4 X − 8

Rule-based Simulation Control in an Object-oriented Environment

Emily Candell Yasuyuki Tauchi
SECOM Intelligent Systems Laboratory
SECOM Co Ltd

1 Introduction

In order to develop a training simulator*, we have developed Tec++, an object-oriented expert system shell. The system employs knowledge objects by combining object-oriented functionality, to represent the entities which exist in the simulated environment, with rule-based reasoning techniques, to describe the behavior of the entities in the simulation.

In most object-oriented systems, object control is handled by an embedded scheduling mechanism. This method limits the flexibility in the user's control of objects. Simulation of a real-world system requires the user to accurately control the occurrence of simulated events. In this object-oriented shell, delegation of control among the objects is handled by a knowledge object. The rules of this knowledge object, the Scheduler Object, are used to govern which object is in control of inferencing at any time. Additional scheduling rules can be defined to control the objects in the simulation, thus giving the user more flexibility in overall object and event manipulation.

2 Tecalbe

Tecalbe is a real-time, rule-based expert system shell.† It was developed using EA+†, an adaptation of Scheme[ea89] which supports event handling.

Tecalbe uses inferencing to correlate a rulebase with the system's working memory (WM). The inference engine of Tecalbe uses a Rete Network [For82] to quickly match working memory elements (WME) with the left-hand-side (LHS) of the pattern rules in the rulebase.

3 Tec++

Tec++ combines Tecalbe's rule-based inferencing capabilities with general object-oriented functionality, including classes, objects and message passing.

3.1 Classes

3.1.1 Rules

In an object-oriented environment, classes may be used to define the behavior of a group of related entities. In Tec++, a set of rules is associated with each class. These rules represent the behavior of all objects created from this class. The rules are defined using the if-then form employed by the OPS5 Rule-Based System[BFKM85].

3.1.2 Inheritance

At class creation time, a super class may be specified. A class inherits the rules defined in its super class. The user can define rules in the sub-class which will override its super class' rules in order to create a variation of a previously defined class. If a rule of the same name as a super class rule is defined in a class, the new rule will replace the super class rule.

*This simulator will be used for the training of SECOM security system operators.

†Developed at SECOM Intelligent Systems Laboratory

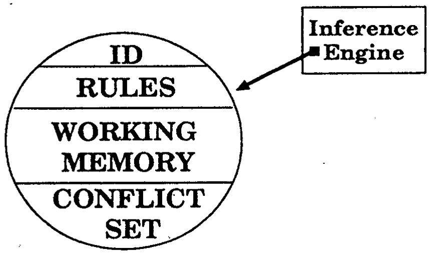

Figure 1: Knowledge Object

3.2 Knowledge Objects

From each class which is defined, objects can be instantiated. Tec++ uses a knowledge object to represent each entity of the simulation. These objects are similar to the knowledge objects used in [MUT87]. An object's behavior is governed by the rules defined in the class from which it was instantiated. All objects which are created from the same class behave in the same way.

Each knowledge object consists of four parts (See figure 1):

- ID: Gives an object a unique name by which it can be referred in message passing
- Rules: Govern the behavior of a knowledge object with respect to the simulated entity which it represents
- Working Memory: Contains working memory elements which describe the current status of the object during the simulation
- Conflict Set: Contains the set of rules which are fireable in an object at any time during the course of the simulation

The object in control at any time functions in the same way as Tecalbe. Tecalbe's inference engine is shared between all the objects which were defined for the simulation (See figure 1). The object which controls the inference engine at any time is determined by a specialized knowledge object, the Scheduler Object, which will be described in Section 3.4.

3.3 Message Passing

In order to simulate a system accurately, the entities being simulated must have the ability communicate with each other. In Tec++, interaction among the simulated entities is realized by means of message passing among knowledge objects. A knowledge object has the ability to send messages to the other knowledge objects.

All messages which are sent among the objects in the system are first intercepted by the Scheduler Object. The Scheduler Object will be described in more detail in Section 3.4. Ultimately, the message will be forwarded to the receiving object. When a message is forwarded from the Scheduler Object to the receiving object, its contents are inserted into the WM of the receiving object. The inferencing context is changed to that of the receiving object. The new WME may trigger inferencing in the receiving object depending on the contents of its working memory. When inferencing is completed in the receiving object, control is returned to the Scheduler Object.

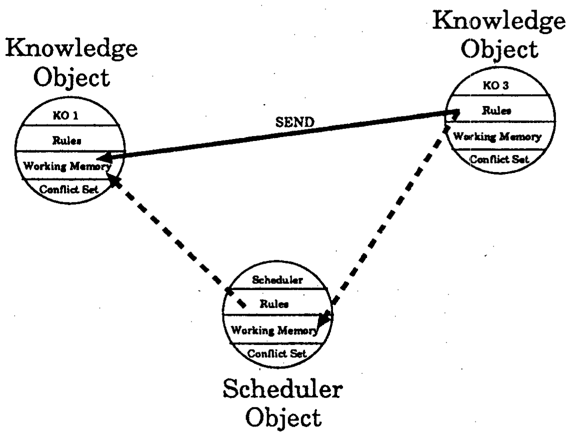

Figure 2: Message Passing in Tec++

3.4 Scheduling

In Tec++, a specialized knowledge object, the Scheduler Object, intercepts all messages which are sent among the objects in the system (See figure 2). The message becomes a WME in the WM of the Scheduler Object. The message WME includes such attributes as a message's receiving object, forwarding time, contents, and other information which may be useful for scheduling the objects in an application.

In a simulation, it may be necessary to delay the sending of a message in order to accurately represent when an event will occur. The Scheduler Object uses a user-specified delay to determine the forwarding sequence of messages. The forwarding time in the message WME indicates the time during the simulation that a message should be sent from the Scheduler Object to its receiving object.

The scheduler in Tec++ is a knowledge object. All message are scheduled according to the rules which exist in the set of rules associated with the general scheduler class (See figure 3). To control scheduling, the Scheduler Object periodically compares the current real time, in the form of a WME, to the forwarding time of all existing message WMEs. If a message's forwarding time is less than the current time, the message is forwarded to its receiving object. If more than one message can be sent at a given time, they will be sent on a FIFO basis. If there is no message having a forwarding time which is less than the current time, the Scheduler Object will update the time WME, and wait until a message is ready to be sent. These scheduling procedures are controlled entirely by the rules defined in the general scheduling class.

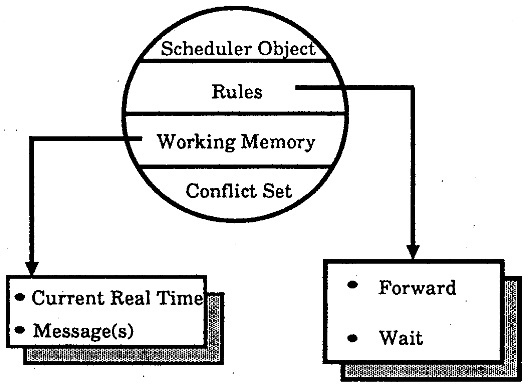

Figure 3: Scheduler Object

Similar to the other knowledge objects in the system a sub-class can be created with the pre-defined general scheduling class as a super class. In the set of rules associated with this subclass, the user can define rules which control specific scheduling needs of an application. User-defined scheduling rules can be used to control such things as initialization of the status of the objects in a simulation and simulation duration. The user can use any of the attributes of the message WME to control scheduling in the application.

4 Conclusion

By using a knowledge object, the Scheduler Object, to control the objects of the simulation, we are preserving the modularity inherent in object-oriented systems. Since all messages must pass through the Scheduler Object, simulation control is centralized by means of the scheduling rules. The rules which can be defined in the scheduler's sub-class give the user more flexible control of objects with respect the real-world system which is being simulated.

References

[BFKM85] Lee Brownston, Robert Farrell, Elaine Kant, and Nancy Martin. *Programming Expert Systems in OPS5 : An Introduction to Rule-Based Programming*. Addison Wesley, 1985.

[ca89] H.Abelson et. al. Revised$^{3.99}$ report on the algorithmic language scheme. Technical report, MIT Artificial Intelligence Laboratory, August 1989.

[For82] Charles L. Forgy. RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, 19:17--37, 1982.

[MUT87] Takeo Maruichi, Tetsuya Uchiki, and Mario Tokoro. Behavioral simulation based on knowledge objects. In *Conference Proceedings of ECOOP '87*, 1987.