The Ptolemy II Framework for Visual Languages

Xiaojun Liu, Yuhong Xiong, and Edward A. Lee
EECS Dept. University of California, Berkeley
{liuxj, yuhong, eal}@eecs.berkeley.edu

Abstract

*This paper presents the Ptolemy II framework that supports the visual modeling and design of heterogeneous systems. Models in this framework are structured as hierarchical clustered graphs. The framework provides the infrastructure to support the implementation of a variety of models of computation as domains. Heterogeneous systems are modeled by hierarchically combining different domains. We describe the implementation of the *charts formalism (a generalization of Statecharts) in Ptolemy II as an illustration of the flexibility of this approach.*

1. Introduction

Component-based design is established as an important approach to designing complex embedded systems, which often have many concurrent computational activities and mix widely differing operations. Such systems are often modeled and designed using a variety of models of computation. Visual languages, with their ability to explicitly represent the concurrent activities and distributed structure of the target systems, can greatly help the designers to understand and reason about system behaviors. To apply visual languages in the design of heterogeneous systems, it is crucial for these languages to support multiple models of computation in a structured way.

An important class of visual languages that supports heterogeneous modeling is pioneered by Statecharts, proposed by Harel [3]. The *charts formalism (pronounced "starcharts") [2] generalizes Statecharts, and allows arbitrary hierarchical nesting of finite state machines (FSMs) with a variety of models of computation.

This paper presents Ptolemy II [1], a visual design environment that supports the modeling and design of complex heterogeneous systems. This environment is based on an abstract syntax of clustered graphs. A variety of models of computation, which define the semantics of the graph, are implemented as Ptolemy II *domains*. This framework for visual languages, and the implementation of the *charts formalism, are discussed in the following sections.

2. The Ptolemy II framework

2.1. Abstract syntax

In Ptolemy II, system models are represented as clustered graphs of *entities* and *relations*. The abstract syntax for such clustered graphs is illustrated in figure 1. E1 is a *composite entity.* E2, E3 and E4 are *component entities*, which are at the bottom of the hierarchy. A composite entity may contain other composite entities, so the hierarchy can be arbitrarily nested. Entities have *ports*, shown as filled circles. Relations, shown as diamonds, connect the ports.

2.2. Executable entities

The executable entities in Ptolemy II are called *actors*. They encapsulate computation and communicate with one another via some form of message passing. In addition, actors have an interface that allows fine-grained control of their execution. Like entities, actors can be composite.

2.3. Domains and message passing

In Ptolemy II, a channel of communication is implemented by an object called a *receiver.* All the receivers share a basic interface, but may implement different communication protocols, such as synchronous or asynchronous message passing. These different protocols, together with the execution control mechanism, define a set of Ptolemy II domains. Each domain implements a model of computation. For example:

- In the **continuous time** (CT) domain, actors represent components that interact via continuous-time signals. This domain is good for modeling systems with continuous dynamics.

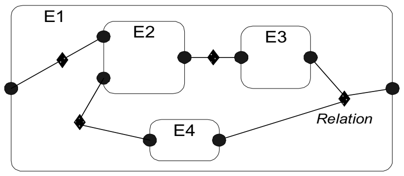

Figure 1. An illustration of the abstract syntax.

- In the **discrete event** (DE) domain, actors communicate via events placed on a real time line. Each event has a value and a time stamp. This domain is good for modeling digital systems.
- In the **synchronous dataflow** (SDF) domain, actors perform regular computations on data streams. The execution order of actors is statically defined. A valuable property of SDF models is that deadlock and boundedness are decidable. This domain is good for signal processing applications.
- In the **synchronous reactive** (SR) domain, the connections represent signals whose values are aligned with global clock ticks. The actors represent relations between input and output signals at each tick, and are usually partial functions with certain technical restrictions to ensure determinacy. This domain is good for applications with concurrent and complex control logic.

When heterogeneous systems are modeled in Ptolemy II, multiple models of computation are hierarchically combined.

3. *charts

The *charts formalism allows nesting FSMs with a variety of models of computation at any level in a heterogeneous model. For example, a state in an FSM can be refined to a DE model and a DE actor can contain an FSM.

3.1. Domain-polymorphic FSMActor

The *charts formalism is implemented in the FSM domain in Ptolemy II. The key component of this domain is FSMActor, which contains an FSM model. This actor can be embedded in another domain. Figure 2 shows an FSM actor that performs run-length coding in a DE model. These models are built in the Ptolemy II user interface environment called Vergil. The same FSM actor will also work in other domains.

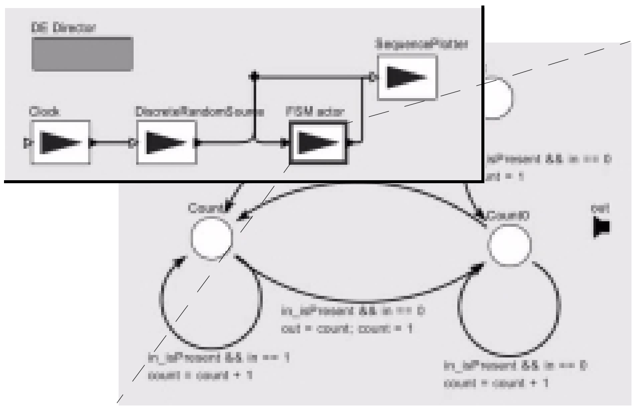

Figure 2. An FSM actor that performs run-length coding.

3.2. Modal systems

Many real-life systems exhibit multiple modes of operation. We use the term *modal* to describe such systems and their models. Using the FSM domain, we can build modal models in a natural way. After identifying the modes of operation, a model for each mode is built, possibly in different domains. Then a mode controller (an FSM encoding the mode control logic) is constructed; its states represent the modes of operation.

4. Conclusion

The Ptolemy II approach for visual language is based on a general abstract syntax, with the semantics given by a variety of models of computation. This approach supports the *charts formalism in a straightforward way.

We have developed a formal framework to study the communication protocols implemented by the receivers in Ptolemy II domains. More information about this framework and the Ptolemy II project in general can be found in the Ptolemy II web site [4].

Acknowledgement

This research is part of the Ptolemy project, which is supported by the Defense Advanced Research Projects Agency (DARPA), the MARCO/DARPA Gigascale Silicon Research Center (GSRC), the State of California MICRO program, and the following companies: Agilent Technologies, Cadence Design Systems, Hitachi, and Philips.

We thank Joern Janneck for many helpful discussions during the writing of this paper. We also thank the whole Ptolemy team for building the wonderful Ptolemy II software.

References

[1] J. Davis II, C. Hylands, B. Kienhuis, E.A. Lee, J. Liu, X. Liu, L. Muliadi, S. Neuendorffer, J. Tsay, B. Vogel, and Y. Xiong, "Heterogeneous Concurrent Modeling and Design in Java," Technical Memorandum UCB/ERL M01/12, EECS, University of California, Berkeley, March 15, 2001. (http://ptolemy.eecs.berkeley.edu/publications/papers/01/HMAD/)

[2] A. Girault, B. Lee, and E. A. Lee, "Hierarchical Finite State Machines with Multiple Concurrency Models," *IEEE Transactions On Computer-aided Design Of Integrated Circuits And Systems*, Vol. 18, No. 6, June 1999.

[3] D. Harel, "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming*, vol. 8, no. 3, p. 231-274, June 1987.

[4] The Ptolemy II Project, http://ptolemy.eecs.berkeley.edu/ptolemyII/.