Visual Specification of Component-based Slow Intelligence Systems

Shi-Kuo Chang\textsuperscript{1}, Yingze Wang\textsuperscript{1} and Yao Sun\textsuperscript{1}
\textsuperscript{1}Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA
{chang, yingzewang, yaosun}@cs.pitt.edu

Abstract

A new approach for the visual specification of component-based software systems is proposed. The main idea is to employ two visual representations - the I-card (information card) and the C-card (control card) - together to specify a component-based software system. This approach is especially useful in the specification and design of component-based slow intelligence systems. This approach is currently being applied to the design of slow intelligence systems for social influence analysis.

Keywords

Visual specification, component-based software system, slow intelligence system, super component

1 Introduction

In the specification of component-based software systems, control and timing information is often unclear or even lost. If a formal, mathematical approach is used, control and timing information could be buried in the details and become unclear to the designer/programmer. If a less formal approach is used, such information could be lost.

In this paper, a new approach for the visual specification of component-based software systems is proposed. The main idea is to employ two visual representations - the I-card (information card) and the C-card (control card) - together to specify a component-based software system. This approach is seemingly simple, but actually quite powerful. This approach is also especially applicable to the specification and design of slow intelligence systems where super components are employed to search for solutions.

A slow intelligence system (SIS) is a general-purpose system characterized by being able to improve performance over time through a process involving enumeration, propagation, adaptation, elimination and concentration [Chang2010]. A SIS is a component-based software system, but it differs from ordinary component-based systems by employing super components, in the sense that multiple, similar components can be activated either sequentially or in parallel to search for solutions. The proposed visual specification approach is especially suitable for the specification and design of such systems with super components.

The paper is organized as follows: Section 2 explains in detail this visual specification approach using dual visual representations. In Section 3, we describe the user interface design to produce and manage the dual visual representations for the generic SIS system. Finally, Section 4 discusses future works.

As described in [Chang2011b], we are applying this approach to design the slow intelligence system for social influence analysis (SIA).

2 Visual Specification of Component-based Software Systems

2.1 Dual Visual Representations

The main idea is to employ two visual representations - the I-card (information card) and the C-card (control card) - together to specify a component-based software system. A generic example is illustrated in Figure 2.1. The I-card specifies the logical relationships among the components, and the C-card specifies the control and timing relationships among the components. Their inter-relationships are represented by bi-directional arcs (the dotted lines) connecting components, or other entities, in the I-card and the C-card.
2.2. An Example

In the following example, the two components X1 and X2 can be either simple components or super components. We will first give an example involving only simple components. Suppose there are two components X1 and X2. The component X1 is invoked first, followed by the invocation of component X2.

As shown in Figure 2.2, the two components are specified in the I-card as two rectangles. This specification for components can be further refined. For instance each component can be described by a detailed class diagram in UML. The timing relationships among the two components are specified by the Petri net in the C-card, where places are represented by small circles, and transitions by vertical rectangles. The Petri net’s transitions in the C-card correspond to the components in the I-Card, therefore they are linked by dotted lines.

Although a Petri net is used in the above example, we can also replace the Petri net by other kinds of timing diagrams such as the sequence diagram or the activity diagram in UML. We can now give another example as illustrated by Figure 2.3. This example illustrates the simplicity, flexibility and versatility of this approach. We can use different diagrams in the I-card and the C-card to present different views of the component-based software system.

2.3. Super Components

As mentioned before, in the above example the two components X1 and X2 can be either simple components or super components. A super component X1 is a collection of similar components x11, x12, ..., x1j, retrieved from some component database or generated by some algorithm. As shown in Figure 2.4 the visual representation of a super component is a box with double-edged border.

With super components, the Petri net in the C-card is actually a concise representation of a much larger underlying Petri net, such as the one shown in Figure 2.5. It should be pointed out this is by no means the only possible expanded Petri net. As noted above, depending upon the meaning of the association between the I-card and the C-card, different expanded Petri nets can be generated.
2.4. SIS Design using Super Components

The super component can also be further refined. A super component is essentially a concise representation of a collection of components. Therefore we can employ a generic design pattern or a component generator to specify such a super component.

3 User Interface for SIS System

In this section we describe the user interface design to create and manage the dual visual representations for the SIS system. Based on SIS system framework, the user interface can be designed for three specific tasks: (1) Interface for designing control card. In our generic SIS system, control card (C-Card) can define functional dependency repertoire. (2) Interface for designing information card (I-Card). For SIS system, I-card can define candidate functions repertoire. (3) Interface for designing testing data, which in our SIS system is stored in test data repertoire.

3.1. User Interface for designing C-card

Each functional component is a super component or a simple component that can be represented by a transition in a Petri Net. Here super component is a collection of components; while simple component is a single component. We need to define the dependencies between
different functional components. User interface is used to draw Petri Net as shown in Figure 3.1.

We developed our user interface based on [Bonet 2007] PIPE tool, which is a powerful software for editing Petri Net. In Figure 3.1, particularly, unfilled rectangle represents transition for defining simple component. Filled rectangle represents super component. Circles represent the places. User can utilize the Petri-net to define the sequence and dependency among functional components. Moreover, we extend the meaning of Petri Net that places can represent the messages sending from one transition to another, as shown in Figure 3.2. Since slow intelligence system uses messages to communicate among each component. Thus, in message editor, user can define a list of messages related to the particular place, which enhance the power of Petri Net. Note that messages can be automatically mapped to MsgID defined in system, so that users don’t need to know the exact MsgID. After designing the Petri-net for a specific system, user can save the diagram in xml format, such as the Petri Net Markup Language (PNML), which is an XML-based syntax for high-level Petri nets designed as a standard interchange format for Petri net tools [Weber2003].

We give an example as shown in Figure 3.1. This is a simple Petri Net with two transitions named “Enumerator” and “Diffusion Model”. Enumerator represents a simple component while “Diffusion Model” represents a super component. This Petri Net shows that Diffusion Model depends on Enumerator. Such dependency relationship indicates that super component Diffusion Model should execute after Enumerator’s execution. Also Enumerator can send message “Enumeration Notify” to Diffusion Model edited by place P1. The saved xml file of this Petri Net in PNML format is:

```xml
<pnml>
<net id="net1">
  <name>
    <text>System C-Card</text>
  </name>
  <page id="page1-net1">
    <place id="p0">
      <graphics>
        <position x="30" y="55"/>
        <offset x="0" y="-10"/>
      </graphics>
    </place>
    <place id="p1">
      <graphics>
        <position x="180" y="50"/>
        <offset x="0" y="-10"/>
      </graphics>
      <messages>
        <value>521 Enumerator Diffusion Model</value>
      </messages>
    </place>
    <transition id="Enumerator">
      <graphics>
        <position x="115" y="55"/>
        <offset x="0" y="0"/>
      </graphics>
    </transition>
    <transition id="Diffusion Model">
      <graphics>
        <position x="265" y="50"/>
        <offset x="0" y="0"/>
      </graphics>
    </transition>
    <arc id="a0" source="p0" target="Enumerator"/>
    <arc id="a1" source="Enumerator" target="p1"/>
    <arc id="a2" source="p1" target="Diffusion Model"/>
    <arc id="a3" source="Diffusion Model" target="p2"/>
  </page>
</net>
</pnml>
```

### 3.2. User Interface for designing I-card

After designing the dependencies among functional components for SIS generic system, the user needs to devise the information card for each component (super component and/or simple component). For a super component containing a number of candidate components, we need to use I-Card to add each candidate component from database or from the template algorithm with different parameters to this super component. For a simple component in Petri-net, although there is only one component, we still need to add it from database or specific algorithm. In slow intelligence system, there are two types of functional components: system components like enumerator, eliminator, etc; algorithm components like diffusion model for social influence analysis. Each type of component can be simple or super. Usually system component is simple one while algorithm component is super one. The user interface design is shown in Figure 3.3.

When user right clicks on transition, the transition editor
will pop up. User can define the component type in comb box and edit the corresponding component I-Card. If it is the existing one, the user can load it. The system component I-Card editor interface contains a comb box to define name of component. Each type of system component has specified Key. When user selects one, the corresponding key will shown on table. The user only needs to define the value of the Key. Figure 3.3 gives an example. The system component I-Card is saved as a .txt file using format:

$BEGIN and END are keywords
BEGIN ConcurrentComps
5
END ConCurrentComps
Algorithm component I-Card editor contains three Panels. In panel “property”, if the transition is filled rectangle, the type should be super component and the user can add each candidate component to this super set. If the transition is unfilled rectangle, the type should be simple component and the user only needs to add one candidate component. Since the candidate (simple or super) components in each functional block do the same job using different algorithms, thus in the interface, “Functions” edit box describes the job of this functional block. User should choose the way that enumerator of SIS system (Component generator) performs, using all the candidate components or using only one of them. If user chooses “one candidate component”, he/she should input the default candidate ID number that will be utilized. The list on right shows the name of “Candidate components” that user added. Also user can click “details” tab to see the details of each candidate component in a table. For each algorithm functional component (I-Card), there is a test data set for testing each candidate component in this block. Thus in panel “Testing Data”, user can edit the dataset by T-Card editor in next subsection. This algorithm component I-Card can be saved as xml format.

We give a simple example as shown in Figure 3.3. This I-card corresponds to transition “Diffusion Model” in Figure 3.1, which is a super component containing two candidate components. The candidates are from template algorithm with different parameters. Thus the user can load the template and set different parameters as shown in Figure 3.3. Since the user defines by enumerating all candidate components, thus there is no default one. The saved xml file of this Petri Net could follow the format:

```xml
<Msg>
  <Head>
    <Name></Name>
    <Description>
      <Head>
        <Body>
          <Item>
            <Key>Candidate Name</Key>
            <Value>Diffusion Model 2</Value>
          </Item>
          <Item>
            <Key>ID Number</Key>
            <Value>2</Value>
          </Item>
          <Item>
            <Key>Template</Key>
            <Value>Yes</Value>
          </Item>
          <Item>
            <Key>Parameter</Key>
            <Value>3.0</Value>
          </Item>
          <Body>
            <Item>
              <Key>Template</Key>
              <Value>Yes</Value>
            </Item>
            <Item>
              <Key>Parameter</Key>
              <Value>3.0</Value>
            </Item>
          </Body>
        </Head>
      </Description>
    </Head>
  </Body>
</Msg>
```

Tab <Name> shows the name of I-Card. Tab <Description> shows brief description of this I-Card. Tab <Key> indicates the key text such as: Type, Candidate Name, etc. Tab <Value> indicates the value of user’s input in textbox. Thus, <key> and <value> should be pairs. For example, some pairs as shown in Figure 3.3 are:

3.3. User interface for designing test data

Figure 3.4 shows the interface of designing testing data for testing each candidate component in algorithm functional component block. This split panel includes two parts. The left part is used to load testing data from existing files and to specify the output results property. User can browse existing file and define the place to store the result. The right part is designed for the purpose of simple testing and debugging. User can manually edit the data on the table. The input data could be any value including number, string, boolean variable. The output data is the correct result corresponding to input. Then user can compare their algorithm’s results to these standard results.

We give an example as shown in Figure 3.4. This T-card corresponds to algorithm component I-Card defined in Figure 3.3. The T-Card is saved as a .txt file using format:

```plaintext
$BEGIN and END are keywords
BEGIN InputDataList
/*List test data files*/
Twitter.txt D:\SIS\Data\Twitter.txt
END InputDataList

BEGIN OutputProp
/*List output results property*/
txt D:\SIS\Result
END OutputProp
```

We give an example as shown in Figure 3.4. This T-card corresponds to algorithm component I-Card defined in Figure 3.3. The T-Card is saved as a .txt file using format:
Figure 3.3. Screenshot for I-Card design.
4 Discussion

In this paper we described a new approach for the visual specification of component-based software systems by employing two visual representations - the I-card (information card) and the C-card (control card). The user interface for visual specification of component-based software system is currently being developed. For practical convenience, a T-card can be used to specify test data.

We can use different diagrams in the I-card and the C-card to present different views of the component-based software system. It is also possible to associate multiple diagrams for multiple views. Furthermore, the bi-directional arcs between the two visual representations can be labeled to specify the meaning of the associations so that different interpretations can be generated from the visual specification. How to check the consistency of multiple views, and how to synthesize a consistent spec from multiple views, will be topics for further research.

References


