1.5 Approach

Step1: We start with a literature study on business process modeling in order to acquire knowledge about business process concepts and business process modeling.

Step2: To assess the effectiveness of Reo as a business modeling language, we first study requirements for business modeling language from various research articles [24][25][26][27].

Step 3: We model an on-line payment process to gain hands-on experience of using Reo as a business modeling language.

Step 4: Based on the requirements from step 2 and the case study, we analyze the effectiveness of Reo in e-business process modeling.

1.6 Structure

The rest of the thesis is organized as follows. Chapter 2 provides an overview of relevant business process concepts, e-business process modeling, and the Reo language. In chapter 3, we use the Reo coordination language to model an online payment process as a case study. In chapter 4, we summarize requirements for a business process modeling language. Then based on the case study, we use these requirements to evaluate the Reo coordination language as a business process modeling language. In chapter 5, we present our conclusions.

Chapter 2
Concepts and Terminology

In this chapter, we look at the concepts of business process and business process modeling. Then we introduce Reo language that we will use to model e-business process.

2.1 E-business process modeling

2.1.1 What is business process?

Business process is defined in different contexts. Bill Curtis defined a process as a partially ordered set of tasks or steps undertaken towards a specific goal [9]. Glossary [11] defines business process as “the collection of related, structured activities--a chain of events--that produce a specific service or product for a particular customer or customers”.

When analyzing these definitions, we notice the following reoccurring themes: 1) a business process has separate parts, as so-called steps, activities or tasks; 2) there are relationships between these parts and these parts are connected through certain order, structure, or coordinating mechanism.

2.1.2 Business process modeling

Business process modeling is a means of capturing a business process, including the activities, their connections, coordinating mechanisms and the underlying structure of the business process. “A business process model enables a business to document,

share, implement, measure the success, and continually improve a business process". [1]

With the advent of E-business, business process and information technology become closely inter-related. "On the one hand, the choice of a particular way of conducting business in an organization will influence the design and structure of the information systems to support this process. On the other hand, advances in information technology can generate completely new opportunities for organizations and thus influence the design of specific business process layouts."[24] In this context, the term business process modeling, or e-business process modeling, has been used to "incorporate all business process properties relating to the transformation of knowledge about information systems into models that describe the processes performed by organizations"[30]. It is useful in the following aspects [12]:

1) Describing a process: to make the process understandable for various target audiences (humans and machines).

2) Analyzing a process: to make it easy to see the structure, properties of the process and facilitate identifying problems, such as process redundancy, and contributes to the business process reengineering, improvement and optimization.

3) Enacting a process: either for simulation purposes or to provide some level of support for process execution.

2.2 The Reo coordination language

Reo is a connector-based coordination language. From the point of view of Reo, a system consists of a set of component instances, communicating through connectors that coordinate their activities. Connectors are built compositionally out of simpler ones, where channels constitute the atomic connectors. In this section, we introduce the main and concepts of Reo: component instance, connector, node, Reo operations and component encapsulation.

2.2.1 Component instance

A component instance is a non-empty set of active entities (e.g., processes, agents, threads, actors, etc.) whose only means of communication with the entities outside of this set is through input and output operations that these entities perform on connected channel ends [23]. Reo completely abstracts away from the details of the communication within a component instance and focuses only on inter-component-instance communication.

2.2.2 Connector

A channel has precisely two directed channel ends: source and sink. A source end accepts data into its channel; a sink end dispenses data out of its channel. Each channel has predefined semantics. Figure 2.1 shows the channels that we use in our modeling work.

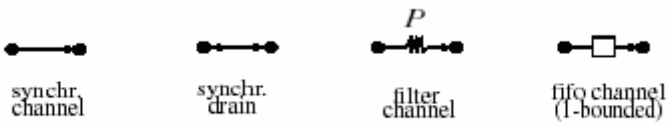

Fig. 2.1 Examples of Reo channels

1. Sync

Sync channel has a source and a sink. The pair of I/O operations on its two ends can succeed only simultaneously.

2. SyncDrain

SyncDrain has two source ends. The pair of input operations on its two source ends can succeed only simultaneously. All data items written to this channel are lost.

3. Filter (pat)

Filter channel has a source and a sink. It behaves as a Sync channel except that it will lose all the data that doesn't match the pattern pat.

4. FIFO1

FIFO1 has a source end and sink end, and contains buffer of size one. Writing a message to FIFO1 succeeds on the source end only when its buffer is empty.

Channels can be composed into complex connectors. A connector is modeled as a graph of nodes and edges, where 1) every channel end coincides on exactly one node; 2) zero or more channel ends coincide on every node; 3) an edge exists between two nodes if and only if there is a channel whose ends coincide on those nodes. Through composition of channels, various coordination mechanisms can be realized.

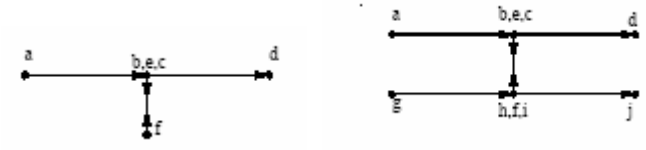

Fig. 2.2 Examples of Reo connectors

Figure 2.2 shows two examples of Reo connectors composed out of Reo channels. The first one allows an input from the node "a" to "d" only when there is also an input from the node "f" at the same time; the second one allows an input form the node "a" to "d" only when there is an input from the node "g" to "j" simultaneously.

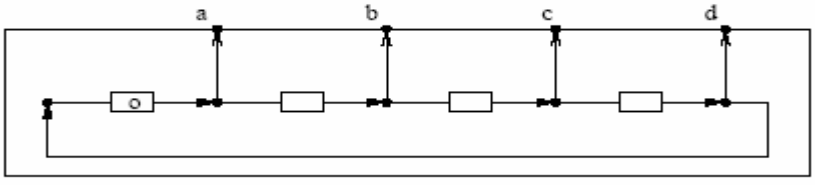

Fig. 2.3 Sequencer

Figure 2.3 shows an example of a more complex connector, called "sequencer", which allows a series of inputs following the sequence of a, b, c, d.

2.2.3 Node

Reo has three types of nodes: *source node*, *sink node* and *mixed node*. A source node contains only source channel ends. A sink node contains only sink channel ends. A mixed node contains both source and sink channel ends.[22][32]

Source node & Sink node

A component can only be connected to a (set of) source channel ends through a source node, or to a (set of) sink channels ends through a sink node. If a component writes a data item to a source node, the node replicates the data item to all of its source channel ends only when all of them can accept the data item. When a component tries to take a data item from a sink node, the node non-deterministically selects a data item available at one of its sink channel ends.

Mixed node

Source channel end(s) and sink channel end(s) can coincide on a mixed node. The node non-deterministically selects a data item available at one of its sink channel ends and replicates the data item to all of its source channel ends only when all of them can accept the data item.

2.2.4 Reo operations

A component instance can perform operations on a channel end. Reo defines two types of operations [32][22]:

1) topological operations that allow manipulation of connector topology;
2) I/O operations that allow data input and output.

Here we only introduce those operations that will be used in our modeling work (Table 2.1).

Topological Operation	Description
Join	Join two nodes identified by two channel ends, coincident with the nodes respectively
Split	Split a node into two nodes by specifying the channel ends that the performer requires to coincide on the new nodes

I/O Operation	Description
Read	read a data item from a sink without removing the data
Take	read and remove a data item from a sink
Write	write a data item to a source

Table 2.1 Reo operations

2.2.3 Component encapsulation

In analogy with electrical circuits, a design is called a “circuit” in Reo[22]. When designing large circuits, we can abstract away the details of the circuit of a particular connector and encapsulate it into a new component, in order to instantiate it and reuse it. In a visual notation, the encapsulated component is represented as a box. On its border, a set of nodes are positioned as the component’s ports, which its internal circuit exposes to the outside. The internal behavior is entirely defined by Reo.[22]

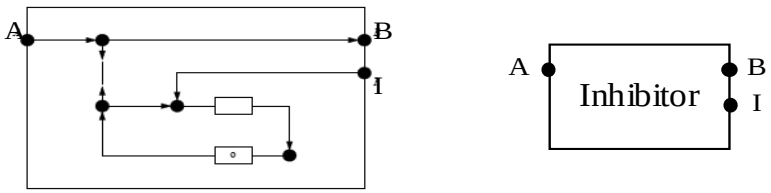

Fig. 2.4 Inhibitor

Figure 2.4 shows an example that how we define the Inhibitor [34] and how we depict an instance of it.

Chapter 3
Case Study:
PayPal's Online Payment Process Modeling

In this chapter, we use Reo to model the online payment business process of the PayPal system. First we give an overview of the PayPal' online payment process. Then we describe the business protocol which defines the online payment process. After that, we model the online payment process by analyzing and implementing the business protocol.

3.1 Overview

An online payment process is achieved through connectivity over the internet between online customers, merchants, buyers, sellers, and the financial networks that move money between them. The PayPal system provides a novel process which has effective mechanisms to guarantee a business transaction. Its process has gained more than 10 million dollars and continuously growing community of customers. Now PayPal transactions constitute over 10% of all Internet traffic in the financial services category, more than Citibank, Wells Fargo and Bank of America combined [33]. For that reason, we choose to model PayPal in our case study.

3.2 Business protocol

To model the PayPal's online payment business process, we define its business protocol and related roles:

3.2.1 Roles

- A user. We distinguish two kinds of users:
  1. Sender – the party that places a money transfer order;
  2. Receiver – the party that acts as the recipient of money as described in a money transfer order;
- The Server – the PayPal system that makes possible money transfers from a sender to a receiver over the Internet.

3.2.2 Business protocol

We summarize the protocol as described in [36].

1. Before using the PayPal system, both the sender and the receiver must authenticate themselves to the system. Both the sender and the receiver can do this by registering an account and then logging in the PayPal system.

2. After authentication, the sender can place a money transfer order in the PayPal system by providing needed information such as the shipping address, the receiver's e-mail address, amount and the way of payment, either by using an existing PayPal account or valid credit card. The PayPal system sends an email to the receiver's email address to notify him about pending money transfer orders. If the PayPal system already contains an account with email address equal to the receiver's e-mail address from the money transfer order, the PayPal system transfers the money to the receiver's account in "pending" status and waits for the receiver's confirmation. Otherwise, the PayPal system waits for the receiver to register an account and claim the pending money transfer order.

3. After a receiver logs in the PayPal system, he/she can accept or reject a money transfer order from the "pending" money transfer orders. Accepting a money transfer order resets the "pending" status of the transfer and completes a transaction between the sender, the receiver and the PayPal system; Rejecting a money transfer order causes PayPal system to return the money to the sender's account;

4. The Paypal system removes any money transfer orders that a receiver has not claimed within 30 days from their date of placing in the system;

5. A sender can cancel a pending money transfer order;

6. The sender can claim a money transfer back within 30 days from the date of the completion of the transaction, if and only if the sender has used the money transfer order to pay for a physical product that the receiver must ship (with the information of valid shipping address) to the sender and the receiver accepted the money transfer order, under the conditions that:
    i. The sender didn't claim money on this transaction before;
    ii. The sender has filed less than two claims in current calendar year;
    iii. The date of the claim is within 30 days from the date that the receiver accepted the money;
    iv. The PayPal legal authority has approved the sender's claim as legal.

    If the transaction is no more than $500, the system transfers the money from receiver's PayPal account to the sender's PayPal account. If the receiver's PayPal account has insufficient funds, the sender still gets the refund in his/her PayPal account, and the PayPal system blocks the receiver's account until the receiver pays the proper amount to PayPal.

If the transaction is for more than $500, the sender only gets $500 in his/her PayPal account immediately. The PayPal system then blocks the receiver's account until he/she settles the claim.

7. A receiver can play the role of a sender to return the money he owes to a sender of a transaction that the sender has claimed back.

3.3 Analysis

We use a top-down approach to analyze this protocol. At the top level, we make a use case diagram to capture the functionality of the PayPal system (Fig. 3.1).

3.3.1 Use case diagram

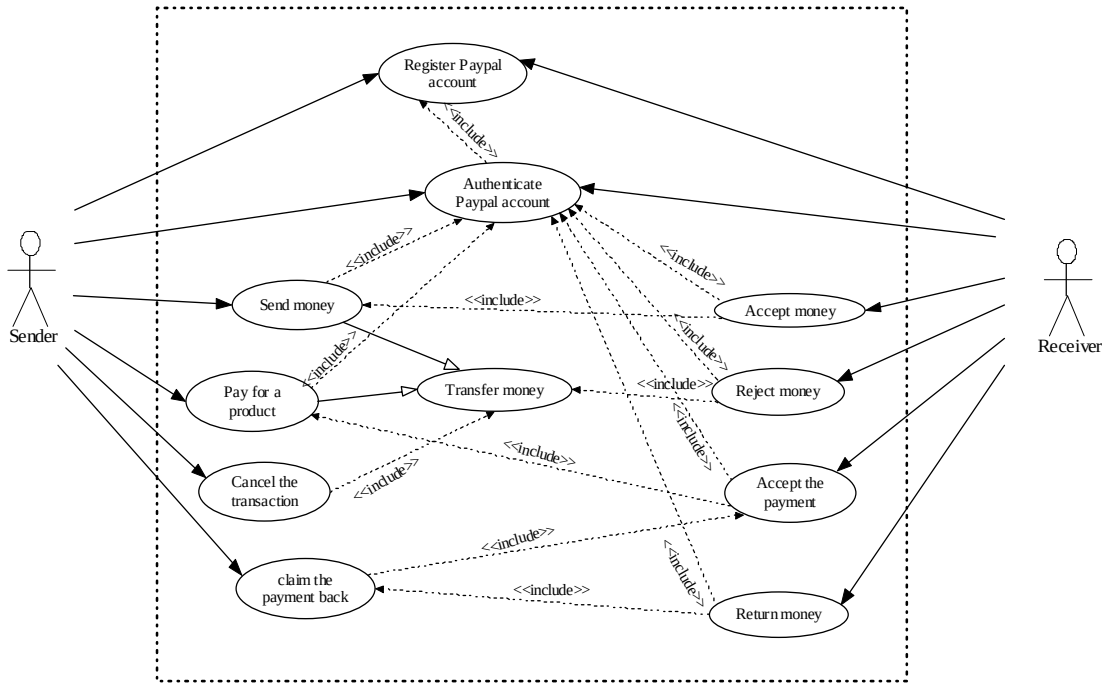

Fig. 3.1Use Case Diagram

The system offers interface to the sender actor and the receiver actor. In the use case diagram, each use case identifies an activity that actors want to perform.

The sender can register a PayPal account, “send money” and “pay for a product”, which are the two specified cases of the use case “transfer money”. The sender can also “cancel the transaction” and “claim the payment back”. All these use cases can happen after authentication (“authenticate PayPal account”), as the system requires the sender to be registered.

For the receiver’s use cases, the “register PayPal account” and “authenticate PayPal account” are the same as the sender’s activities. The receiver can also “accept money” or “accept the payment”, “reject money” and “return money” (owed to the sender of a claimed transaction).

The time dependencies are also shown on the use case diagram. The receiver’s acceptance and rejection can happen only when there is some transfer from a sender. The sender can claim only after he has sent a payment and the payment has been accepted by a receiver. Dependencies in the use case diagram are also the guidance when implementing the PayPal’s online payment protocol in Reo.

3.3.2 High-level Structure

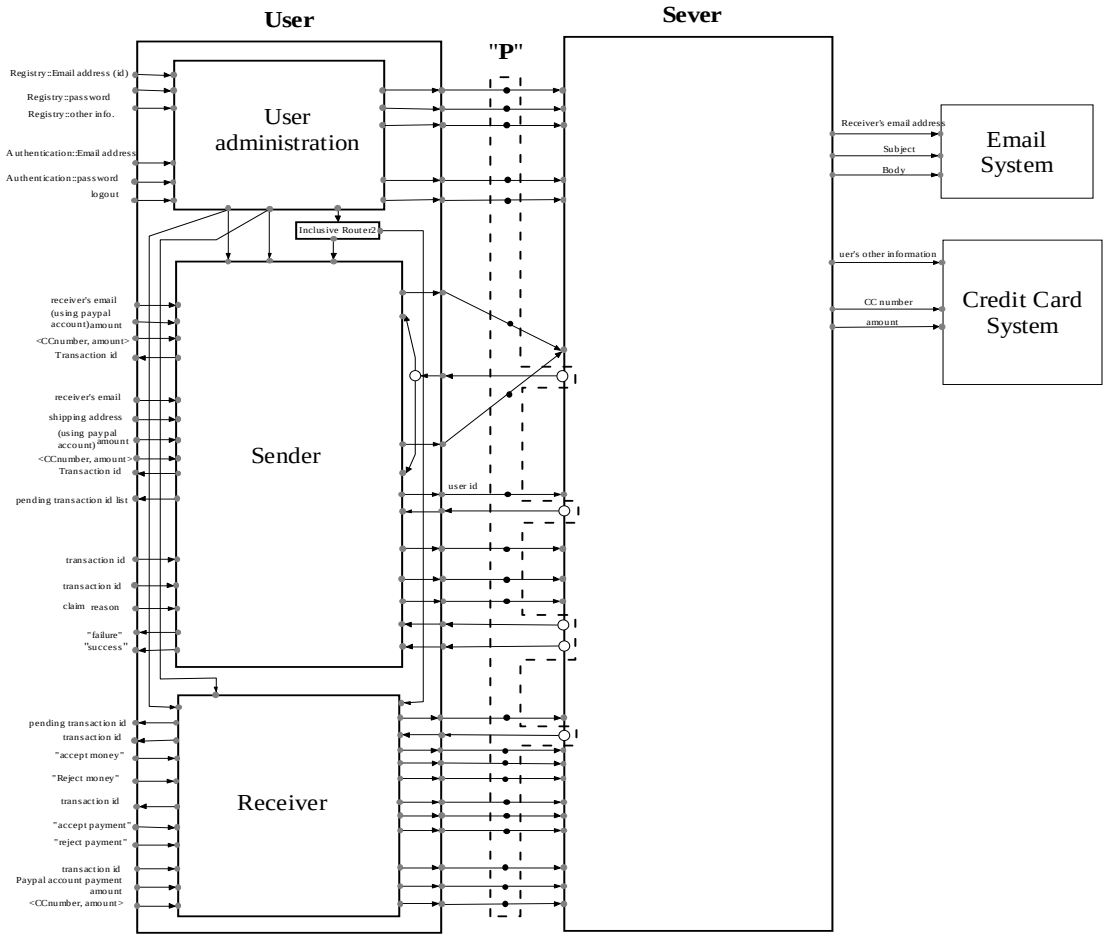

Fig. 3.2 High level structure

The high level structure of the specification (Fig. 3.2) contains two primary parts, as the first level components, the “User” and the “Server”. In addition, there are two external systems interacting with “Server”: the “Email system” and “credit card system”. In the “User” component, there are three sub-components:1) the “user administration” sub-component deals with user registration and authentication; 2) the “sender” sub-component deals with functions and operations of the user when acting as sender, and 3) the “receiver” sub-component concerns the functions of a receiver’s role. The “Server” component deals with data checking and processing. Many “Users” can attach to the “Server” through the nodes and exclusive routers designated in the “P” region.

3.4 Construction

Using the high level model, we construct the specification of the online payment process protocol. For the understanding purpose, we first introduce the basic components and encapsulated connectors that we use in the modeling; then we present our designed components; finally we present the overall online payment process specification in Reo circuit.

3.4.1 Basic components

In addition to Reo primitive channels, thirteen basic components are used for operating data passing through channel, shown in Fig.3.3.

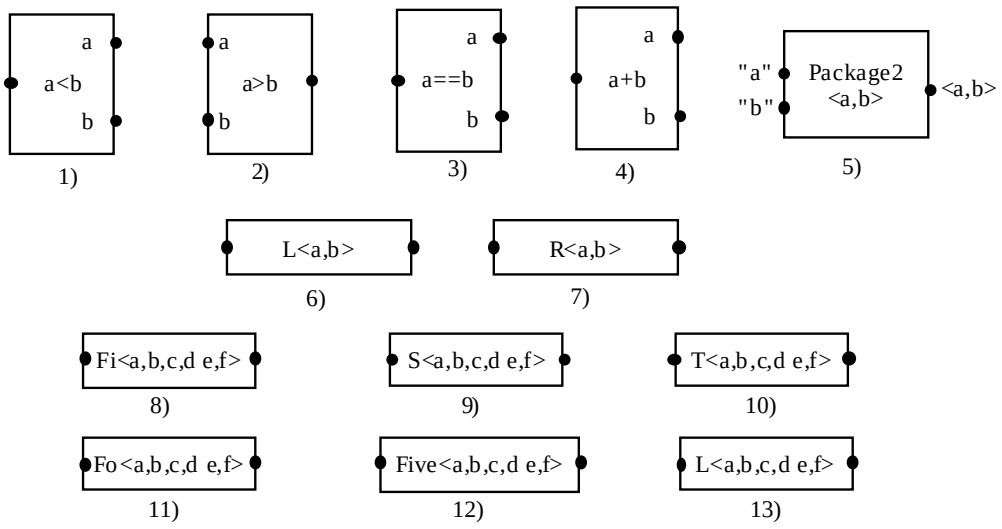

Fig. 3.3 Basic components

The first component has two source nodes for inputs, labeled by “a” and “b”, and a sink node for output. When two messages “a” and “b” representing two integers are input through the two source nodes, if a<b the output on its sink node will be true and otherwise false. The second component is the same as the first one except that when a>b the output on its sink node will be true and otherwise false. The third one is the same kind of the first one and the second one except that when a==b the output on its sink node will be true and otherwise false. The forth one is a little different. Its output on the sink node gives the sum of “a” and “b”. The fifth one, a “packager 2”, takes two inputs and generates a package <a, b>. The “package 2” can be parameterized to “package N”.

The sixth component has one source node on the left and one sink node on the right. The input from its source is a pair of data in the form of < a, b >. The component outputs the first element of the input pair on its sink node; in this case “a” is outputted. The seventh component outputs the last element of the input pair on its sink node, in case “b”. Similar to the sixth and seventh component, each of the rest components receives a tuple of data elements on its source node, and outputs the first, second, third, forth, fifth and last of the element in the tuple on its sink respectively.

One can define the behavior of these components algebraically [7].

3.4.2 Encapsulated connectors

We also use a set of encapsulated connectors, shown in Fig.3.4.

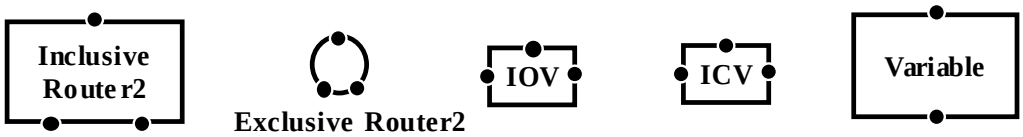

Fig. 3.4 Encapsultated connectors

1. Inclusive Router

The Inclusive Router has one input node and two or more output nodes. It can route synchronously its input to less than or equal to its total outputs.

2. Exclusive Router

The Exclusive Router has one input node and two or more output nodes. It routes synchronously its input to precisely one of its outputs. In our specification, we use a hollow circle to present it for the convenience.

3. Valves

There are two kinds of valves regulating the flow of data, initially opened valve (IOV) and initially closed valve (ICV). The IOV initiate allows data to flow and the ICV doesn’t. Both valves offer a node through which one can toggle a valve’s state from “opened” to “closed” or the other way around.

4. Variable

The Variable offers a source node and a sink node. It always accepts a message written on its source and the last message written represents the value of the variable. The variable always offers a message containing its value to anyone that makes a take operation on its sink [35].

One can find the definitions of these connectors in [34].

3.4.3 Designed components

1. User administration component

User administration component (Fig.3.5) constitutes two sub-components: the Registration component and the Authentication component. The Registration component has the responsibility to allow valid registering in the system. The Authentication component lets only registered users to login to the system. When a user successfully logs in the system, the user administration component outputs the data of the user's "email address" and "login" signal for sender component and receiver component to use. When a user logs out the system, the "logout" signal is outputted to disable the work of the sender and the receiver components.

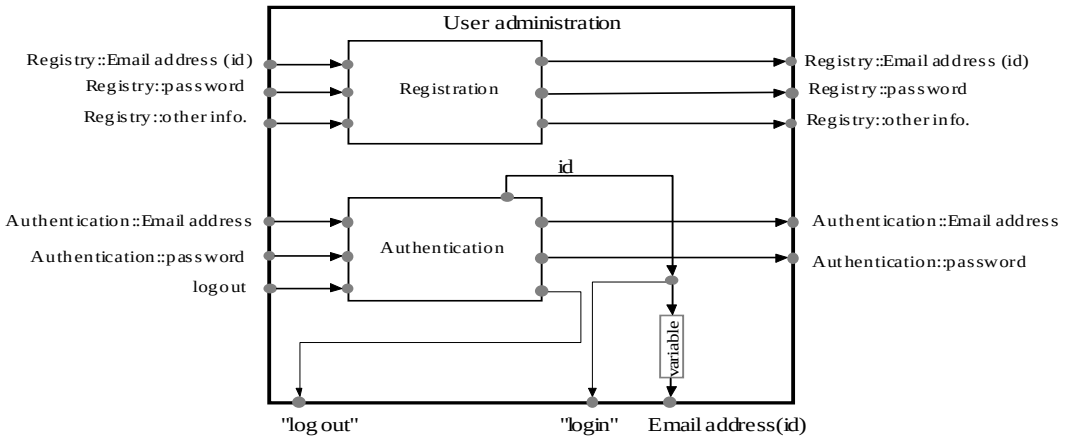

Fig. 3.5 User administration component

1) Registration Component

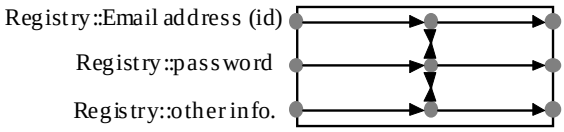

Fig. 3.6 Registration component

Registration component (Fig.3.6) has three input nodes and three output nodes. It requires three synchronous input messages. The three “write” operations corresponding to these messages on the three input nodes to succeed only when three “take” operations succeed at the same time on the outputs.

2). Authentication Component

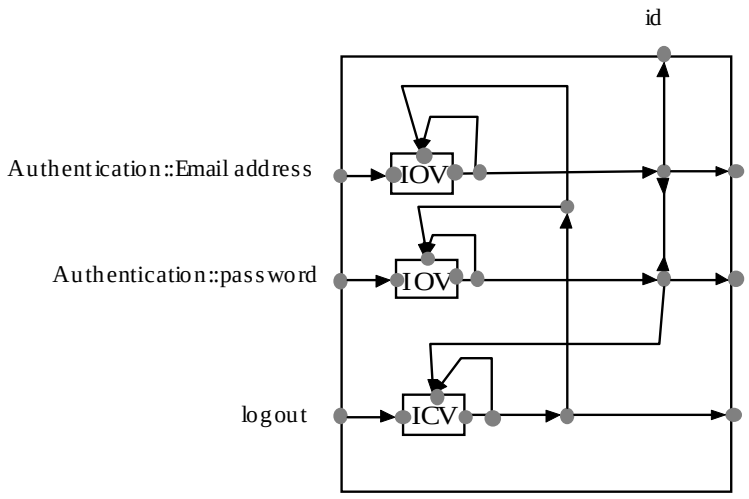

Fig. 3.7 Authentication component

Authentication component (Fig.3.7) has three input nodes. We use “IOVs” to allow the message writing on the input nodes only once. Meanwhile, the “ICV” is opened to allow an input for user logout. When there is an input from “logout” input node, the signal will be passed to disable any operation of the other components and open again the upper two “IOVs” for receiving new authentication information.

2. Sender component

Sender component (Fig.3.8) constitutes six sub-components: “Send money” component, “Send payment” component, “Information for senders” component, “Cancel sending” component, “Claim payment” component and “ID tester” component.

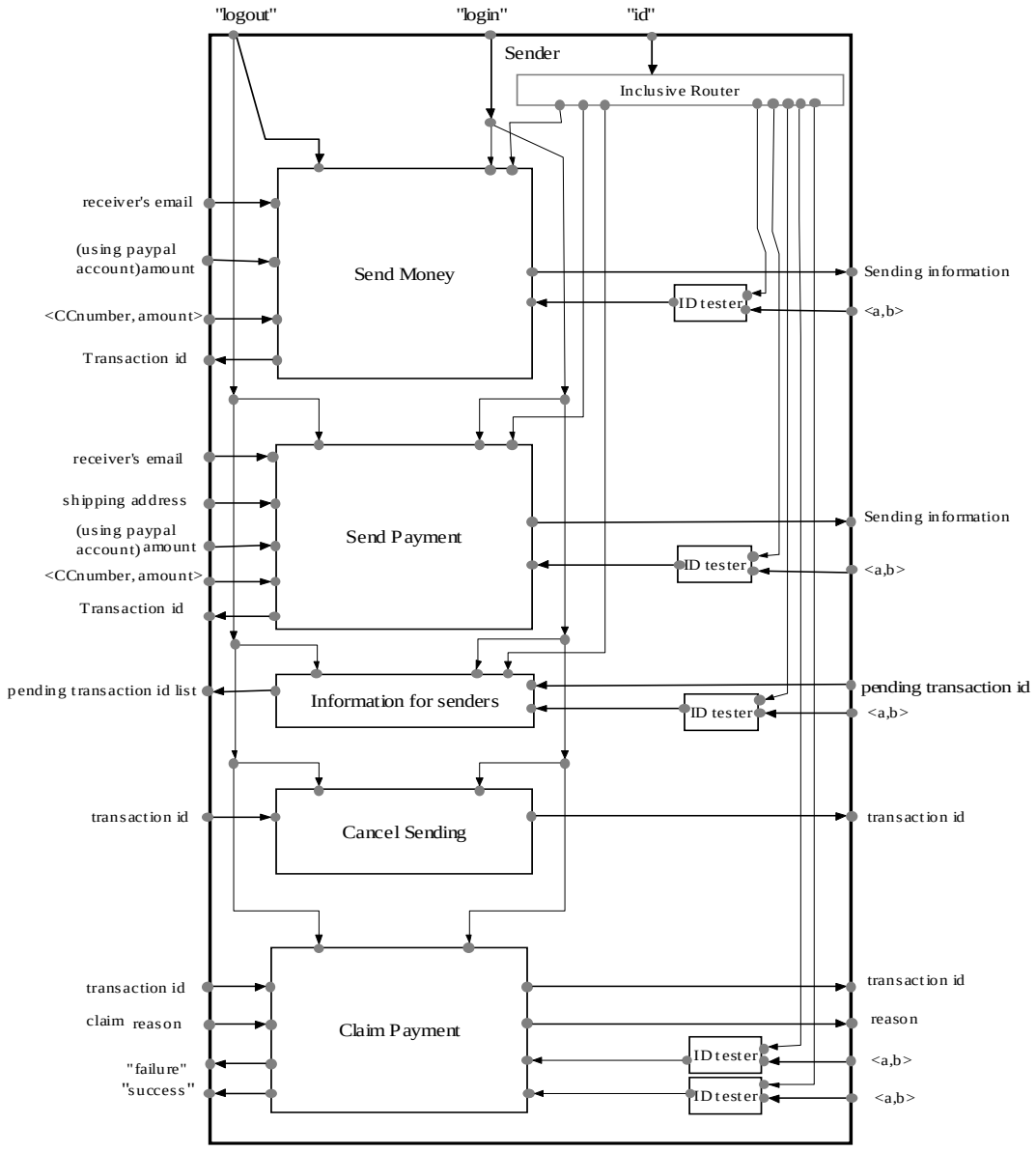

Fig. 3.8 Sender component

1) “Send money” Component

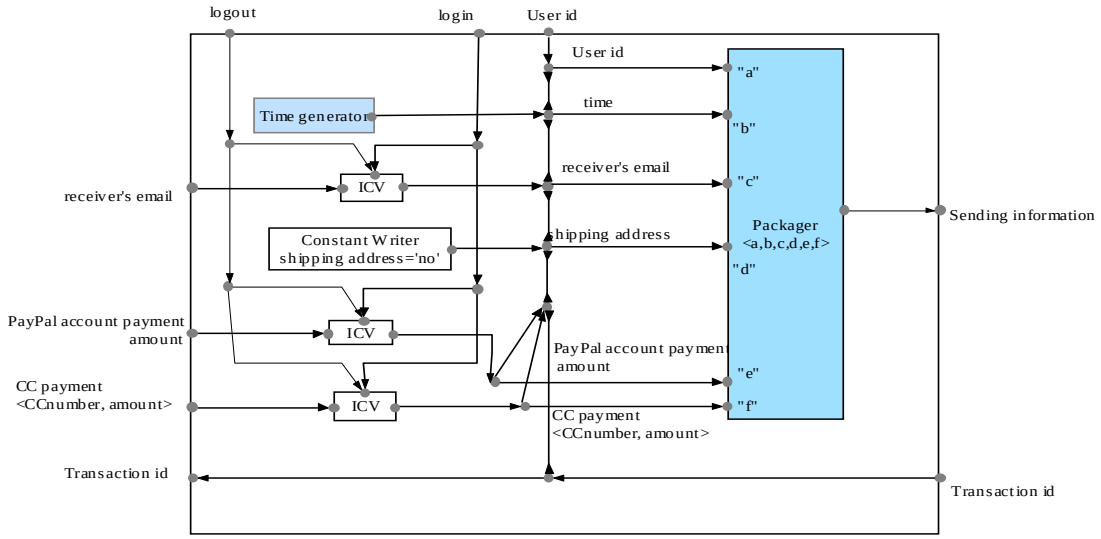

Fig. 3.9 “Send money” component

“Send money” component (Fig.3.9) allows the sender to send money to the receivers when he/she is authenticated. The “login” signal opens the left three “ICVs” to enable three input nodes, from which the component receives the input of receiver’s email address and the payment information (either PayPal account payment or Credit Card payment). The “logout” signal turns off the “ICVs” and disables the left three input nodes. The “Time generator” automatically generates a time stamp. The “Constant Writer” writes the value “no” to the shipping address, to indicate transfer of money without shipping anything.

If the current transaction is successful, the input node on the right receives the “transaction id” of this transaction in the meanwhile，and “transaction id” is passed to the left output node to the user, to allow the user to use the id in further interaction with the system.

2) “Send payment” component

“Send payment” component (Fig.3.10) allows the sender to send payment to receivers when he/she is authenticated.

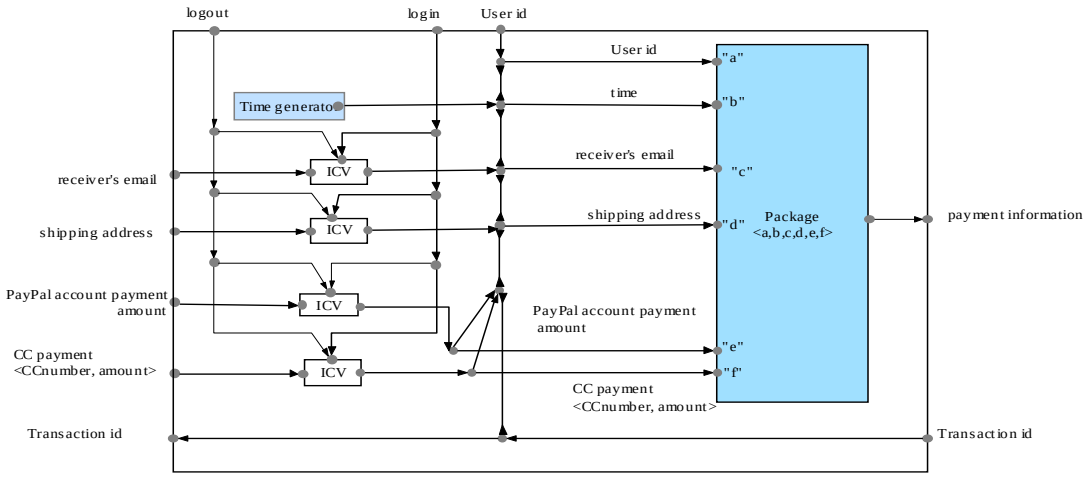

Fig. 3.10 “Send payment” component

“Send payment” component works the same as the “Send money” component, except that there is one additional input of “shipping address” instead of the “Constant Writer” in the “send payment” component, because in this case, the sender sends money to pay for a product and therefore the shipping address is required.

3) “Information for senders”

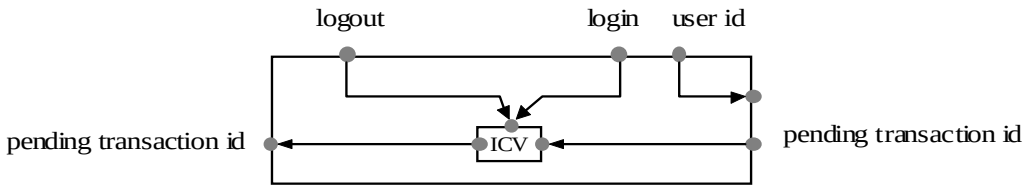

Fig. 3.11 “Information for senders” component

“Information for senders” component (Fig.3.11) allows a sender to get the transaction id list of his or her pending transactions (the money or payment sent by the sender but not yet accepted by the receiver) from the “Server” component.

4) “Cancel sending” Component

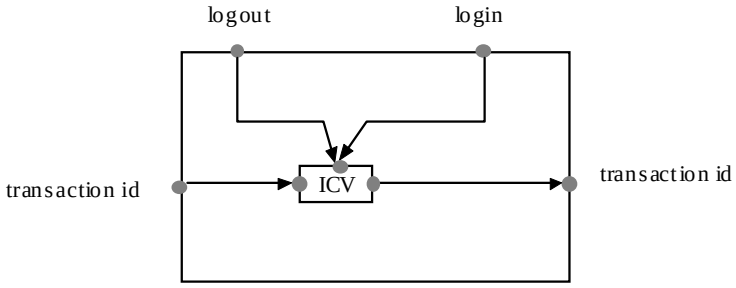

Fig. 3.12 “Cancel sending” component

“Cancel sending” component (Fig.3.12) allows the user to input the “transaction id” to cancel the pending transactions. The “transaction id” is forwarded to the “Server” component.

5) “Claim payment” Component

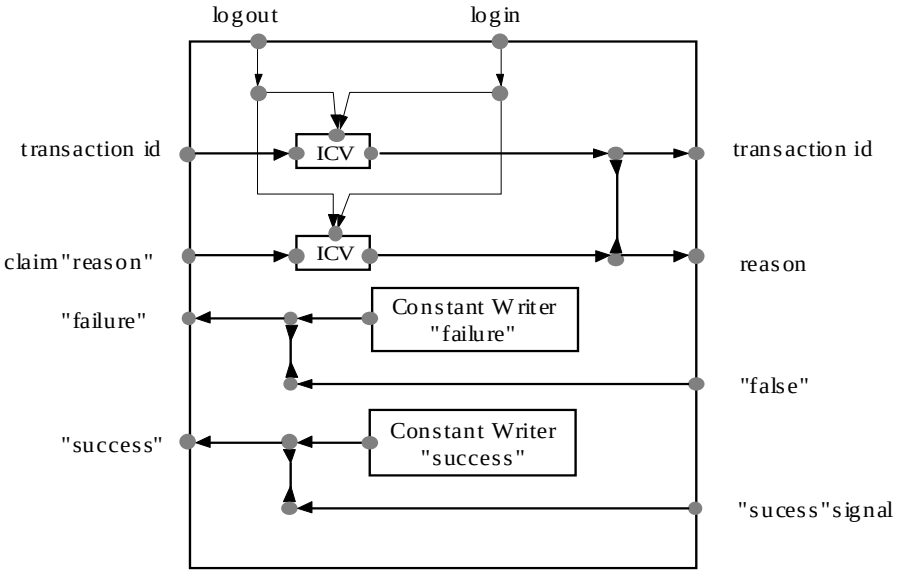

Fig. 3.13 “Claim payment” component

“Claim payment” component (Fig.3.13) allows a sender to ask for returning the payment of unsatisfied transactions. This component has the input of “transaction id” and the “reason” why the sender claims. The two output nodes on the left pass the data to the “Server” component. If the claim is legal, “Claim payment” component will receive a “success” signal from the input node on the right. Otherwise it receives

a “false” signal. When there is a “success” or “false” input, the two “Constant writer” accordingly writes the “success” or “failure” value and pass it to the left output node.

6) “ID tester” component

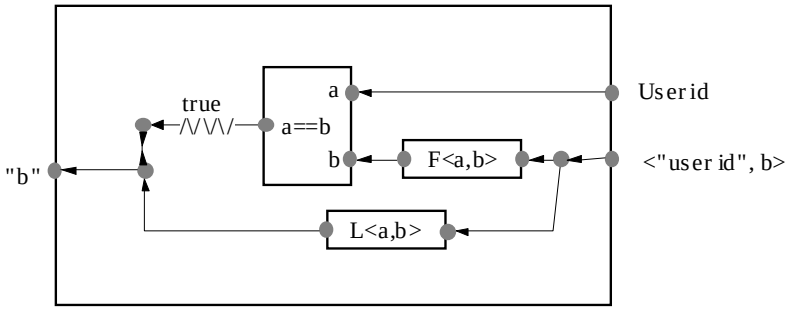

Fig. 3.14 “ID tester”Component

“ID tester” component (Fig.3.14) judges whether the information sent from the “server” is for the current “user” (either a sender or receiver). The component compares the current “user id” received from user administration component and the “user id” in the received pair of data elements from “Server” component. If they are equal, the second element, “b”, is outputted through the left output node.

3. Receiver component

Receiver component (Fig.3.15) constitutes five sub-components: “Information for receivers” component, “Response to the received money” component, “Response to the received payment” component, “Return payment” component and “ID tester” component.

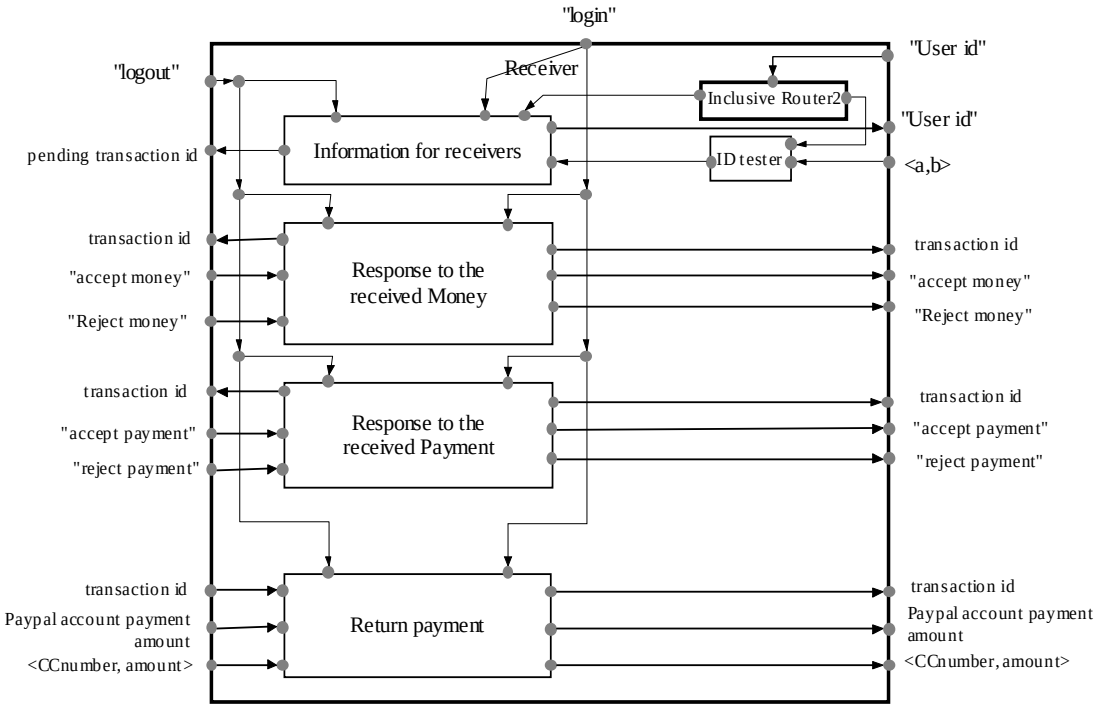

Fig. 3.15 Receiver component

1) "Information for receivers" component

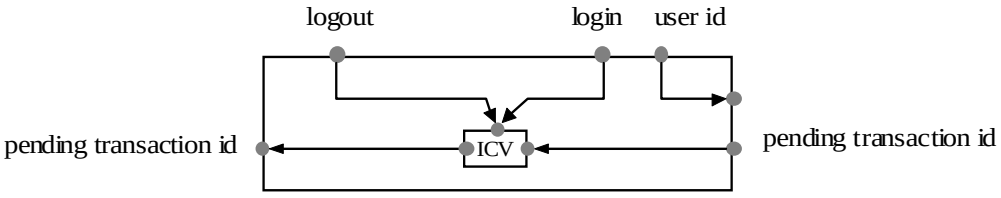

Fig. 3.16 "Information for receivers" component

This component (Fig.3.16) is a reuse of the "Information for senders" component. Here it receives the "transaction id" of the all the received money and payments that the receiver hasn't accepted yet.

2) “Response to the received money” Component

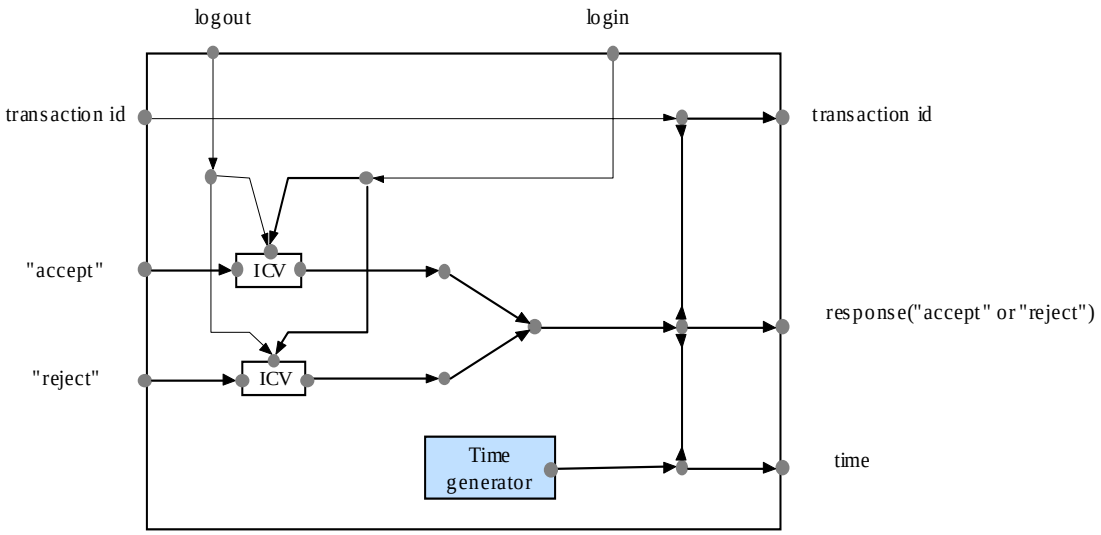

Fig. 3.17 “Response to the received money/payment” component

“Response to the received money” component (Fig.3.17) allows a receiver to accept or reject the received money from sender(s). This component receives data of “transaction id” and a response, either “accept” or “reject”, from the receiver. When there are input messages, the “time generator” automatically generates the current time. All information is passed to the output nodes on the right.

3) “Response to the received payment” component

“Response to the received payment” component allows a receiver to accept or reject the received money from sender(s). It is a reuse of the “Response to the received money component”.

4) “Return payment” Component

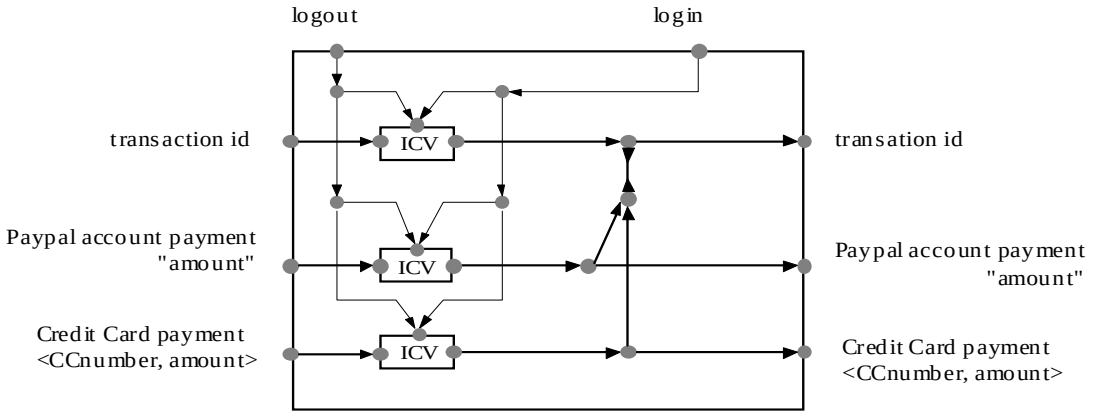

Fig. 3.18 “Return payment” component

“Return to payment” component (Fig.3.18) allows the receiver to return the payment that is owed to sender in a claimed transaction. (Because when a sender files a legal claim, the receiver may not has enough money in his/her account or the transaction amount is over $500 and the system only transfer $500, the receiver will owe the money to the sender.) The component receives the input of “transaction id” and the payment, either through PayPal account or credit card, from a receiver. This information is then outputted to the “Sever” component.

5) “ID tester” component

“ID tester” component here has the same responsibility as introduced before.

4. Server component

Server component (Fig.3.19) constitutes a “user DB” sub-component, an “E-mail generator” sub-component, a “transaction DB” sub-component and six “check validity” sub-component.

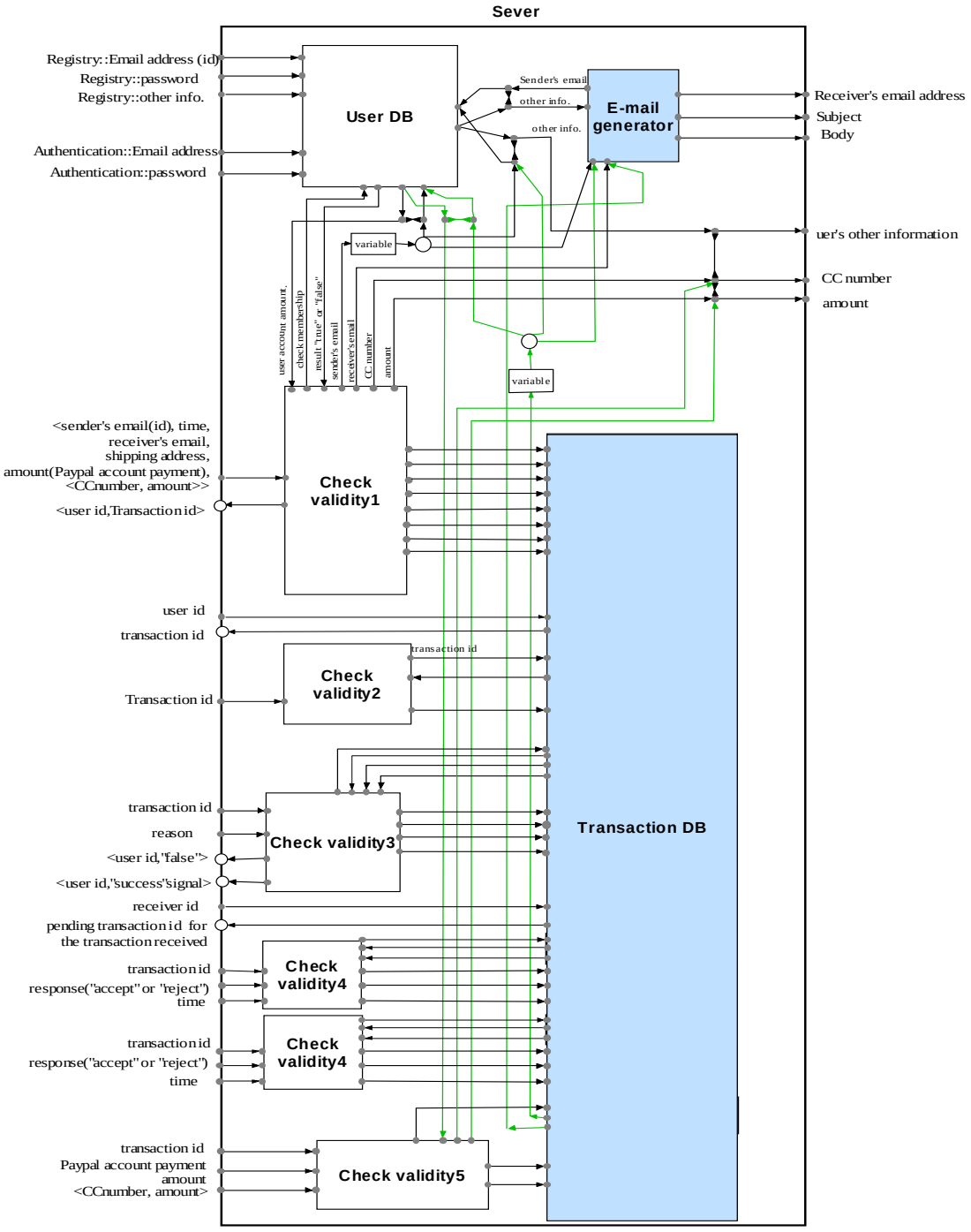

Fig. 3.19 Server component

1) “User DB” Component

“User DB” component (Fig.3.20) deals with user information checking and processing.

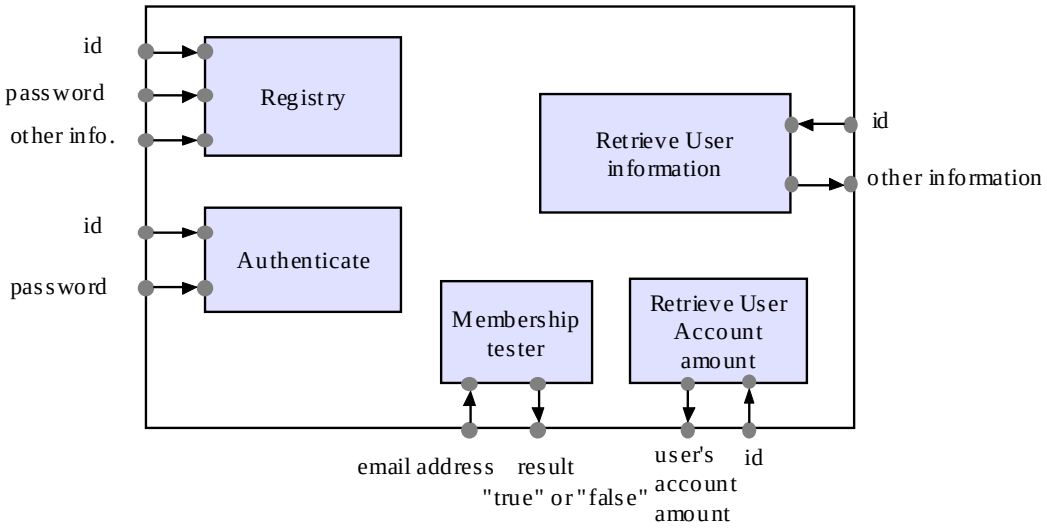

Fig. 3.20 “User DB” component

“User DB” constitutes five sub-components. One is “Registry”. It takes the data that passes from the “registration” component and stores the data successfully only when the information is valid and “id” (Email address) is unique. The second is the “Authenticate”. It takes the data passing from the “Authenticate” component and succeeds only when the “id” and “password” exist and match. The third is the “Retrieve User information” which takes the input of user id, and passes other user information to the output node. The “Retrieve user account amount” component receives the “user id” and sends out the account amount of the user. The “membership tester” component receives “email address” and searches for the member registered with this email address. If it finds such members, it sends out the value “true” and otherwise “false”.

2) “Email generator” component

When a sender sends money or payment to a receiver, the “Email generator” component (Fig.3.21) has the responsibility of generating an email to the receiver to notify him/her about this transaction.

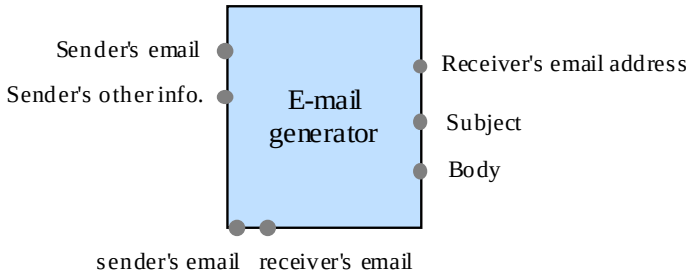

Fig. 3.21 “Email generator” component

The “Email generator” receives the data of the “Sender’ email address” and “receiver’s email address” from the two input nodes at the bottom, and outputs the “Sender’s email address” through the left output node to the “User DB” component. It receives the “Sender’s other information” from the “User DB” and outputs the “receiver’s email address”, the “subject” and the “body” of the email through its three output nodes on the right to the external Email system, which sends an email to notify the receiver about the money transaction.

3) “Check validity1” Component

“Check validity1” component (Fig.3.22) has the responsibility to check whether the money or payment sent by a sender meets the requirements of the protocol for doing payments.

“Check validity1” component has one input nodes on the left, which receives the tuple of data from the output nodes of “Send money” component or “Send payment” component. “Check validity1” component checks the receiver’s status: 1) whether the receiver is a member of the system. It sends out the “receiver’s email address” to the “check membership” input node of “User DB” component, and receives the value “true” if the receiver is a member of the system, otherwise “false”. 2) The validity of the payment method which is done through the “check validity” sub-component”: If the sender uses PayPal account to pay, the “check validity” sub-component (Fig.3.23) checks whether his/her account has enough money by comparing the “user’s account amount” and the “amount” that the user wants to send to a receiver.

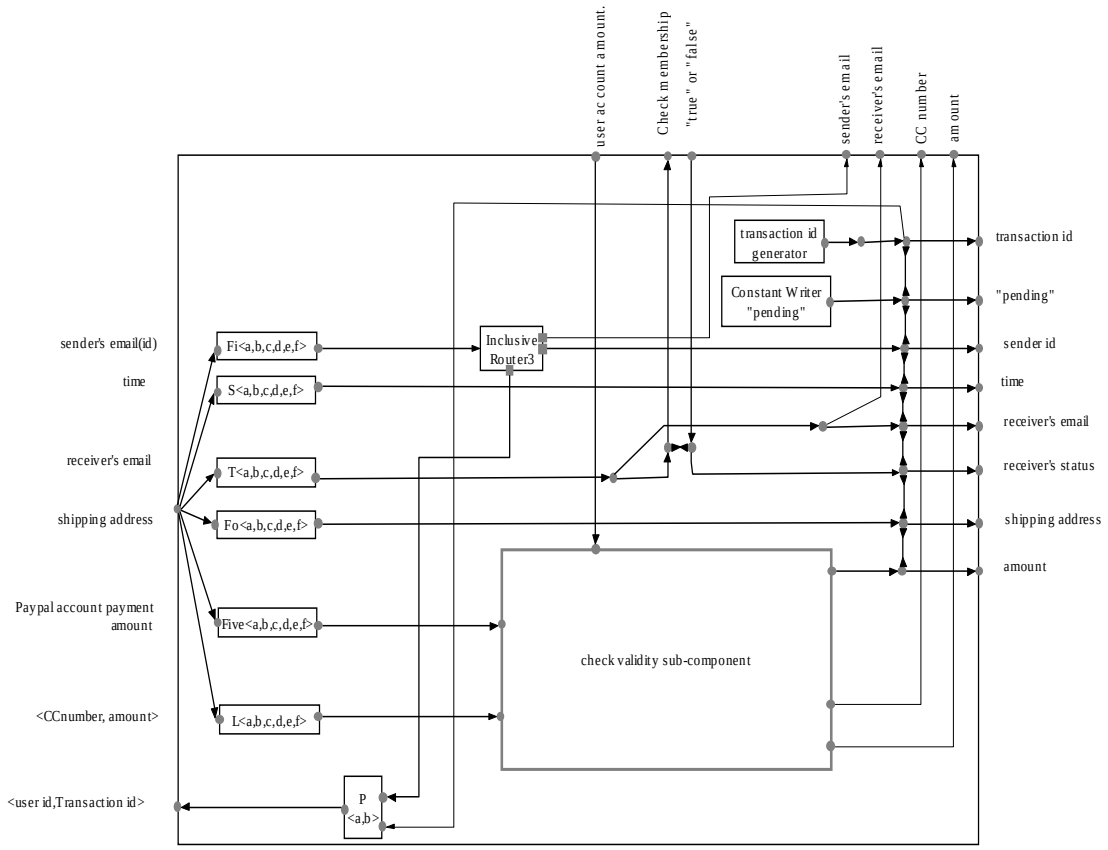

Fig. 3.22 “Check validity1” component

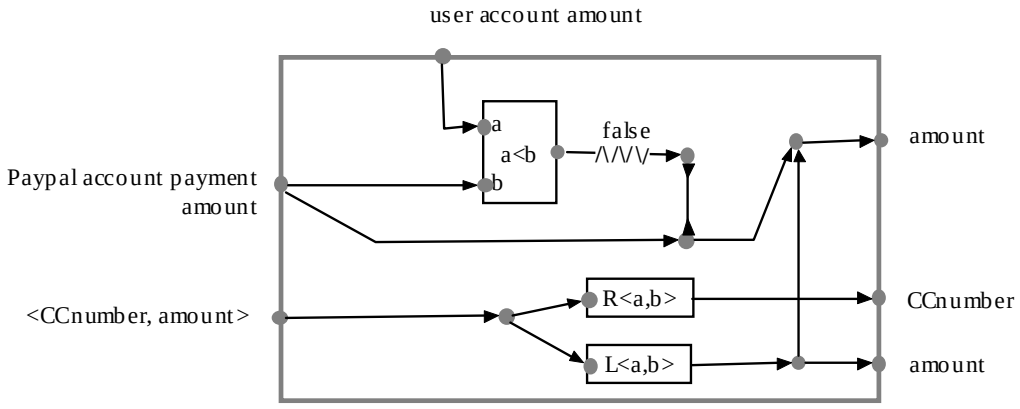

Fig. 3.23 “Check validity” sub-component

If the sender uses credit card to pay, the credit card information (CCnumber and amount) is sent out with the other two data: sender’s email address and receiver’s email address for the use of: 1) Email generating, which is introduced in the “E-mail generator” component; and 2) Credit Card validation: If a sender pays by a credit card,

the sender's email address is sent to the "User DB" component to retrieve the "sender's other information". "Sender's other information" and the credit card information (CCnumber, amount") are sent together to the external Credit Card system to be validated.

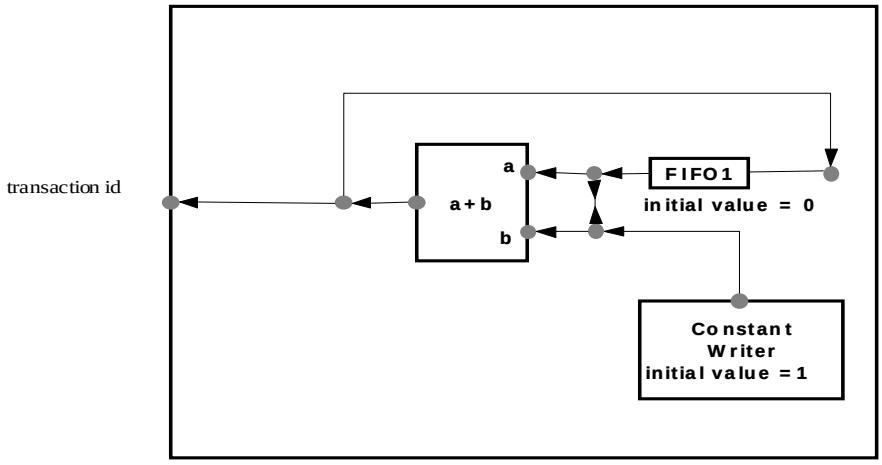

Fig. 3.24 "Transaction id generator" component

The "Transaction id generator" component (Fig.3.24) automatically generates the transaction id of the current transaction.

"Check validity1" component's eight output nodes on the right successfully pass the data to the "transaction DB" component when all synchronous channels succeed and meanwhile, the "P <a,b>" transforms the "user id" and "transaction id" of this transaction to a pair of data elements. The pair of data elements is sent to the sender component.

4). "Check validity2" component

"Check validity2" component (Fig.3.25) checks whether a sender can cancel the current transaction with the inputted "transaction id".

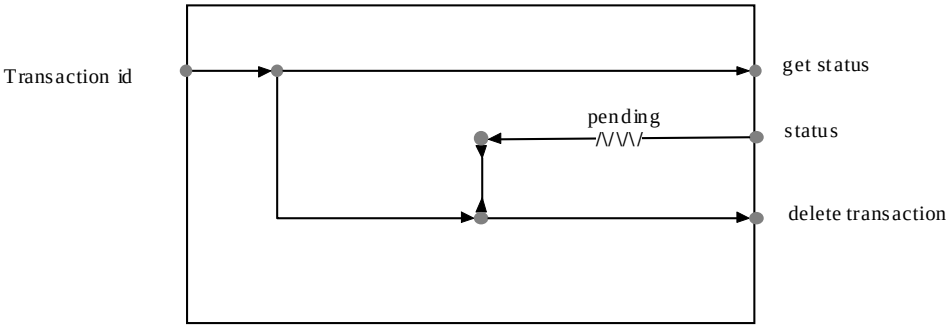

Fig. 3.25 “Check validity2” component

“Check validity2” component receives “transaction id” from “cancel sending” component and sends it out to the “transaction DB” component to retrieve the status of this transaction. A filter tests whether the status is “pending”. If it is a pending transaction, the “transaction id” is output to the “transaction DB” component.

5) “Check validity3” Component

“Check validity3” component (Fig.3.26) has the responsibility of checking the validity of a sender’s claim payment order. It constitutes two sub-components: “check transaction data” component and “justify” component.

“Check validity3” component receives data from the “Claim payment” component. “Transaction id” is sent out through the upper output node to the “transaction DB” component to retrieve the data of “receiver’s account amount”, “the sender’s email address (sender’s id) and “transaction information”. The data of “receiver’s account amount” and “the sender’s email address” is passed to the two “FIFO1s”. The data of “transaction information” is passed to the “check transaction data” sub-component to verify whether this transaction data is validated.

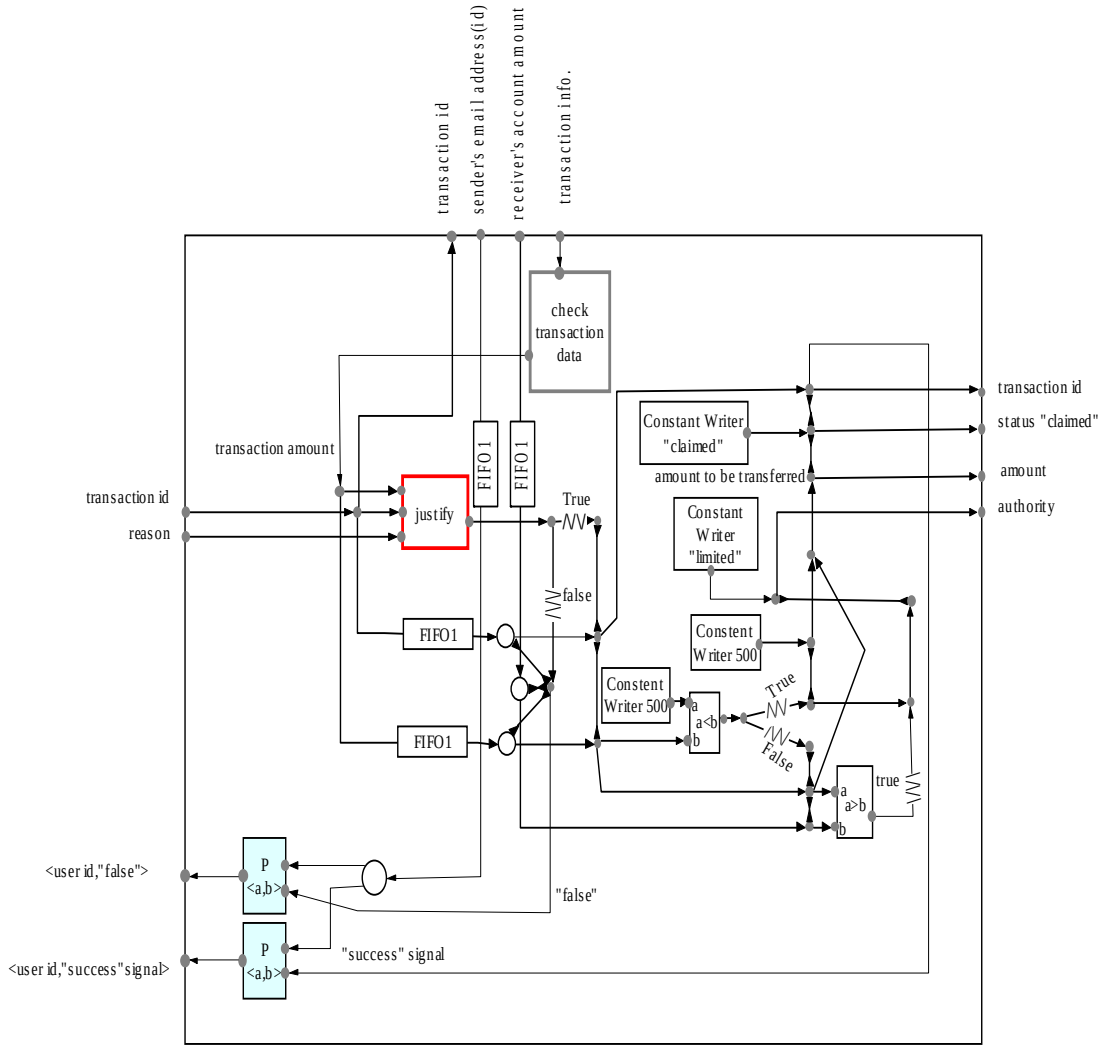

Fig. 3.26 “Check validity3” component

- “Check transaction data” component

The “check transaction data” component (Fig 3.27) receives the information from the node on the upper side. The four boxes retrieve the first, second, third and the last element from the tuple of five data elements respectively. The transaction “amount”, retrieved by the fifth box, is passed to the outside “justify” component if 1) the status of the payment is “accepted”; 2) the shipping address is “legal”; 3) the reception time is within 30 days ahead of now, which is done by the “conditioner1” sub-component (Fig.3.28); and 4) the number of existing claim time is less than 2, which is done by the “conditioner2” sub-component (Fig.3.29).

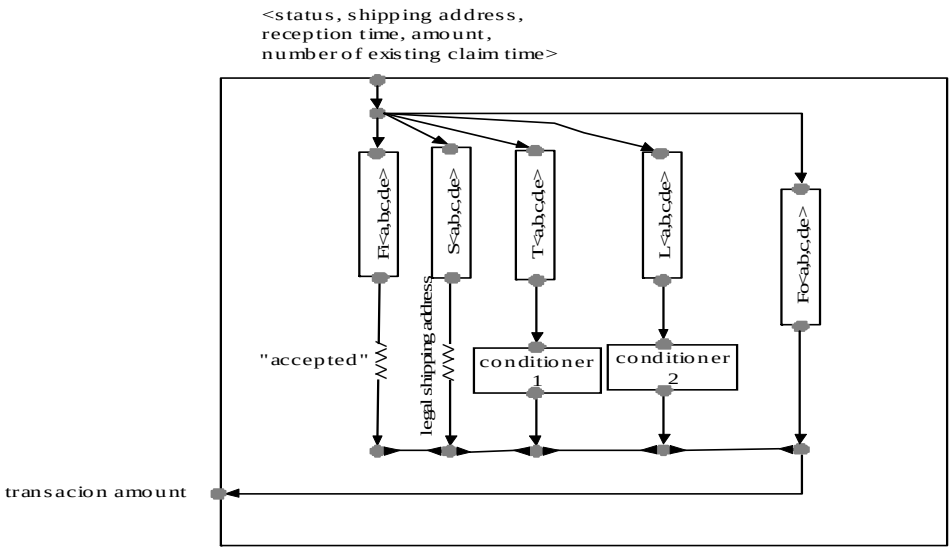

Fig. 3.27 “Check transaction data” component

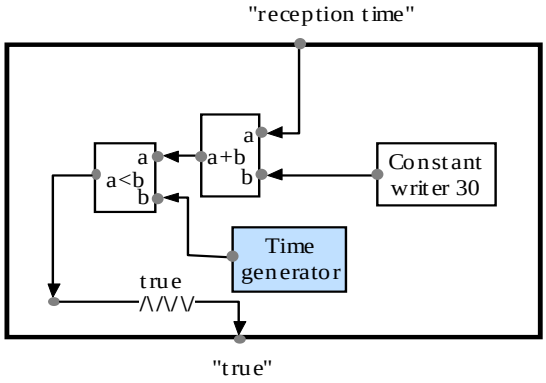

Fig. 3.28 “Conditioner1” component

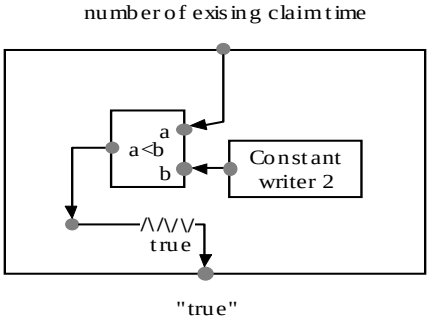

Fig. 3.29 “Conditioner2” component

- “Justify” component

Fig. 3.30 “Justify” component

The “justify” component (Fig.3.30) receives data of “transaction id”, “claim reason” and “transaction amount” and justifies whether the claim is legal. Actually, there may be human work involved. When the user files a claim, it takes some time to decide whether it is legal or not, and the user will be notified afterwards. If the “justify” component judges this claim to be illegal, the value “false” is outputted, otherwise the value “true” is outputted.

If the “justify” component outputs the value “true”, the “check validity3” component checks whether the amount of the transaction is over $500. If it is over $500, “the amount to be transferred” (from receiver’s account to the sender’s account) is $500. The receiver’s authority is “limited” (With the value “limited”, PayPal system blocks the receiver’s account until the receiver pays back the rest of money to the receiver).

If the amount of the transaction is no more than $500, the data of the “receiver account amount” of this transaction in the “FIFO1” is passed to the “a>b” component to be compared with the “amount” of this transaction. If the “amount” of this transaction is larger than the “receiver’s account amount”, the amount to be transferred is the “amount” of the transaction, and the receiver’s authority is “limited” (PayPal system blocks the receiver’s account until the receiver pays the proper amount to PayPal).Otherwise, “check validity3” does the same except that the receiver’s authority will not be limited.

The four output nodes output the information of a legal claimed transaction to the “transaction DB” component.

If the claim is successfully filed, a “success” signal is passed with the “sender’s id” together to a “P <a, b>” to be transformed to a pair of data elements, which is sent out to the “claim payment” component. Otherwise, a pair of elements of “false” value and “sender’s id” is sent out to the “claim payment” component.

6) “Check validity4” component

“Check validity4” component (Fig.3.31) checks the validity of the receiver’s response (accept or reject) to the received money or payment.

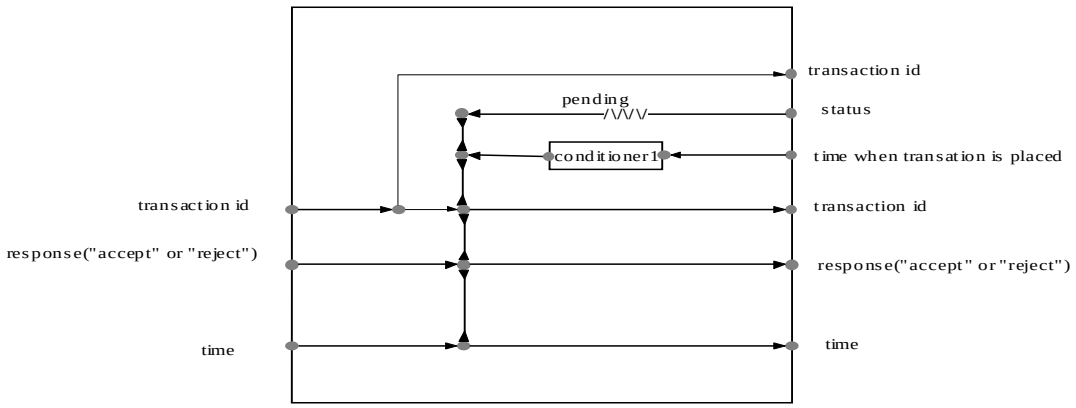

Fig. 3.31 “Check validity4” component

“Check validity4” receives data from “response to the received money/payment” component. It sends “transaction id” to the “transaction DB” component and retrieves the “status” and the “time when transaction is placed” of the current transaction. If the status of this transaction is “pending”, and the time when the transaction is placed is within 30 days ahead of the time when the activities done by the users (which is done by the “conditioner1”), the data of “transaction id”, response (“accept” or “reject”) and the time when the activities done by the users is sent out to the “transaction DB” component.

7) “Check validity5” Component

“Check validity5” component (Fig.3.32) checks validity of a receiver’s returned payment.

“Check validity5” component receives the data from the “return payment” component and checks the validity of the payment. This is similar to the validity checking when the user sends money or payment. The “transaction id” is passed out to the “transaction DB” and “User DB” component, and the data of the “sender’s id” and “user account information” of this transaction is retrieved and sent to the “check validity sub-component”, which is introduced before. If the payment is validated, the data of “transaction id” and “amount of the payment” is outputted to the “transaction DB” component.

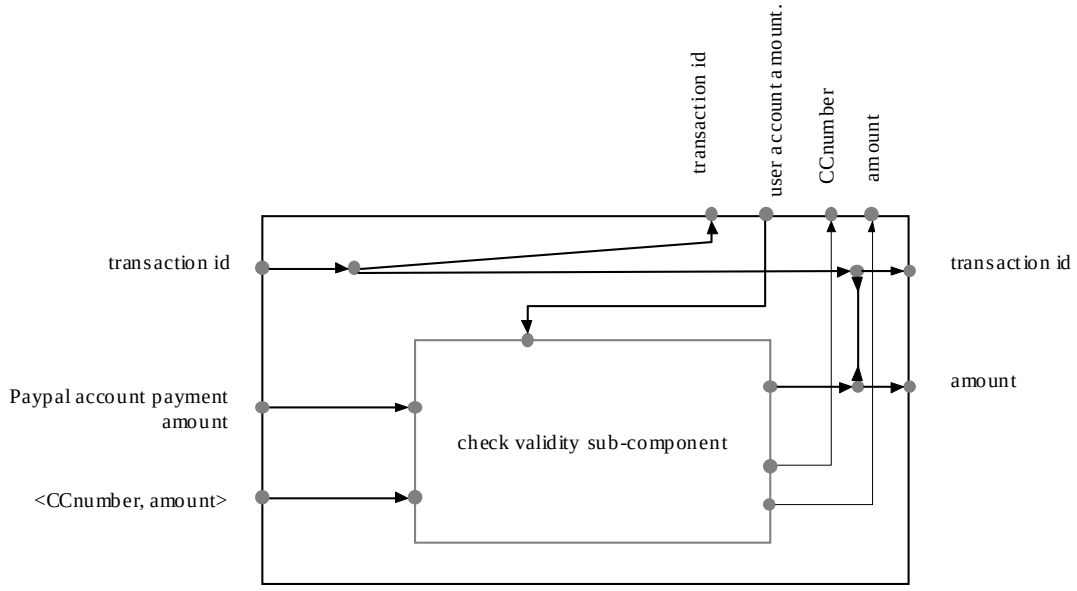

Fig. 3.32 “Check validity5” component

16. “Transaction DB” Component

“Transaction DB” component (Fig.3.33) deals with all transaction data.

“Transaction DB” component has some sub-components, “retrieve transaction data by key” component, “store transaction data by key” component, “delete transaction data by key” component and “edit transaction data by key” component. Each component has one or more component instances. We mark them each with a figure.

“Store transaction data by key” component instance receives data from “Check validity1” component and stores transaction data to the database with the key “transaction id”.

“Retrieve transaction data by key1” component instance receives the key of “sender id” (email address), and retrieves the pending “transaction id” of the all sent transactions by the sender. The pending “transaction id” is passed through “ID tester” component to the “information for senders” component.
“Retrieve transaction data by key2” component instance receives the key of “transaction id” from the “check validity2” component and retrieves the “status” of the current transaction.

“Delete transaction data by key” component instance receives data from “cancel sending” component and delete the data of pending transaction by the key of “transaction id”.

“Retrieve transaction data by key3” component instance receives key of “transaction id” and retrieves the data of the sender’s email address (sender id) of this transaction, the amount of this transaction’s receiver’s account and the other information of this transaction. This data is passed to the “check validity3” component.

“Edit data by key1” component instance receives the data from “check validity3” component and edit the database with the key “transaction id”.

“Retrieve transaction data by key4” component instance receives the key of “receiver id” and retrieves data of the pending “transaction id” of all received transactions by the receiver. The pending “transaction id” is passed through “ID tester” component to the “information for receivers” component.

“Retrieve transaction data by key5” component instance receives the key of “transaction id”, and retrieves the data of the “status” and the “time when transaction is placed”. The retrieved data is passed to the “check validity4” component.

“Edit transaction data by key2” component instance receives the data outputted from “check validity 4” component, and edit the transaction data with the key of “transaction id”.

“Retrieve transaction data by key6” component instance receives the “transaction id” from “check validity5” component and retrieves the “sender’s id” and “receiver’s id” of the transaction with the key “transaction id”.

“Edit transaction data by key3” component instance receives the data of “payback amount” from “check validity5” component, and edit the transaction data with the key of “transaction id”.

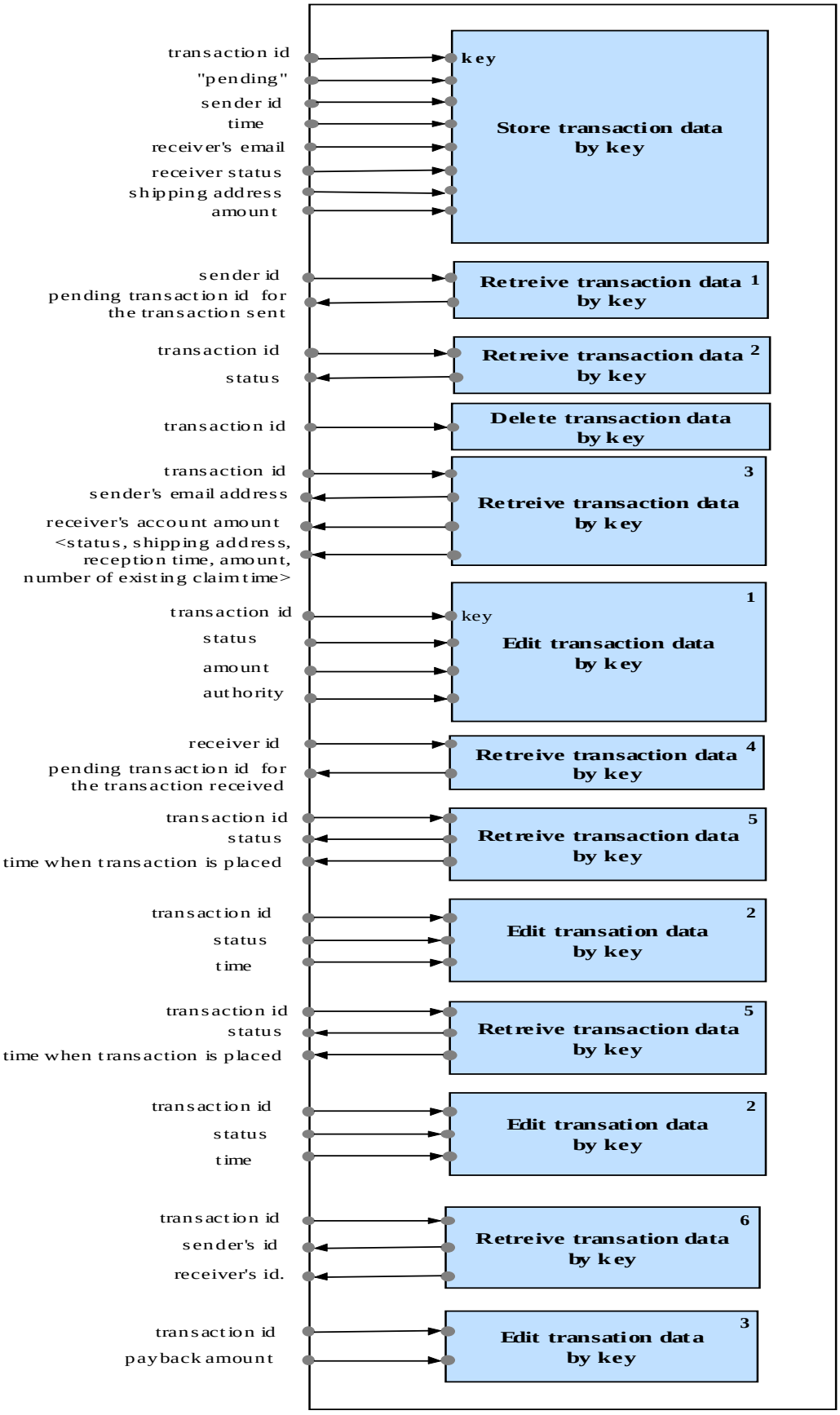

Fig. 3.33 “Transaction DB” component

3.4.4 Online payment circuit

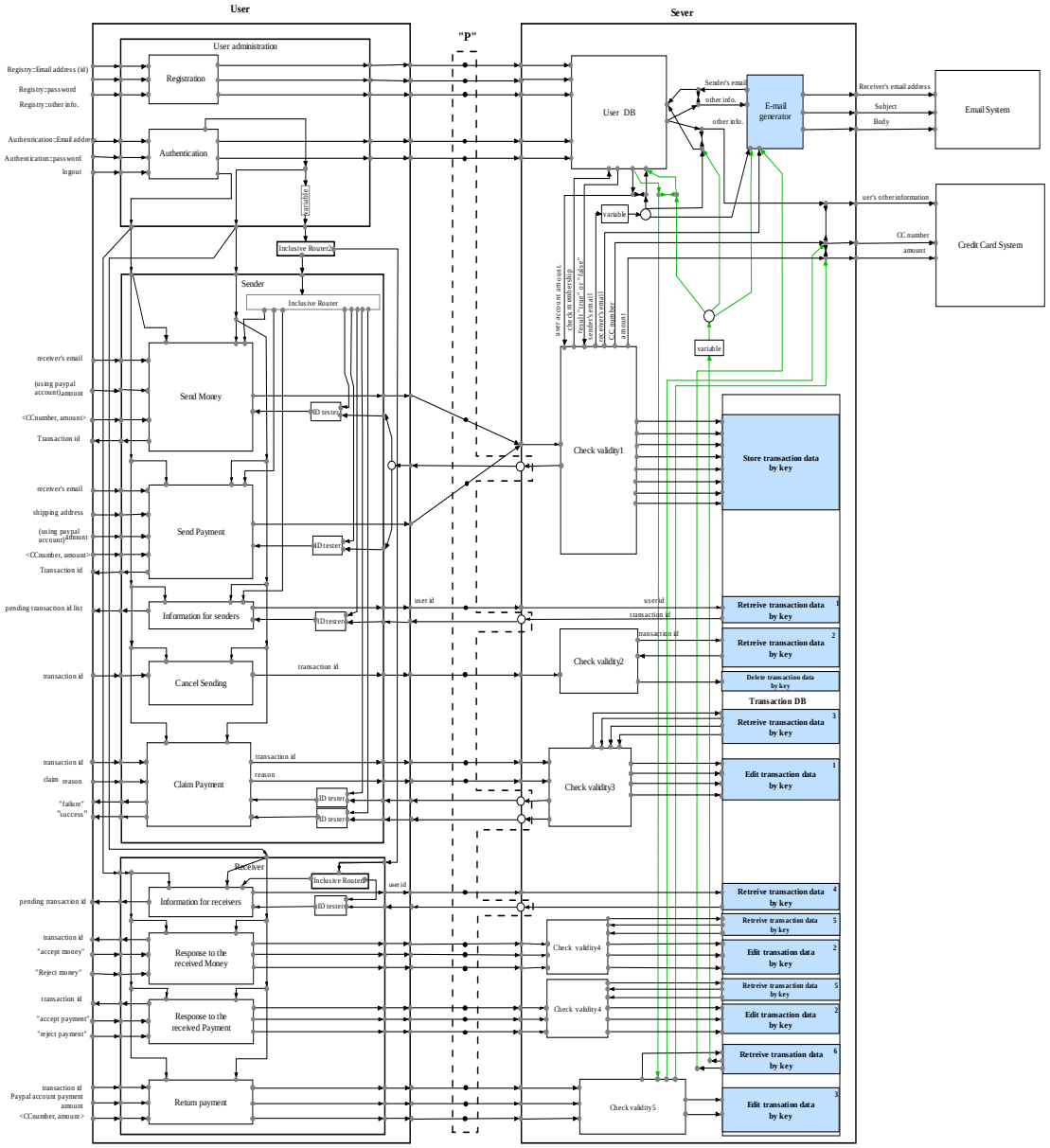

Fig. 3.34 Online payment circuit

Figure 3.28 shows the integrated Reo circuit of the online payment process specification. A user can use the input nodes of the “User” component to authenticate him/her, send money or payment, and receive the money or payment. The User Authentication, Sender and Receiver sub-components take the corresponding responsibilities to these three user’s main activities. Within each sub-component,

there are a series of sub-sub-components to perform responsibilities corresponding to the business activities identified in the use case diagram, e.g. “send money” and “send payment” components. The “Server” component contains a set of sub-components, which deal with detailed data checking and processing and make the money transfer possible.

This specification supports multiple “Users” dynamically entering the “Server”. Multiple “Users” can attach to the “Server” through the nodes and exclusive routers designated in the “P” region. Through the five “Exclusive Routers” on the “Server” interface, the “Server” can send messages to multiple “Users”.

In this online payment circuit, all the implement components are represented in white blocks. For the dark ones, such as the database components and Email generator components, we only define their functions and interfaces.

Chapter 4
Analysis of Reo as A Business Process Modeling Language

In this chapter, we analyze the effectiveness of Reo in modeling e-business process. To make such analysis, the first section gives the requirements identified in [24][25][26][27] for a business process modeling language, with respect to the problems of existing business process modeling languages summarized in chapter 1. According to these requirements, in the second section we evaluate Reo based on the case study. In the third section, we use a table to summarize our evaluation of Reo.

4.1 Requirements of A Business Process Modeling Language

To effectively model a business process, we consider four categories of requirements for a business process modeling language:

1) Language expressiveness
2) Visual notation and language semantics
3) Analysis and reasoning
4) Simulation and execution

4.1.1 Language expressiveness

Language expressiveness refers to the requirements what the language can model. Using a business process modeling language to model a business process, the business process model must be capable of providing various information elements to its users. Such elements include, for example, what activities constitute the process, who performs these activities, when and where the activities are performed, how and why they are executed, and what data elements they manipulate[24]. To provide such

information, a business process modeling language should be capable of representing one or more of the following “process perspectives” [25]:

1. The *functional perspective* represents *what* process elements (activities) are being performed.
2. The *behavioral perspective* represents *when* activities are performed (for example, sequencing) as well as aspects of *how* they are performed through feedback loops, iteration, decision-making conditions, entry and exit criteria, and so on.
3. The *organizational perspective* represents *where* and *by whom* activities are performed, the physical communication mechanisms used to transfer entities, and the physical media and locations used to store entities.
4. The *informational perspective* represents the informational entities (*data*) produced or manipulated by a process and their interrelationships.

A business process modeling language may have a set of concepts with their properties and relationships to express information in each process perspective. The integration of four process perspectives points toward the need of incorporating all business process elements into the requirements of e-business system[24]: “functional modeling to document the detail of individual tasks, behavioral modeling to identify how individual tasks interact with each other to produce the whole process, organizational modeling to examine user roles within the process, and informational modeling to document the details of information systems that support process execution”.

4.1.2 Visual notation and language semantics

Visual notation and language semantics refer to requirements how the language can represent the business process being modeled.

Visual notation provides visual shortcuts about the constructs of the language so that designers can work easily; Semantics support the expressiveness (the meaning) of the

visual notation. Only a language with formal semantics defines unambiguously what a model expresses in this language means. Therefore [26] :

1) Visual notations should be supported by formal semantics.

2) Visual notation discrimination should be easy. It should be easy to distinguish which of the notation in a model any graphical mark is part of;

3) The use of visual notations should be uniform; i.e. a visual notation should not represent one concept in one context and another one in a different context; neither should different visual notations be used for the same concept in different contexts if there is no good reason for this;

4) Visual notations composition should be made in an easy and aesthetically pleasing way.

Furthermore, representation depends on the granularity of a business process model. Granularity represents the level of decomposition of the modeler's vision. Using a business process modeling language, the granularity can be considered via the number of necessary diagrams which address different perspectives of a business process to reflect a complete vision of the business process [24]. Therefore,

5) the consistency of different diagrams should be ensured.

"Granularity can also be reflected by the levels of description of a business process diagram."[24] For instance, an abstract diagram can be progressively detailed in more operational ones according to the modeling need. In this way, a business process is specified in a hierarchical structure with each level properly abstracted. The higher level diagrams are the means to get rapid and general idea of the functions of the business process. The detailed one permits the reflection of all crucial details about how the business process is working, e.g., within the context of involved organization. In this case, 6) the integration of the business process model should be guaranteed. It should be easy to identify the relations between the low level description and the high level description, and allows tracking between abstract description and more detailed description [24].

4.1.3 Analysis and reasoning

Analysis and reasoning refer to the requirement that techniques and theories for analysis of and reasoning about models expressed in a language allow formal verification of models so that [28]:

1. The process of making a (more) formal model may reveal errors and ambiguities at an early stage.
2. Formal proofs may be available.
3. The remaining (or unprovable) rules may be translated into executable constraints in some language.

4.1.4 Simulation and execution

Simulation allows for testing of a business process model by executing it in a simulator (toy environment). Execution allows for using (a part of) a model to automate (a parts of) a business process in the computing environment of an organization. Therefore, to achieve business process simulation and execution, a language should be supported by:

1. Simulation tool(s);
2. Automation tool(s).

4.2 Evaluating Reo

In chapter 3, we model an online payment process as our case study. Using the requirements as criteria, we evaluate how Reo language performs in this case study.

4.2.1 Language expressiveness

Reo is a connector-oriented language. Component and connector are the basic concepts that we use to model the PayPal's online payment process, where connectors represent encapsulated Reo circuits:

- The high level components are used to represent the user roles in the business protocol.
- The low level components are mapped from the business activities (identified in the use case diagram) performed by user roles.
- The connectors are used to represent the relations of the user roles and business activities through their coordination among components on different levels of abstraction.
- The data carried by a connector reflects the information flow throughout the business process.

We observe that:

1. The functional perspective is mainly reflected by the low level components. The name of a component hints what the activity is. For example, the component named “send money” responds to the activity that a sender sends money to a receiver for non-purchase purpose; the component named “send payment” responds to another activity that a sender sends money to a receiver to buy some product(s); and a series of “check validity” components are corresponding to the checking activities in the online payment process. However, we think the functional perspective supported by Reo is at a low level. Reo lacks high level support to directly capture the functionality of the business process, for which we use the use case diagram of UML.

2. The behavioral perspective is well supported through the Reo connectors among components. This support lies in the rich expressiveness of Reo connectors. Through primitive channels and their dynamical composition, various kinds of the relationships among business activities can be modeled. For example, in our modeling, the “authentication” component’s output “login” is transferred by connectors to one of the necessary inputs of some other components, such as “send money” and “send payment” component etc.. This way, the working of “authentication” component is the prerequisite of other components: only when

authenticate activity is performed can other activities (e.g. send money, send payment) be done. One can refer to [34] to find complex connectors, such as “Sequencer”, “Or selector”, to model more complex rules in a business process.

3. As we focus on the automated part of the online payment process, the process is executed by e-business system., In this case, we do not bring in organizational issues in our modeling, such as different department participants, physical places where business activities happen etc.. Reo can not represent organizational perspective. We use the component naming convention to provide connectors for the different stake holders in the PayPal process.

4. The informational perspective is poorly supported by the initial set of primitives in Reo. To support more elaborate data processing we needed to introduce basic components, e.g., Sum(+) and Less(<). In the online payment circuit, every component has its inputs and outputs, so that one can observe the data entities consumed and produced in the process. The connectors transfer the data from outputs of one or more component into the inputs of some other component(s). The relationships of data flows are clearly represented.

Using components and connectors, a Reo circuit represents all information elements of the business process being modeled in one diagram. The integration of functional, behavioral and informational perspectives as well as the user roles of the process enhances the transformation from business process modeling to the e-business system implementation.

4.2.2 Visual notation and language semantics

With respect to the visual notation and language semantics of the business process model, we make the following observations:

- Reo offers visual notation for components, connector encapsulation, channels, and nodes, connecting the nodes. Channels and nodes have clear formal semantics, which reduces the ambiguity of the process model. The visual notations for component and connector encapsulation that Reo provides facilitate top-down decomposition and refinement.

- Notation discrimination is clear. Using connector encapsulation properly, a designer can reduce the size of the circuit specification.

- Notation composition in Reo is easy and clear. Rules for channel composition to all channels and component instances in the circuit are based on Reo's formal algebraic semantics.

- Using Reo's connector encapsulation, the granularity of the process model is achieved by the levels of abstraction in a top-down design manner. The high level structure specifies user roles and their relations identified in the business protocol through high level components and connectors, such as the user (send and receiver) and server. The low level description specifies detailed activities performed by each user role and their relations through low level components and connectors. From high level structure to the low level specification, information is progressively conveyed in detail.

- The consistency between different levels of the specification is ensured through the directly mapping from high level components to their sub-components on the lower level. This also attributes to Reo's formal semantics and component encapsulation, which reduce ambiguity in defining components and their decomposition. This way, the relations among different levels of components are identifiable. The relationships among different levels of specification are clear and traceable.

This kind of granularity enabled by Reo leads to the following observations:

1. The functional, behavioral and informational process perspectives as well as user roles are integrated in one diagram. This allows a business process modeler to have an integrated view of the business process, which overcomes the traceability and inconsistency problems among separate diagrams, brought by some software modeling languages when used to model business process, such as UML. As such Reo favors integrated views than separate ones.

2. From high level modeling to the low level specification with more crucial operational details conveyed, we can follow a top-down approach to implement the business protocol using Reo. This is further facilitated by the exogenous coordination of Reo connectors, which makes the single component implementation independent from each other.

4.2.3 Analysis and reasoning

1 Reo has formal semantics for the business process modeling. This allows for formal verification of Reo designs.

2 In the sense of reliability of the business process model, using the synchronous primitives of the Reo language, designers can enforce an explicit business scenario without the need to design additional fortification code. This way, any intermediate and incomplete results are avoided [29].

4.2.4 Simulation and execution

1. To achieve business process model simulation, Reo is supported by a simulator tool that can implement Reo operational semantics to run and test protocol prototypes locally. [22] The simulator is currently in its final stage of development.

2. To achieve business process model execution Reo is supported by:
    - A distributed coordination middleware (under development), which can implement full Reo operational semantics to run Reo circuit in the distributed environment.
    - A model checker tool (under development) can check the properties of the designed system.

4.3 Summary

Table 4.1 shows the summary of our evaluation of Reo. We use zero to 3 stars to score how Reo meets the requirements in each category. No stars means Reo does not meet the requirement, 3 stars mean Reo completely meets the requirement.

Category	Language Expressiveness			
Requirements	Functional perspective	Behavioral perspective	Organizational perspective	Informational perspective
Score	★★	★★★		★

Category	Visual notation and Language semantics					
Requirements	Formal semantics	Notation discrimination	Uniform use of Notation	Notation composition	Granularity	
					Diagrams consistency	Views integration
Score	★★★	★★★	★★★	★★★	★★★	★★

Category	Analysis and Reasoning	Simulation and Execution	
Requirements	Techniques and Theories allow formal verification of model	Simulation tool(s)	Execution tool(s)
Score	★★★	★★	★

Table 4.1 Summary of evalution

Chapter 5
Conclusions

5.1 Summary of findings

In this thesis, we assess the fitness of the Reo coordiantion language as a business process modeling language. We presented a specification of PayPal's online payment business process model using Reo as a case study. We use the requirements for a business process modeling language identified in [24][25][26][27] as evaluation criteria in the context of the case study. According to these criteria, we conclude that:

1. Reo is good at representing the behavior perspective of a business process. However, it lacks language expressiveness to represent the other three process perspectives: The functional perspective is supported at a low level. Reo lacks high level support to directly capture the functionality of the business process; The organizational perspective is not supported by Reo; The informational perspective is poorly supported by the initial set of primitives in Reo which only work with data elements with simple structure;
2. Reo's visual notation, backed by a correspondence to its formal semantics, recduces ambiguity in modeing business processes. Reo's strict compositionality, also presenting its visual notation, helps to provide an integrated view for large business process models. This overcomes exsiting problems of views seperation and inconsistency among separate diagrams in e-business process modeling;
3. Reo offers techiniques and theories to make formal verification of business process models;
4. Reo has the ability to support business process model simulation and execution. Simulation and execution tools are under development.

5.2 Expectations for future work

Based on our modeling experience and evaluation of Reo, in its future development, we expect Reo to have an extension that can provide high level support of different process perspectives, which allows representing functional, behavioral, organizational and informational elements of a business process in a more intuitive way. More vocabularies are needed to represent such information, especially the organizational and informational elements, and to facilitate the levels of abstraction in a business process model. In addition, we expect an efficient modeling tool to be developed that can facilitate generic drawing, diagram editing and to guide the modelers through the design and implementation of automation systems for business processes. Furthermore, we expect Reo's simulation and execution tools to be put into practice, so that an e-business process model can be tested using simulation, and automated using a middleware that can directly run Reo models in a distributed environment.

5.3 Personal experience

In this thesis work, one difficultiy was making the requirement analysis, which includes the business protocol definition and use case diagram making. To presicely grasp the rules of the business process, I iterated over the original description of the business protocol to achieve an accurate and useful description (for modeling) in plain English. I then spent a lot of time to learn how to make use case diagram to capture the functionalities of the business protocol for the guidance of the design work. This experience also taught me the importance of a clear requirement analysis in a practical project.

Another difficulty was the design of a high level structure of the process model. On the one hand, the high level structure should be cosistent with the user requirements. On the other hand, it should facilitate the low level design and implementation of the

process model. To meet these needs, the high level structure was adjusted to the current “User-Server” design: the “user” component fulfills the high level user requirements and the “server” component reacts to the “user” component to perform operations of data checking and processing to effectively support and implement all individual user scenarios. This taught me a lesson on how to design a proper specification model.

Finally, the process of writing this thesis taught me a lot of the techinical writing, in which every word should be accurate in its meaning and picked carefully to enhance the understanding of the audience.

References

[1]. Rosemarie Graham, Development Meets Business Process Modeling. March, 2005

[2]. John Moore This year's model: Business process, April 19, 2004

[3]. Hafedh Mili, Guitta Bou Jaoude, Éric Lefebvre, Guy Tremblay, Alex Petrenko Laboratoire de Recherche sur les Technologies du Commerce Électronique, Business Process Modeling Languages: Sorting Through the Alphabet Soup

[4]. B. Hnatkowska, Z. Huzar, J. Magott, "Consistency Checking in UML Models," www.fit.vutbr.cz/events/ism/2001/pdf/hnatkowska.pdf

[5]. Dave Clarke, David Costa, Farhad Arbab, Connector Colouring I:Synchronization and Context Depending, FOCLaSA, August 5th.

[6]. Arbab, F., Abstract Behavior Types: A Foundation Model for Components and Their Composition, In: Proceedings of the First International Symposium on Formal Methods for Components and Objects (FMCO 2002), LNCS 2852, 2003, pp.33-70.

[7]. Arbab, F., Rutten, J.J.M.M, A Coinductive Calculus of Component Connectors, In: Proceedings of 16th International Workshop on Algebraic Development Techniques (WADT 2002), Lecture Notes in Computer Science 2755, Springer, 2003, pp. 35-56.

[8]. Rutten, J.J.M.M., Kwiatkowska, M., Gethin, N., Parker., D., Chapter 5: Component Connectors, In Mathematical Techniques for analysing concurrent and probabilistic systems, CRM Monograph series Volume 23, American Mathematical Society, 2004, p. 215.

[9]. Arbab, F., Baier, C., Rutten, J., Sirjani, M., Modeling Component Connectors in Reo by Constraint Automata, Electronic Notes in Theoretical Computer Science, vol. 97, No. 22, July, 2004, pp. 25-46.

[10]. Curtis, B., Kellner, M.I. and Over, J.: “Process modeling”, Communications ACM, 35, (9), pp. 75- 90, 1992

[11]. Glossary of IT Investment Terms, http://www.gao.gov/policy/itguide/glossary.htm

[12]. M.A. Ould, Business Processes: Modelling and Analysis for Re-engineering and Improvement, 1995, John Wiley & Sons.

[13]. Len Bass, Paul Clements, Rick Kazman, Software Architecture in Practice, Second Edition

[14]. Philippe Kruchten, Architectural Blueprints—The "4+1" View Model of Software Architecture

[15]. Valhie Issarny, Titos Saridakis, Apostolos Zarras, Multi-View Description of Software Architectures

[16]. Meyer Tanuan, Software Architecture in the Business Software Domain: The Descartes Experience

[17]. Carmichael I, Txerpos, V., and Holt, R.C., Design Maintenance: Unexpected Architectural Interactions, Proc. International Conference on Software Maintenance, ICSM 95.

[18]. Murphy, G.C, Notkin, D., and Sullivan, K., Software Reflexion Models: Bridging the Gap between Source and High-Level Models, Proceedings of the Third ACM Symposium on the Foundations of Software Engineering (FSE 95)

[19]. Alexander Ran, Architectural Structures and Views

[20]. Business Process Modeling Language, Business Process Management Institute, January 24, 2003, 96 pages

[21]. Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Aivana Trickovic, Sanjiva Weerawarana, Business Process Execution Language for Web Services (BPEL4WS) version 1.1 , 5 May 2003

[22]. Zlatko Zlatev, Nikolay Diakov, Stanislav Pokraev, Construction of Negotiation Protocols for E-Commerce Applications

[23]. Farhad Arbab and Farhad Mavaddat, Coordination through Channel Composition, page 3

[24]. Georgem.Giaglis, A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques

[25]. Curtis, W., Kellner, M. I., and Over, J., "Process Modeling," *Communications of the ACM*, Vol. 35, No. 9, pp. 75–90 (1992).

[26]. John Krogstie, Evaluating UML: A Practical Application of a Framework for the Understanding of Quality in Requirements Specifications and Conceptual Modeling

[27]. Feriel Daoudi, Selmin Nurcan, A framework to evaluate methods' capacity to design flexible business processes

[28]. Wieringa, R. (1998), A Survey of Structured and Object-Oriented Software Specification Methods and Techniques. ACM Computing Surveys 30 (4) December, 459-527.

[29]. Nikolay Diakov and Farhad Arbab, Adaptation of Software Entities for Synchronous Exogenous Coordination: An Initial Approach

[30]. Scholz-Reiter, B. and Stickel, E., (eds.), Business Process Modelling. Springer-Verlag, Berlin (1996).

[31]. Davenport, T. H. and Short, J. E., "The New Industrial Engineering: Information Technology and Business Process Redesign," Sloan Management Review, Vol. 31, No. 4 (Summer 1990), pp. 11–27.

[32]. Farhad Arbab, Reo: A Channel-based Coordination Model for Component Composition, Mathematical Structures in Computer Science, Cambridge University Press, Vol.14, No.3, pp. 329-366.

[33]. http://www.sungoldsoap.com/paypal.html

[34]. Dave Clarke and David Costa, A Compendium of Reo Circuits, work in progress, CWI, 2005

[35]. N.K. Diakov, Z.V. Zlatev, S.V. Pokraev, Composition of negotiation protocols for E-commerce applications, January,2005

[36]. Dave Burchell, David Nielsen, Shannon Sofield, PayPal Hacks, Publisher: O'Reilly, September, 2004

Appendix

List of Figures

List of Tables