

# Models make sense

How model based design can ease system architecture problems in a range of applications . By **Graham Pitcher**.

Software is becoming an increasingly important part of any system. Depending who you talk to, the software development task can represent up to 70% of a project's cost. Not only is software becoming more important, it's becoming more complex – and the automotive sector is a case in point.

Electronic control units (ecus) are just one example of an application where complexity is growing: in some instances, functionality can be distributed across several ecus. So what is a system designer to do when faced with the prospect of more – and more complex – software?

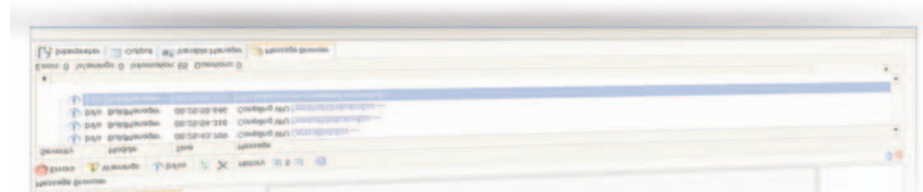
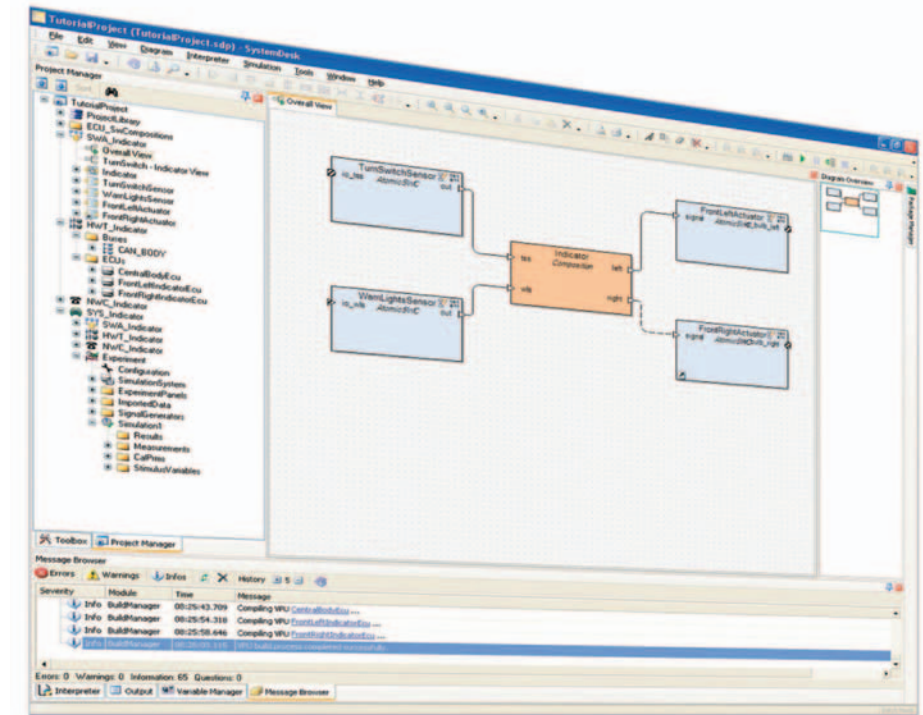
One possible solution is to turn to model based design. According to software developer dSpace, model based development of ecu software and the graphical representation of the components and their relationships improves the user's understanding of complex systems.

The company believes ecu complexity is a challenge for automobile manufacturers and their suppliers. The use of formal system models can therefore ease the communications problems between all those involved.

## Design complexity

But the problem is often made more complex by design being distributed amongst different teams – and these can be on different sites and even in different companies. For example, an oem may develop the overall system design, but require a supplier to develop a part of this. By allowing parts of the design to be exported, each developer will have the same information about the system and will work to satisfy the same requirements. According to dSpace, this can be crucial to integration later in the project.

dSpace's solution is SystemDesk, a system architecture software that allows its users to plan, implement and integrate complex system architectures more quickly than before.



SystemDesk is claimed to bring a number of benefits, including:

- \* Design of functional networks and software architectures
- \* Modelling systems according to the AUTOSAR standard
- \* Formalising hardware topologies and network communication
- \* Integrating ecu code
- \* Simulating a single software component or an entire network.

At the start of the development project, the system architecture is defined, including the software components, their interfaces and ports. The hardware topology is also described. Developers can then use this system template to

develop software components. Based on the specification, these modules can be refined as the development process continues. At this stage, the system has been defined as one software architecture, one hardware topology and one communication network.

Using SystemDesk, software architectures can be modelled graphically and can include the components and their interfaces. A component diagram is used to perform this step, providing an insight of the software components. Software components are brought together using the composition diagram, which provides the user with an overview of the system as a whole.

Software components can be further defined

**Examine the priorities, says Anuje Apte.**

Optimising system resource utilisation is a key design objective for system engineers in the electronics and other industries. System resources – such as processors, memory, or bandwidth on a communication bus – are often shared by various components in the system. To understand how the resource is shared, system engineers must identify constraints on the resource, such as number of processors and memory size, and analyse the effect of input traffic or load on the shared resource.

An example is prioritised task execution in an real time operating system (rtos). Typical tasks include low priority application tasks in the rtos' main loop and high priority interrupt tasks that invoke interrupt service routines (ISRs). Execution of interrupt service routine delays the processing of application tasks, since the application tasks wait in the task queue while the ISR is executing.

Ideally, the number of tasks waiting in the task queue should not increase and the rtos should return to the idle task to avoid excessive processing delays. How can we verify that the rtos satisfies this requirement?

To answer this question, we must model the shared processor and its interaction with application tasks and interrupts. Specifically, the model should demonstrate priority based task execution and preemption. Figure 1 shows the SimEvents model of such a system.

The Task Token Generators section contains subsystems that generate high priority tokens for interrupts and low priority tokens for application tasks. The tokens carry an attribute (TaskExecTime) that indicates the execution time for that task. The Task Token Manager section contains the task queue and the processor shared by the input tasks. The Priority Based Task Queue sorts the current queued tasks according to priority. Interrupt task tokens entering this queue preempt the application task token being served in the processor. Upon preemption, the application task returns to the Priority Based Queue. This task resumes its processing when the interrupt service routine is complete. The dashed function call signal lines are used to generate tasks whose execution depends on the completion of another task.

This model can be simulated and a plot generated showing how the number of tasks in the task queue changes during the simulation time.

If the plot indicates that the number of tasks waiting to be processed increases with time and that the task queue does not return to an empty state, it would imply that the processor is not fast enough to support the high input rate of tasks.

Since the input rate of tasks cannot be controlled, the next steps might include exploring the use of a high speed processor to reduce the number of tasks waiting in the task queue.

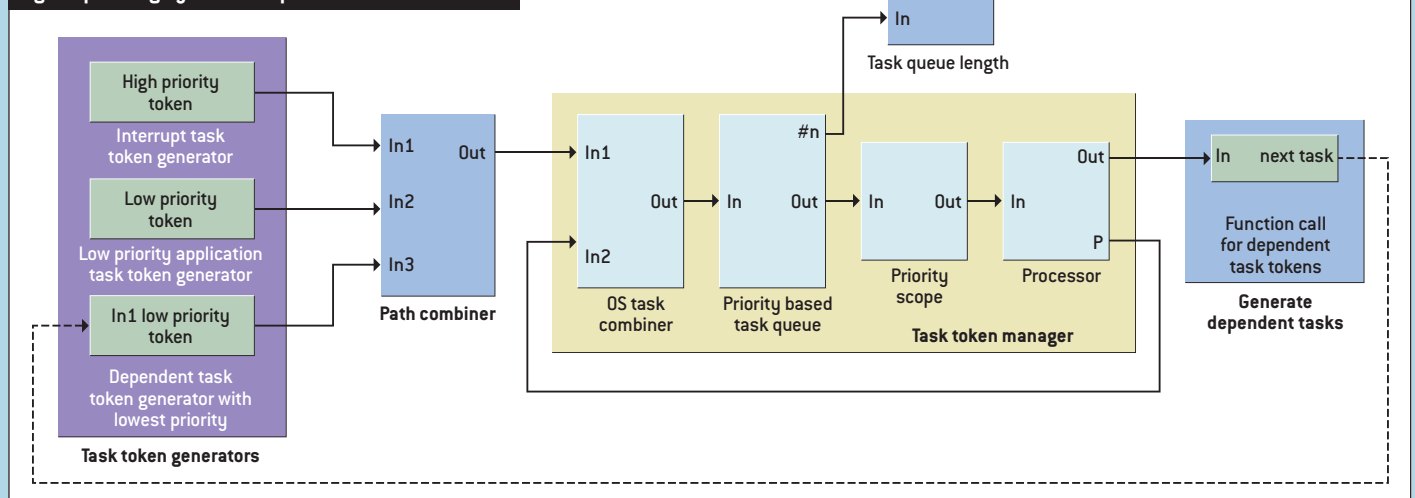
Using SimEvents, designers can:

- Identify constraints on shared resources and bottlenecks in a system
- Simulate the effects of variable input traffic, such as congestion, packet loss, and increased end to end latencies, and
- Explore different design techniques and algorithms to optimise resource allocation.

**Author profile:**

Anuje Apte is a product marketing manager with The Mathworks.

**Fig 1: Operating system with prioritised task execution**



by inclusion of such features as C functions that can be called by an operating system. However, the component itself can be developed outside of SystemDesk using a behaviour modelling tool. The component's description can be exported and this can be used to provide a frame model. The developer can then build the necessary control algorithms.

Once the behaviour of the software

component has been defined, the system can generate C, along with an extended software component description. The component can then be imported into SystemDesk, where it is checked for consistency and made available for further integration.

SystemDesk also features a simulation module that allows offline simulation to be performed on a pc. Single software components

can be simulated in non real time using the original C code, and the interaction of several components can also be verified. Open and closed loop simulations are supported and, according to dSpace, this allows both modelling and simulation to be performed in one environment. With verification taking place earlier in the development phase, both time and cost can be saved.