State, Event, Time and Diagram in System Modeling

Jin Song Dong (dongjs@comp.nus.edu.sg)
Computer Science Department, National University of Singapore

Abstract

The design of complex systems requires powerful mechanisms for modeling state, concurrent events, and real-time behavior; as well as for visualising and structuring systems in order to control complexity. Methods integration has become a recent research trend in software specification and design. In the graphical area, many object-oriented methods have merged into one, the Unified Modeling Language (UML) which combines various diagrammatic modeling techniques to model static and dynamic aspects of software systems. Although traditional formal methods have not scale-up well, new integrated formal methods show great promise. This tutorial will present the state of the art in formal modeling techniques (state-based Object-Z and event-based Timed CSP), their integration (TCOZ), and transformation techniques from the integrated formalism to UML diagrams. An XML web environment for projecting integrated formal models to UML diagrams will also be demonstrated.

1. Overview

In recent years, *integration* has been a popular approach in unifying programming theories [5], standardizing diagramatic notations [10] and combining formal design methods [1, 4].

The aims and objectives of this tutorial are to learn

- the state-based and event-based formal modeling techniques, Z/Object-Z [3, 12] and CSP/Timed-CSP [11]
- the powerful semantic integration of state-based and event-based formalisms TCOZ [9, 8, 2, 7]
- the transformation techniques between the integrated formalism and UML [13]

2. Topics

This tutorial covers the following topics.

2.1. Object-Z

Object-Z [3, 12] is an object oriented extension of the Z formal specification language. It improves the clarity of large specifications through enhanced structuring. The main Object-Z construct is the *class* definition. A class is a template for *objects* of that class: for each such object, its states are instances of the class' state schema and its individual state transitions conform to individual operations of the class. An object is said to be an instance of a class and to evolve according to the definitions of its class. The standard behavioural semantics of Object-Z classes is as transition systems. The transition system begins in a legal initial state and then evolves through a series of state transitions each effected by one of the class operations. The standard semantic model of transition system behaviour is a pair consisting of an initial state binding and a list of the operations invoked on the system in the order they are invoked.

2.2. Timed CSP

Timed CSP [11] extends the well-known CSP (Communicating Sequential Processes) with timing primitives. CSP is an *event* based notation primarily aimed at describing the sequencing of behaviour within a process and the synchronisation of behaviour (or *communication*) between processes. Timed CSP extends CSP by introducing a capability to consider temporal aspects of sequencing and synchronisation. CSP adopts a symmetric view of process and environment. Events represent a co-operative synchronisation between process and environment. Both process and environment may control the behaviour of the other by *enabling* or *refusing* certain events or sequences of events. The standard CSP semantics reflects this by modelling a process behaviour as a *failure* pair consisting of a sequence of events performed by the process (the *trace*) and a set of events that were subsequently offered by the environment but refused by the process (the *refusal*). A process is modelled by the collection of failures exhibited by the process. The standard Timed CSP semantics enhances this failures semantics by recording the timing of each event in the trace and also recording a refusal set for every point in time. This powerful modelling is able to describe an array of real-time concepts such as delays, timeouts, and clock interrupts.

2.3. Timed Communicating Object Z

Timed CSP and Object-Z complement each other in their expressiveness. Object-Z has strong data and algorithm modeling capabilities. The Z mathematical toolkit is extended with object oriented structuring techniques. Timed CSP has strong process control modeling capabilities. The multi-threading and synchronisation primitives of CSP are extended with timing primitives. The approach taken in the Timed Communicating Object Z (TCOZ) [9, 8] is to identify operation schemas (both syntactically and semantically) with (terminating) CSP processes that perform only state update events; to identify active classes [2] with non-terminating CSP processes; and to allow arbitrary (channel-based) communications interfaces between objects [6].

The syntactic implications of this approach is that the basic structure of a TCOZ document is the same as for Object-Z. A document consists of a sequence of definitions, including type and constant definitions in the usual Z style. TCOZ varies from Object-Z in the structure of class definitions, which may include CSP channel and processes definitions. In fact, all operation definitions in TCOZ are considered to define CSP processes. The CSP view of an operation schema is that it describes all the sequences of update events which change the system state as required by the schema predicate. The exact nature and granularity of these update events is left undetermined in TCOZ (at least at the syntactic level), but by allowing an operation to consist of a number of events, it becomes feasible to specify its temporal properties when describing the operation. Since operation schemas take on the syntactic role of CSP processes, they may be combined with other schemas and even CSP processes using the standard CSP process operators. Thus it becomes possible to represent true multi-threaded computation even at the operation level, something that would not be possible with other approaches. In addition to the inherited CSP's channel-based communication mechanism, in which messages represent discrete synchronisations between processes, TCOZ is extended with continuous-function interface mechanisms inspired by process control theory, the sensors and the actuators [7]. Those communication interfaces (channels, sensors and actuators) are given an independent, first class role in TCOZ. This allows the communications and control topology of a network of objects to be designed orthogonally to their class structure and reduces the need to reference other classes in class definitions, thereby enhancing the modularity of system specifications.

2.4. UML and Linking with TCOZ

UML mainly includes various graphical notations which can capture the static system structure (class diagram), system component behaviour (statechart diagram), system component interaction (collaboration and sequence diagram). The shortcomings of UML is that there is no unified formal semantics for all those diagrams, therefore, the consistency between diagrams is problematic. If UML is combined with formal specification techniques, then its power can be further realised and enhanced. We believe that the best companions for UML are likely to be formal object-oriented methods. We have developed the transformation techniques and the XML web enviornment for projecting TCOZ models to UML diagrams.

References

[1] K. Araki, A. Galloway, and K. Taguchi, editors. *IFM'99: Integrated Formal Methods, York, UK*. Springer-Verlag, June 1999.

[2] J.S. Dong and B. Mahony. Active Objects in TCOZ. *the 2nd IEEE International Conference on Formal Engineering Methods (ICFEM'98)*, pages 16–25. IEEE Press, Dec 1998.

[3] R. Duke and G. Rose. *Formal Object Oriented Specification Using Object-Z*. Macmillan, March 2000.

[4] W. Grieskamp, T. Santen, and B. Stoddart, editors. *IFM'00: Integrated Formal Methods,*, Lect. Notes in Comput. Sci. Springer-Verlag, October 2000.

[5] C.A.R. Hoare and J. He. *Unifying Theories of Programming*. Prentice-Hall, 1998.

[6] B. Mahony and J.S. Dong. Overview of the semantics of TCOZ. In Araki et al. [1], pages 66–85.

[7] B. Mahony and J.S. Dong. Sensors and Actuators in TCOZ. *FM'99: World Congress on Formal Methods*, *Lect. Notes in Comput. Sci.*, pages 1166-1185, Toulouse, France, Sep 1999. Springer-Verlag.

[8] B. Mahony and J.S. Dong. Timed Communicating Object Z. *IEEE Transactions on Software Engineering*, 26(2):150–177, February 2000.

[9] B. P. Mahony and J.S. Dong. Blending Object-Z and Timed CSP: An introduction to TCOZ. *The 20th International Conference on Software Engineering (ICSE'98)*, pages 95–104, Kyoto, Japan, April 1998. IEEE Press.

[10] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Languauge Reference Manual*. Addison-Wesley, 1999.

[11] S. Schneider and J. Davies. A brief history of Timed CSP. *Theoretical Computer Science*, 138, 1995.

[12] G. Smith. *The Object-Z Specification Language*. Kluwer Academic Publishers, 2000.

[13] J. Sun, J. S. Dong, J. Liu, and H. Wang. Z Family on the Web with their UML Pictures. http://nt-appn.comp.nus.edu.sg/fm/zml, 2000.