

# Communication Model to Support Rapid Application Development

Yingbo Wang, Peter J. Clarke and Yi Deng  
School of Computing and Information Sciences  
Florida International University  
Miami, FL 33199, USA  
{ywang002, clarkep, deng} @cis.fiu.edu

## Abstract

*Communication systems, along with database systems, have attracted a lot of attention in software engineering research. Encouraged by the success of SQL systems, some early efforts have been devoted to the study of a similar system for communication systems. Even though there has been some research to provide a common language or a virtual machine for building communication systems, there are only few theoretical studies on the complexity of specifying and modeling communication systems.*

*In this paper, we propose an approach to model common communication systems which have no complicated stochastic control logics. Our model contains two parts: (1) a communication logic definition and (2) a set of communication system operations. Our approach has the following benefits: a unified interface between communication roles/systems, a recursive way to model communication systems, and the adoption of a multicast-conference model. These benefits enable us to efficiently build scalable communication systems. Also, it enables us to provide a Graphic User Interface for users to build a customized communication system. We give a case study showing that our approach is effective and efficient in modeling common communication systems.*

## 1. Introduction

Using the state-of-the-art networking hardware and software systems, it is possible to communicate with many kinds of different entities (e.g. persons, computer application system, etc.). Building a communication system, however, is not an easy task. The way to build communication systems is still application-dependant. A hospital, for instance, may have a communication system to communicate patient information between healthcare providers [1, 6]; a fire station may have a communication system to coordinate the communication during an emergency [4]; and messenger-like systems can transmit daily messages. None of these applications provide a way that allows a user to dynamically specify new communication models applicable to that domain.

We still need to spend several months to build a domain specific system even though they have many similar characteristics in common, particularly with respect to the structure of the communication. Our expectation is to build an abstract model that can be used to describe most communication systems. Based on this model, we show how a Communication Modeling Language [2] can easily specify several instances of communication. The approach to building and analyzing communication models will be presented in this paper. Previous work that supports the feasible implementation of interpreting and executing the communication specified by the communication modeling language is presented as well.

A large number of applications can be classified as communication applications or have a major component that deals with communication. Examples of these communication systems include: messaging system, video conference, teleconferencing, telemedicine, distance learning, and disaster management. The characteristic of all communication applications is that they contain functionality for exchanging information or opinions, which is the meaning of "communication". In building a powerful and easy-to-use communication modeling language, we have to model the communication systems in a direct and straight forward way.

Based on the nature of a communication activity, we define the list of basic components to describe a communication. The basic components in a communication application are: *participants*, *channel for communication* and the *transmitted data*. In another words, a communication system specifies that who, when, where, and what are exchanged during the data exchanges. Traditional communication models do not easily facilitate the modeling of today's communication systems. One of the general communication models [7] is the model of one time communication activity. It is a model composed of one sender, one receiver, transferring messages and the channel. Such a system focuses on the resolution of connecting, sending, receiving and disconnecting. The limitation of using the traditional approach to model communication systems is that it does not handle more than two participants and it does not provide a way to handle a complex connection architecture (e.g., sending

multiple files of different formats). For example, if the communication application is a video conference of more than two participants, the existing communication model can not handle it. If a communication requirement is to pass a piece of data to each member in a list of participants, and each one who receives it should update it before passing it to the next participant, we need to construct and connect several instances of this model together with a description of the connection status for each participant in such an application. In fact, for even the most domain specific communication systems, we still need to spend several months to build such applications.

The main contribution of this paper is the development of a methodology to model general communication activities and the application dependent system structure more conveniently. We give an approach in this paper to model communication logic. Our methodology satisfies following features:

1. Simple communication systems can be modeled in this system easily.
2. Complicated communication systems can be defined using a recursive approach.
3. Provides adequate interfaces for communication components that allow new communication systems to be created from existing communication components.

We use an approach that is similar to techniques used in other domains (for example, the database domain) to divide the model into system structure definition part (Communication Logic Definition) and system operation part (Communication System Operation). *Communication Logic Definition* models the components of who, where and how to communicate. It is a global picture of communication structure. *Communication System Operation* handles what is communicated. It focuses on users' activities during a communication, such as connecting, sending data and disconnecting.

By the structure of separating whole system into two parts, the difficulty of modeling communication is decomposed. The system operation part handles the detail of building real connection and transferring data while communication logic definition part handles the entire structure of the connection state among all participants.

We have completed some previous work which focuses on designing and implementing the Communication System Operation part [2]. This paper however will focus more on the Communication Logic Definition. Our method for modeling Communication Logic has the following benefits:

*Uniformed interface definition.* Connection between interfaces is the only way to exchange data among nodes. The constraint of interface definition makes the connection among different nodes uniformed. We can build data exchange channels between different persons and communication models in the same method of defining input and output interface pairs.

*Recursive definition of system:* A participant in a communication model can be a single person, or it can be another communication logic model. This means that all previously defined communication logic models can be used as participants when constructing more complex models. For more complicated communication logic models, we first decompose them into several simple communication sub-models and build each sub-model respectively. After building up subsystem models, we can construct the whole logic model using these subsystem models that have been built. On the other hand, if we have the task of specifying the cooperation of two existed communication systems, we can simply use the interface of each communication model to make the new communication logic of cooperation. Note that the remaining logic defined in the two systems is unchanged.

*Constraints definition:* a constraint defines the rule of status transition in a communication logic model. By defining a constraint on a model, a user can specify a more powerful rule-based communication model which may respond to events at run time.

Last but not the least, the third advantage of our approach is the ability of easily model common communication applications. Using a model for general conference communication and our recursive definition, we can cover the requirements of most communication systems. That is, we are able to save time and effort when building complex communication logic models. In addition, since multicast conference models and recursive constructs don't need many inputs, we can even build a Graphic User Interface (GUI) for users to build customized communication systems, thereby avoiding the use of a programming language. This is quite important in software engineering because once we can provide a GUI for users to build communication models, the users' input logic can be confined and therefore the models they build can be easily validated. Evidence of this augment is the dominating success of SQL systems over data file systems. The paper is organized as follows: Section 2 provides the formal definition of communication logic model. Section 3 presents some application cases and we conclude in Section 4.

## 2. Model Description

In this section we provide the description of a communication model using our approach. Data exchange activities among different participants are the fundamental elements of communication systems. Therefore, who, when, where and what are exchanged during the data communication are the major pieces of information required to specify a communication system. We define those who participate (senders and receivers) in the communication as "participants"; the mechanism

used to exchange the data is defined as a “channel”; data that are exchanged as “transmitted data”. Complicated communication systems include other features, such as communication permissions, action constraints, access rights and event trigger sequences.

Our objective is to find a methodology that can be used to model most communication applications quickly and easily, even though complicated communication systems might have stochastic rules.

Inspired by the success of SQL language system, which is consist of Data Definition Language (DDL) and Data Manipulation Language (DML), our approach to modeling communication systems includes two counterparts: Communication Logic (CL) Definition and Communication System Operation. The CL Definition defines the structure and policy for a communication application (Section 2.1). Communication system operation monitors and controls a communication system when it is running (Section 2.2).

## 2.1. Communication Logic Definition

A communication logic model includes information of three aspects: (1) a set of symbols or notation for data communication, (2) the information on the communication status, and (3) the rules of how the model works.

We define a communication logic model as a five tuple:  $CL = (Participant, Channel, Interface, RelationInit, Constraint)$

**Participant** contains all participants in this communication system, i.e.:

$$Participant = \{p_1, p_2, \dots, p_n\}$$

where a participant  $p_i$  is an entity capable of data exchange. A participant can be (1) an instance of some communication logic, which has internal communication logics, or (2) a communication node.

A *conference meeting*, for instance, is a communication logic model, but it can also be a participant of some larger communication logic as long as this conference model has the input and output interfaces to communicate with other participants.

A *node* is the basic communication participant which can not be further decomposed. For example, an instant messenger in a communication is a node.

Participants can be classified into different participant type in order to manage properties of different group easily. Type of participant is *role*.

We recursively define a participant,  $p_i$ , as:

|                                 |
|---------------------------------|
| a node                          |
| a communication logic model CL' |

**Figure 1: Definition of Participant**

**Channel** is the virtual space where data will be exchanged among participants. Participants have to be

connected to channels before data may be exchanged. One channel may connect to multiple participants. If only two participants are connected to one channel, this communication is a one to one communication. If more than two participants are connected to the same channel, this communication is the common conference model.

**Interface** is defined as a list of participants of a communication logic model that are capable of exchanging data with participants outside of this model. The communication logic can only receive or send data from outside via an interface participant. An interface is an important mechanism which enables the recursive definition of a communication logic model.

For completeness, we define that a node itself is an interface to input and output data when it is regarded as a participant.

If a communication logic model contains a certain number of interfaces, this communication logic can be used to replace participants in other communication logic models. The reason is that the communication logic with input and output interfaces can be considered as a participant capable of exchanging data with other participants (either a node or a communication logic model).

**Relation** describes a connection between a participant and a channel. This connection also specifies the interface through which the participant is connected and the data exchange direction between the participant and the channel. The direction can be:

*Send*: participant send data to channel.

*Receive*: participant get data from channel.

*Bi-direction*: participant can both send and receive data from channel.

Therefore, we can define a relation as a three-element tuple.

$$\langle participant.interface, channel, direction \rangle$$

For example, the relation  $\langle p1.o1, c1, send \rangle$  means that the connection relation between participant  $p1$  and channel  $c1$  is that  $p1$  can send data to channel  $c1$  via interface  $o1$ , but  $p1$  can't get data from  $c1$ . One channel can have a relation with multiple participants. These participants communicate with each other through this channel.

The relation in a communication logic model may change during the communication. In our model, we only describe the initial status of relation **RelationInit**. The initial relation combined with the definition of the constraints gives us a clear view of how this communication model works.

Based on the *relation*, we define the **Status** of a communication logic model. The *status* of a communication logic model is defined as the direction of all the relations between every participant and every channel. Suppose there are  $m$  participants and  $n$  channels in a communication logic model, then we can represent

this model's status using an  $m$  by  $n$  status matrix  $\{S\}$ . Each entry of the matrix,  $s_{i,j}$ , is the direction of the communication from participant  $i$  with respect to channel  $j$ . The value of  $s_{i,j}$  shows the status of relation as one of *send*, *receive*, *bi-direction* or *not-connected*. We use the vector (00) as not-connected, (01) as receive, (10) as send and (11) as bi-direction.

*Participant Status* is the status of all relations that involve a specific participant; it can be represented using a row of the status matrix.

*Channel Status* is the status of all relations that involve a specific channel; it can be represented using a column of the status matrix.

**Constraint** The *Constraint* describes the legal transitions between communication logic model *statuses*. It is defined as a function

$$f : E, S \rightarrow S$$

$E$  is the set of events which may affect the connection state of the communication.  $S$  is the set of all legal status of a communication model. The *Constraint* function defines all possible transitions of model states. In other words, we can consider the constraint as some kind of state machine.

Constraint helps to model non-trivial communication requirements and supports automatic system responses in communication applications. For example, suppose we have to build a communication logic model with only two participants and the participants will send messages iteratively, that is one participant can only send a message after the other participant has sent a message. This requirement means that once a participant has sent data to the channel, the relation of this channel to this participant should not have a direction of *send*.

Our definition of a *status* only includes the basic information of all relation states in this model. The purpose of this restriction is to simplify the process of defining a set of constraints. The operation part cover some other kind of rules e.g., the rules of transmitted data type and structure.

A communication logic model defines data flow directions that can be used during data communication. Related to the operation part of this language, the communication logic model defines the initial relation state based on which the real data exchange will occur during operation. Using the recursive structure of this model, it is possible to build a complex communication connection and rules quickly and easily.

The Figure 2 shows a more formal definition of a communication logic model using EBNF. The grammar shown in Figure is an attributed grammar where the subscript "<sub>A</sub>" represents that the entity is an attribute of the entity on the left hand side of the production.

---

```
communication logic ::=
  {Participant}A+ {Channel}A+ {RelationInit}A+
```

---



---

```
{interface}A+ {constraint}A+
Participant ::= communication logic | node
node ::= nameA idA {authority}A+
Interface
channel ::= idA
RelationInit ::= participantIdA interfaceIdA
channelIdA directionA
Interface ::= interfaceIdA
constraint ::= {event status}A+
event ::= idA descriptionA
status ::= idA descriptionA
```

---

Figure 2: Communication Logic Definition

## 2.2. Communication System Operations

The definition of a communication logic model can be considered as some pre-process which may happen off line before the real communication happens. The benefit of this pre-process is that we can avoid the long time process of modeling communication logic at runtime. However, we do not negate the possibility of constructing a communication logic model at runtime. The work of defining a complex communication logic model may be resolved by domain experts. End users can simply use logic models defined for different domains when specifying their communication model.

The modeling part and operation part of our system focus on different views of the communication. The logic model represents a global view of the specific communication activity. It doesn't care if a participant is "local" or "remote". The logic model emphasizes the application of rules to build the model which focuses on the entire communication activity. This logical view allows for the analysis of the communication model.

Unlike the communication logic model, the communication operation focuses on the communication activity from one user's point of view. The key point of operation is how to manage the communication specification and data exchange activities of a current user. After loading the pre-defined communication logic model, a user will carry out his communication activity in the communication operation part. This part implements the channel constraints defined in communication logic model definition, and furthermore supports how the different media are exchanged and displayed.

The approach for the system operation part is based on the previous work of the Communication Virtual Machine (CVM) by Deng et al. [5]. The communication operations considered in CVM includes: connection initiate, connection close, data or data stream transfer and grouping the transferred data so that the receiving sides become aware of their association [8]. For additional details of CVM, please refer to [5]. A definition of the Communication Modeling Language (CML) [2, 3] used in CVM is shown in Figure 3.

1. `userSchema ::= local connection {connection}`
2. `connection ::= mediaAttached connection remote {remote}`
3. `local ::= person isAttached device`
4. `remote ::= device isAttached person`
5. `mediaAttached ::= {medium} {form}`
6. `device ::= device deviceCapability {deviceCapability}`
7. `form ::= {form} {medium} | form`
8. `person ::= personNameA personIDA personRoleA`
9. `device ::= deviceIDA`
10. `medium ::= builtinTypeA mediumURLA suggestedApplicationA actionA`
11. `deviceCapability ::= builtinTypeA`
12. `form ::= suggestedApplicationA actionA`
13. `actionA ::= "send" | "doNotSend" | "startApplication"`

**Figure 3: System operation scope**

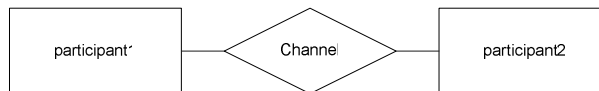
The CML definition shown in Figure 3 is implemented using a graphical representation (G-CML) and an XML representation (X-CML). For additional details of CVM, please refer to [3]

### 3. Application of the Model

In this section we present two examples that illustrate how communication applications can be modeled using our methodology. The cases studies show how the requirements of a real scenarios are modeled using the communication logic model described in Section 2. The two examples emphasize different aspects of the communication logic model. The first example focuses on how to model constraint requirements and the second example shows how to model a complex network connection with a large number of participants.

#### 3.1. Example 1

Let us consider an example of a communication logic model involving only two participants (Figure 4). One requirement of the communication is that the two participants can only send messages iteratively. That is, a participant can send a message only after the other participant sends a message; once a participant sends a message, he can't send anymore messages until the other participant sends a message. We assume that both participants' initial status can send or receive a message. Figure 4 shows an abbreviated graphical representation of the communication model.



**Figure 4: A communication example between two participants**

We can define the initial communication logic as:

```

{Participant = {participant1, participant2},
 Channel = {channel1},
 Relation = {<participant1, channel1, send>,
 <participant2, channel1, receive>}
 Interface = {}
 Constraint = ...
}

```

The *Relation* is very easy to build while the most difficult part of building this communication logic model is how to define the *Constraint*. The *Constraint* of this model is described after we describe the *Relation*.

We have mentioned earlier that we can represent a communication logic model's status using an  $m$  by  $n$  status matrix, where  $m$  is the number of participants and  $n$  is the number of channels. In this example, there are two participants and there is only one channel. As in the following matrices, each entry of the status matrix could be any value of {S(end), R(eceive), B(i-direction), U(nconnected)}.

|         |              |              |
|---------|--------------|--------------|
|         | Participant1 | Participant2 |
| Channel | S/R/B/U      | S/R/B/U      |

There are only three legal state matrices:

The initial state matrix:

|         |              |              |
|---------|--------------|--------------|
|         | Participant1 | Participant2 |
| Channel | Bi-direction | Bi-direction |

The legal state matrix after initiation:

|         |              |              |
|---------|--------------|--------------|
|         | Participant1 | Participant2 |
| Channel | Send         | Receive      |

|         |              |              |
|---------|--------------|--------------|
|         | Participant1 | Participant2 |
| Channel | Receive      | Send         |

The first matrix shows the initial states of each participant, where each participant can either send or receive a message at first. The second matrix shows the connection status when participant1 is connected to channel and ready to send a message while participant2 is about to receive message. The third matrix shows the connection status when participant2 is connected to channel and ready to send a message while participant1 is about to receive message.

The constraint on the communication can be represented as an automata machine as in Figure 5.

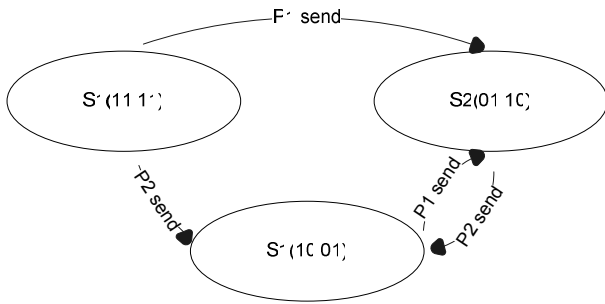


Figure 5: The Communication Logic Constraint

### 3.2. Example 2:

In this example we illustrate how to build a complex communication logic model using our methodology. Suppose that we are given the following communication scenario. There is a regular teleconferencing between the heart divisions of the Miami Children's Hospital (MCH) and Arnold Palmer Hospital (APH) in Orlando Florida three times each week. The doctors from these two hospitals discuss different topics on different days which require different types of data to be shared, such as text, images and voice recordings [1, 6].

For this scenario, we first define the communication logic sub-models for both MCH and APH respectively. After that we construct the complete model describing communication between these two sub-models where each sub-model is regarded as a participant.

We define the sub-model for MCH as a conference model at MCH.

```
MCH_conference =
{
  Participant = {doc1, doc2, ..., p_interface}
  Channel = {conference1}
  Initial relation =
{<doc1, conference1, bi-directional>,
 <doc2, conference1, bi-directional>...
 <p_interface, conference1, bi-directional>}
  Interface = {p_interface}
  Constraint = {}
}
```

The participant "p\_interface" is a virtual participant defined as an interface of the whole conference model. When the MCH conference model works as a participant and communicates with other participants, data exchange of this model (MCH\_conference) goes through the virtual participant "p\_interface".

Here we assume that this communication will use the simplest default constraint for communication model. The status of the model will remain the same during communication, so there is no need to define constraint

functions. Obviously we can also define complex constraints for the conference inside the MCH without affecting the cooperation of this model with others.

The APH conference model can be defined similarly to the MCH conference model.

A simple model of constructing one channel between the MCH conference model and APH conference model satisfy this complex communication requirement.

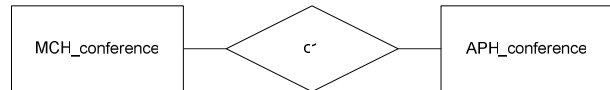


Figure 6: MCH\_APH\_conference communication logic model

```
MCH_APH_conference =
{
  Participant = {MCH conference, APH conference}
  Channel = {c1}
  Initial relation =
{<MCH_conference.p_interface, c1, bi-directional>,
 <APH_conference.p_interface, c1, bi-directional>}
  Interface = {}
  Constraint = {}
}
```

This communication logic model can be used for the regular meetings between these two hospitals.

The main difficulties of modeling a complex communication can be divided into two groups:

Model the communication with complex relation state.

Model the rules of how the communication status changed.

Support different transmitted data types for different communications.

This example illustrates how to build a complex model by recursively defining three simple communication logic models. The complex relation structure can be decomposed into simple relation structure piece in sub-logic models during the process of recursively defining the communication logic model. The rules which may be difficult to captured, especially for large scale systems, are distributed in different channels thus simplifying the whole process of defining a complex system. We can build up small logic models and make use of them later to construct larger one.

In our second example, doctors who attend the meeting between the two hospitals just apply to attend to their own hospital's conference model. Once they can communicate within their local hospital conference, the MCH\_APH conference model will handle the big meeting between two local conferences. Because the communication logic model only focuses on the status defined in this paper without considering the different

types of data exchanged in the model, the same logic model can be used to exchange different data.

#### 4. Conclusions

In this paper, we proposed an approach to model communication systems for rapid application development. Our approach has the following benefits: it decomposes communication modeling difficulties and resolves them separately; provides a unified interface between roles/systems; provides a recursive approach to model communication systems and the adoption of multicast-conference model; and provides a constraint mechanism supporting the definition of rules by user. These benefits enable us to build a common communication system easily and quickly while keep the capability to build very complicated communication systems.

We describe an application of our methodology using two examples to show that our approach is effective and efficient in modeling communication systems.

The future work will focus on modeling more general communication systems, providing theoretical analysis of system complexity and modeling capability. In addition, we plan to develop a reliable GUI system for builders to create communication logic models.

#### References

[1] R. P. Burke and J. A. White. Internet Rounds: A congenital heart surgeon's web log. In Seminars in

Thoracic and Cardiovascular Surgery, 16(3):283-292, 2004.

[2] Peter J. Clarke, Vagelis Hristidis, Yingbo Wang, Nagarajan Prabakar and Yi Deng. A Declarative Approach for Specifying User-Centric Communication, In Proceeding of the International Symposium on Collaborative Technologies and Systems (CTS 2006) IEEE, May 2006, pages 89 – 98.

[3] Peter J. Clarke, Vagelis Hristidis, Yingbo Wang, Nagarajan Prabakar and Yi Deng. "A Declarative Approach for Specifying User-Centric Communication" Technical Report FIU-SCIS-2006-01-02

<http://www.cis.fiu.edu/~prabakar/cts2006/> (June 2006).

[4] DisasterHelp. (Jun. 2005).

<https://disasterhelp.gov/portal/jhtml/index.jhtml>

[5] Yi Deng, S. Masoud Sadjadi, Peter J. Clarke, Chi Zhang, Vagelis Hristidis, Raju Rangaswami, and Nagarajan Prabakar. A Communication Virtual Machine. Proceeding of the 30th Annual International Computer Software and Applications Conference (COMPSAC 06). . Accepted April 2006.

[6] Teges™ Corporation.

<http://www.teges.com/index.asp> (Dec. 2005).

[7] W. Weaver and C. E. Shannon, *The Mathematical Theory of Communication* Illinois, 1949.

[8] Chi Zhang, S. Masoud Sadjadi, Weixiang Sun, Raju Rangaswami, and Yi Deng. User-centric communication middleware. Tech. Report FIU-SCIS-2005-11-01, Nov. 2005.