# Integrating the Synchronous Dataflow Model with UML

Peter Green
Department of Electrical Engineering and Electronics, UMIST, Manchester, UK.
peter.n.green@co.umist.ac.uk

Salah Essa
Department of Computation, UMIST, Manchester, UK.
salahshaban3@hotmail.com

## Abstract

*UML has attracted significant interest as a system description language. However, some aspects of embedded system behavior are difficult to model in UML. In particular, applications with significant dataflow components are not well represented. This paper considers how the synchronous dataflow model can be integrated with UML to provide behavioral descriptions, in an object oriented context, for system elements that perform stream processing. The integration of the SDF model with the UML state machine model is also discussed.*

## 1. Introduction

Since UML has become the standard modeling language for object oriented (OO) software, there has been a significant interest in applying it in the broader context of the codesign of embedded systems [1, 2, 3]. This work is part of a project concerned with establishing a homogeneous development method, based on UML, for all aspects of embedded systems: software and hardware, application and platform. UML provides facilities for modeling system structure, and constructs for representing control-oriented behavior, but dataflow modeling (used for DSP and video systems) is not well supported. Hence, we indicate how dataflow modeling can be integrated into UML so that dataflow graphs can be used to describe the behavior of classes and objects. We focus on synchronous dataflow (SDF) [4], introduce a data-oriented communication mechanism to support the new behavioral description, and then consider how both state-based and dataflow behavioral descriptions can be combined in the context of a single class. This latter issue is related to other work that has sought to integrate different models of computation. In particular, it draws on [5] and [6].

## 2. UML and Dataflow

The context of this work is our development method, HASoC (Hardware and Software Objects on Chip), which is discussed in detail in [3]. HASoC notation is based on UML-RT (real-time), for reasons discussed in [3]. UML-RT extends UML via the specialization of a number of concepts. It emphasizes a hierarchical system organization via *capsules*, a form of active (concurrent) class, with specific execution and communication semantics. A capsule typically contains either sub-capsules, a single state machine, or both. Capsules communicate via owned objects (called *ports*) over c*onnectors* which show communications paths. The execution model is based on sending and receiving UML signals through ports, resulting in transitions in capsule state machines. Computation and communication can be associated with transitions, state entry, exit and occupancy.

The capsule model can be generalized to support alternative forms of behavioral description. Hence we introduce the notion of a *shell*, which is an active, abstract class with no concrete form of behavioral description, and which communicates through public ports. Shells form the basis of an embedded systems profile that we are developing. In this context, the features of this profile of direct relevance are *CShells* (control-oriented shells), *DShells* (data-oriented shells), and *HShells* (hybrid shells). CShells are equivalent to capsules, having state-based behavioral descriptions and signal-based ports (CPorts). DShells have a dataflow-oriented description and stream-based ports (DPorts) - see below. HShells (hybrid shells) have both control and data-oriented behavior and both CPorts and DPorts. Shell notation is shown in Figure 1.

We assume that the SDF model is used to represent dataflow behavior in DShells and HShells. In order to integrate SDFGs (SDF graphs) into UML, we must address the following issues:

*Communication and concurrency:* The SDF model involves communication via token sequences, produced and consumed by the firing of actors. This is not easy to represent in UML, without biasing the model towards a software implementation. Hence we introduce a new communications mechanism into UML: the *stream*. This models token-based communications at a high level of abstraction. The semantics of streams are discussed in [7].

In general, an object whose behavior is described by

1530-1591/04 $20.00 (c) 2004 IEEE

an SDFG will be continuously receiving and emitting token streams. Hence such objects will typically be active. *Class and Object Relationships:* A key aspect of OO development is the description of a system in terms of classes. If the internal behaviors of classes are described via SDFGs then the meaning of class relationships, such as inheritance, association, aggregation etc must be defined. [7] discusses these relationships for DShells. By drawing an analogy with inheritance in state machines, it can be shown that the relationship between SDFGs in base and derived classes is defined by the concept of actor clustering [4]. Based on [4], a relationship between the schedules in the base and derived classes can be obtained [7]. [7] also shows how a composition relationship amongst DShells can lead to the synthesis of a single SDFG, representing the overall behavior of the composite, and indicates that the schedule for the composite can again be determined via the results on clustering from [4].

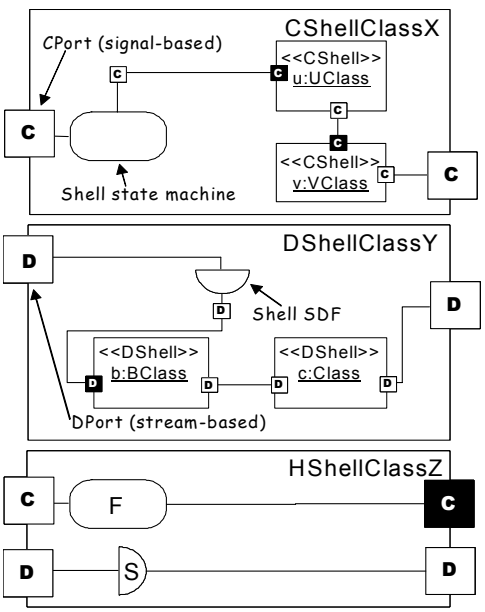


**Figure 1 CShell, DShell, and HShell notation**

*Integration with the State Machine Model:* In the most general case, represented by HShells, a class will exhibit both state-based and dataflow-oriented behavior, and so we must consider how the two aspects are linked within a class. In our model, SDFGs are represented as in-state 'do-activities' [7]. UML actions to support the initialization, execution, resetting, pausing and resumption of an SDFG upon state entry/exit have been defined in [7]. These enable a SDFG to be controlled without the need to modify the SDF model in any way at all. The operations simply achieve their effect by removing or restoring tokens, relying on token presence/absence to control the SDF operation.

The above considers the control of an SDFG by a state machine. An SDFG influences the behavior of its state machine via UML *change events*. Change events allow transitions to be triggered when a Boolean expression attached to the transition becomes true, and are generated by the actors in an SDFG. The integration of SDFGs and state-machines is further discussed in [7].

## 3. Discussion and Conclusions

This work aims to utilize OO methods for the modeling and design of embedded system software and hardware, using specialized behavioral descriptions for class/object behavior. The work is in progress, and much remains to be done. However, the SDF model can be integrated into an OO context, and for DShells, SDFG behavioral descriptions can be combined via standard OO relationships. This means that OO techniques can be used for high level modeling/design, and then behavioral descriptions of a processing chain can be aggregated and synthesized into a low (RTL/assembly) level implementation. It is also possible for dataflow and state machine models to be integrated within an OO context, and state machines and SDFGs can interact in a manner that is transparent to the SDF model and natural in the context of state machines.

The next stages of the work are to consider more complex interactions between state machines and SDFGs, e.g. data-dependent decision-making. It is also necessary to investigate how the composition and synthesis of behaviors that is possible with DShells can be achieved with HShells. The impact of the proposed extensions on the UML meta-model must also be assessed.

## 4. References


[1] Ferandes, J.M., Machado, R. and Santos, H. Modeling Industrial Embedded Systems with UML. In Proceedings of CODES 2000, ACM Press, pp18-23.

[2] Martin, G., Lavagno, L, and Louis-Guerin, J. Embedded UML: a Merger of Real-Time UML and Co-design. In Proceedings of CODES 2001, ACM Press, pp23-28.

[3] Green P.N., Edwards M.D. and Essa S. HASOC – Towards a New Method for System-on-a-Chip Development, Volume 6(4), Design Automation for Embedded Systems, July 2002.

[4] Bhattacharyya, S.S., Murthy, P.K., and Lee, E.A. Software Synthesis from Dataflow Graphs, Kluwer, 1996.

[5] Balarin, F., et al, Hardware-Software Co-design of Embedded Systems: The POLIS Approach, Kluwer1997.

[6] Girault, A., Lee, B., and Lee, E.A. Hierarchical Finite State Machines with Multiple Concurrency Models, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 18(6), June 1999.

[7] Essa, S. Object-Oriented Technology for System-Level Design. PhD Thesis, Dept. of Computation, UMIST, 2003.