xtUML-COP: xtUML modeling tool with COP extensions

SHINTARO TAKENA[†1] , KENJI HISAZUMI[†1]

Abstract: Recent Embedded System like IoT product development requires behavior that responds to changing environments. The changing environment is called context, and it cross-cutting concerns multiple classes when implemented. This cross-cutting concern makes development difficult because it is not grouped together as a module. Context-oriented programming (COP) uses COP-specific modules called layers to modularize cross-cutting concerns. However, the design of COP behavior is difficult because of the lack of models and modeling tools supporting the behavior. Authors proposed models of behaviors, which extend UML state machine diagrams. In this paper, we propose modeling tools that extend the previously proposed UML state machine diagrams with COP. The tool can handle state machine diagrams separately for normal and layered states to separate two types of states. Also, it is possible to show or hide states within layers according to the activation status of each layer. Therefore, state machine diagrams can be displayed according to layer activation status. We implemented this tool using Sirius, which can develop original domain-specific modeling language tools. As a result, the tool can deal with layers in COP on UML state machine diagrams. We were also able to represent layer activation by implementing a layer hiding function. Future issues are further improvement of usability, relationship with UML class diagrams, and code generation.

Keywords: Context-oriented programming, Layer, UML state machine, xtUML

1. Introduction

Embedded systems, as typified by the Internet of Things (IoT) and Industry 4.0, require the ability to change their overall behavior in response to ever-changing external or internal environments. The environment in which such systems are placed is called the context. Context-Oriented Programming (COP)[1] is a method for constructing software that behaves according to context. A layer consists of a set of partial methods whose context-specific behavior is described. The layer has two states, activated or deactivated, and the partial methods are executed when the layer is activated.

This paper proposes an xtUML-COP tool that supports COP extended eXecutable Translatable UML(xtUML). As a feature, the entire flow, which changes according to conditions, can be shared while switching in a single diagram. Also, the tool is easy to extend so that anyone can try out new concept of executable model.

The paper structured as follows. Section 2 describes the details of the proposed method, Section 3 introduces the features of the modeling tool and future issues, and Section 4 summarizes the paper.

2. Layer Hierarchical State Machine Diagram

In this section, we describe our previously proposed layer hierarchical state machine diagram[2]. In the proposed method, the state in which all layers are deactivated is represented as a base layer to represent the basic behavior. Then, the behavior when a layer is activated is represented as tracing paper superimposed on the diagram described in the base layer. In addition, since multiple layers may be activated at the same time, the difference in behavior is necessary.

Figure 1 shows the state machine diagram before adapting the proposed method. Examples of the proposed method are shown in Figures 2 and 3. Figure 1 represents the movements of a person depending on the weather. The basic behavior of a person is to stand still, run when it is sunny, and walk and hold up an umbrella when it is raining. Figure 2 shows the representation of a sunny day using the proposed method. In the figure, the sunny layer is superimposed on the state machine diagram of the base layer. In this way, it represents the addition and modification of behaviors.

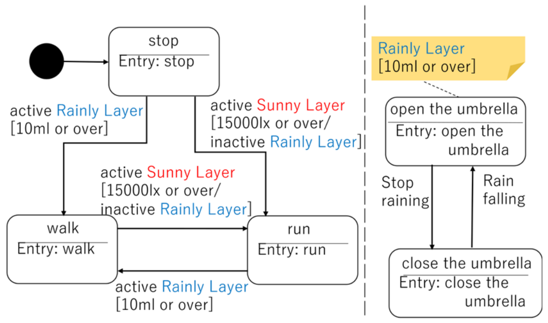

Figure. 1 Layers defined usingexisting state machine diagrams

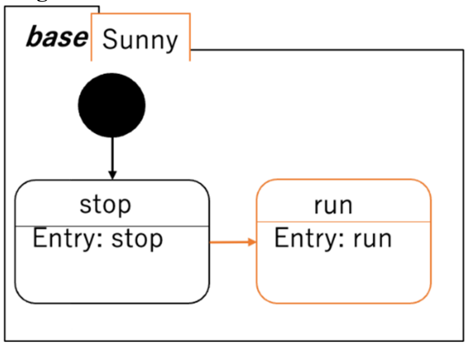

Figure. 2 Sunny Layer

3. Modeling Tools

This section describes the modeling tool. The tool features a state machine diagram that is extended by multiplying the layers of a COP with reference to the layer function used in illustration tools. In addition, it is possible to place states related to multiple

†1 Shibaura Institute of Technology.

modules that change according to conditions in a single diagram. In existing tools, there was a problem of difficulty in understanding the flow between the state machine diagram, in which the flow is shared by the team by looking at the entire diagram, and the COP, in which the overall behavior changes depending on the activation conditions. This tool allows users to share the flow of overall changes that vary according to conditions, switching from one layer to another. And although the layer function is a function that does not exist in existing state machine diagrams, it is a function that is well known by users because it has been introduced in many general tools. Therefore, users can use this extended function without any sense of unease.

Figure 3 shows the meta-model of the tool. The upper side is a metamodel related to class diagrams. The lower part is a metamodel related to the state machine diagram, which also includes classes of layers that are extensions. Figure 4 is a screen shot of the Sirius[3] development tool. On the right side, the developed modeling tool is displayed. Figure 5 shows the layer when it is activated. The black frame indicates the layer, and the states within the layer are displayed in red. Since the layer is a transparent image with a black border, the state is visible even when the layers are superimposed. Figure 6 shows all layers. Internally, as shown in Figure 6, layers exist, but they are hidden on the screen. Although this is a prototype, it is possible to generate code from the state machine diagram created with this tool.

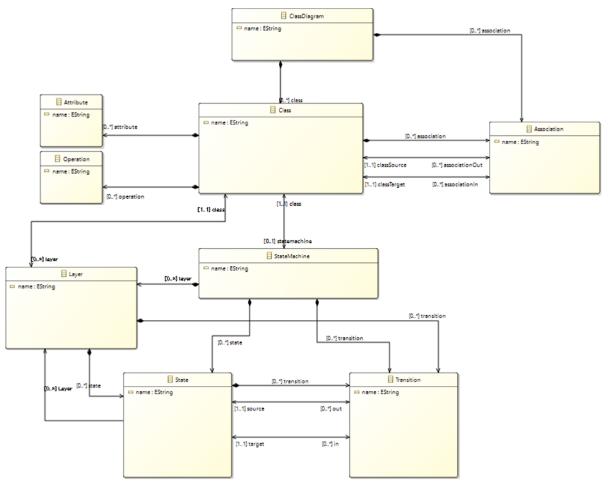

Figure. 3 Metamodel

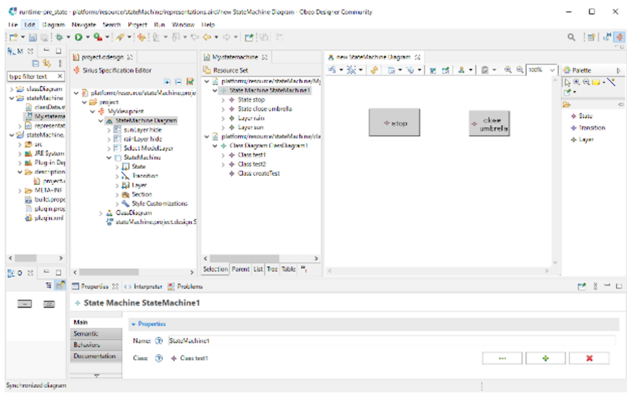
runtime-pre_state - platform/resource/stateMachine/representations.aird/new StateMachine Diagram - Obeo Designer Community
File Edit Diagram Navigate Search Project Run Window Help
stop
close umbrella
Palette
State
Transition
Layer
Properties
Interpreter
Problems
State Machine StateMachine1
Main
Semantic
Behaviors
Documentation
Properties
Name: StateMachine1
Class: Class test1

Figure. 4 Tool Development Screenshot

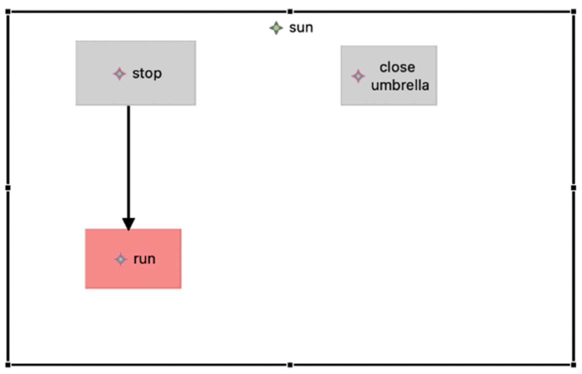

Figure. 5 Screen with Sunny Layer Overlaid

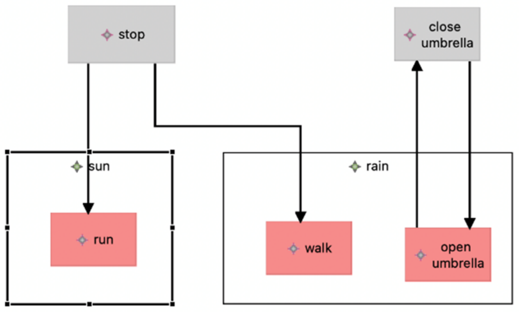

Figure. 6 Screen with all layers displayed

3.1 Modeling Tool Goals

Programs that solve cross-cutting concerns such as COP are more complex than existing programs because they create new modules between modules. Therefore, the benefits of generating source code from models are even greater. The goal of this tool is to generate source code from models that make complex configurations easy to understand.

4. Conclusion

This paper introduces a previously proposed modeling tool for extended state machine diagrams. It is characterized by the use of transparent images for layers and the ability to simultaneously display states inside and outside of a layer when layers are stacked. The challenges are that it is not possible to manipulate states outside of a layer when layers are stacked, and it is difficult to prioritize layers. Future work includes improving the usability of the states and implementing code generation.

Reference

[1] Robert Hirschfeld, Pascal Costanza, Oscar Nierstrasz: "Context-oriented Programming", in Journal of Object Technology, vol.7, no.3, pp.125-151, 2008.

[2] Shintaro Takenaka, Kenji Hisazumi:" Context-oriented Design Method for UML State Machine Diagram" APRIS2021,2021

[3] "Sirius Homepage" https://www. eclipse. org/sirius/ (accessed 2022-9-18).