

# Architecture Driven Development for Cyber Physical Systems

Gopal Raghav and Swaminathan Gopalswamy  
Emmeskay, Inc.

Copyright © 2009 SAE International

## ABSTRACT

Cyber-physical systems consisting of networks of interacting systems are often developed by distributed teams in a production environment. Processes, tools and work products supporting development of cyber-physical systems are continuously evolving through the different design phases. A growing trend to manage the development process has been the use of model-based development approaches. However, these approaches primarily use behavioral models to represent complex systems, rendering them inadequate to address collaborative and non-functional program requirements. This paper discusses an architecture-driven process that can address the challenges posed during the development of cyber-physical systems. Two key enabling technologies – the SAE AADL (Architecture Analysis and Design Language) and the IME (Integrated Modeling Environment) are leveraged in this process.

## 1. INTRODUCTION

Cyber-physical systems (CPS) are systems that have computational and physical elements, with tight coordination between the two groups of elements. The initial cyber-physical systems essentially had one (potentially complex) physical system interacting with one (potentially complex) cyber system. An example would be the internal combustion engine and the Electronic Control Unit (ECU) controlling the engine. The development of such systems has already proved quite challenging – leading to the adoption of Model Based Development (MBD) approaches (e.g. [1]). MBD involves capturing the behavior of the physical system as well as the cyber system as mathematical models that can be used for analysis, optimization, and verification and validation against desired functionality. However, there has been a dramatic increase in the complexity of cyber-

physical systems in two major ways that render traditional MBD inadequate:

**SYSTEMS OF INTERACTING CYBER-PHYSICAL SYSTEMS** - The landscape of the cyber-physical systems are increasingly characterized by large systems of cyber-physical systems interacting with each other in myriad ways, essentially exploding the complexity of behavior. For example, the ECU and the engine have to work in close coordination with the vehicle chassis and chassis control unit, the brakes and the vehicle with the anti-lock braking system, etc.

Another example would be a swarm of unmanned aerial vehicles (UAVs) working together to achieve a common mission. The individual UAV is itself a cyber-physical system of sufficient complexity whose development requires attention. However the swarm is a complex system where the individual UAVs are communicating through a cyber system while interacting through the physical medium in which they are performing.

The modern aircraft is another example of such a complex cyber-physical system with several ECUs that each control specific subsystems of the aircraft, but work together towards the flight missions.

**DEVELOPMENT PROCESS COMPLEXITY** - A growing trend in model-based development of cyber-physical systems is that development has moved from being practiced by single or small group of engineers working on R&D environment to large and globally distributed groups working in production environment. The processes, tools and work products continuously evolve during the development of the cyber-physical system. Integration of different aspects and parts of a work product worked on by different experts becomes a

challenge. Several variants of model-based approaches are today advocated to facilitate systems integration [2].

MBD is characterized by tools that focus on authoring, analyzing and simulating behavioral models. While these are essential for development of cyber-physical systems, they prove incapable of addressing larger needs arising from the two trends listed above. In particular, the needs related to analyzing a complex system without the need for specifying the exact behavior, as well as the needs related to variation management, are not addressed explicitly by MBD alone. This has led to the evolution of the development process from MBD to Architecture Driven Development (ADD). ADD offers a framework for development of large scale (systems of) cyber-physical systems, that leverages MBD as well as other related techniques.

In this paper we discuss the challenges faced during the development of large scale cyber-physical systems. We articulate an Architecture Driven Development approach for cyber-physical systems, along with two key enabling technologies that we have leveraged to support the ADD approach. Section 2 elaborates on the challenges faced in the areas of model management, knowledge management, variants management and distributed development. We also discuss the non-functional analysis needs that come from presence of systems of systems that is not addressed by traditional MBD. In Section 3 we layout the architecture driven approach that we believe addresses challenges described in section 2. Section 4 describes the two enabling technologies that support ADD, namely AADL and IME and finally conclude in section 5.

## 2. CHALLENGES TO DEVELOPMENT OF MODERN CYBER-PHYSICAL SYSTEMS

The challenges to development processes for cyber-physical systems arise from two key trends:

**SYSTEM OF SYSTEMS** - Modern cyber-physical systems consist of multiple systems interacting through each other through a variety of mechanisms. For example figure 1 shows a set of distinct cyber-physical systems each consisting of its own physical system P and cyber system C. The distinct physical systems are themselves coupled to each other through physical interactions, while the distinct cyber systems are coupled to each other through communication methods such as shared memory access or some protocol based communication (CAN, LIN, TCP/IP, etc). The nature of the communications and the topology of the cyber systems impact the overall performance of the system of systems.

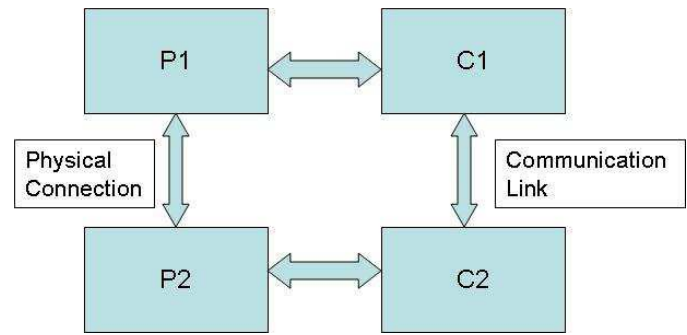


**Figure 1: Interacting cyber-physical systems**

In a traditional approach, to study the impact of the communications and topology, the system of system model is constructed from models of the individual systems by adding models for the communications. Clearly there are two major issues with this approach:

- The study of the impact of the topology and communications within the system of systems has to wait until the development of the (models of the) individual systems is completed. This in turn would introduce significant delays in the development time and cost for very large and complex system of systems.
- The complexity of the resultant systems can be so overwhelming that even if they can be analyzed, the interpretation of the analysis results in a meaningful way becomes tedious and error-prone to the practicing engineer, rendering the entire analysis relatively useless.

The current approaches to address the above difficulty have essentially adopted the approach of designing the individual cyber-physical systems as best as can be done, and keeping the communications delay as small as possible; but this essentially postpones the identification of the problems to verification and validation experiments conducted after the entire system is completed.

**VARIANT MANAGEMENT** - Many modern cyber-physical systems are large in scope, and involve large and globally distributed teams. The work products and the tools to support the work change throughout the life-cycle of a cyber-physical system. While it is clear that MBD is a necessary ingredient in any development solution for cyber-physical systems, the management of the changing work product is a significant challenge that needs to be addressed. The sources of the change are manifold [3].

One type of change is change due to evolutionary development – here the dimension of change is time. Work products (models, data, scripts, etc) need to be version controlled. Today there are many technologies available for basic version control. However, the work products are intricately dependent on each other. These dependencies are not natively captured by traditional version control technologies. (Some temporary solutions have been to create meta-data on top of these work