2006-01-1242

Using Model Driven Architecture for the Development and Integration of Platform-Independent Vehicle Application Software across Different OEMs

Brendan Jackman
Waterford Institute of Technology, Ireland.

Shepherd Sanyanga
Ford (Europe), United Kingdom.

ABSTRACT

This paper proposes a solution to the challenge of developing vehicle software application functions which are decoupled from their intended target hardware platforms. Once developed, these software application functions can be utilised across any OEM vehicle platform and vehicle variants, saving the supplier time and money in terms of system development and giving a number of OEMs similar tried-and-tested system application software.

The proposed solution is to use the Model Driven Architecture (MDA)[1], a UML-based development approach that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform. MDA allows a vehicle function to be modelled in a semantically rich UML [1,2] model which is completely independent of any implementation detail.

INTRODUCTION

The amount of software and electronics in vehicles is increasing at a steady rate in response to customer demands for increased safety, comfort and performance. Many new vehicle functions are implemented by integrating existing vehicle applications and by so-called sensor fusion, where data is shared on vehicle networks between many co-operating functions. Usually these software applications are developed by different suppliers, presenting a big integration problem together with associated costs for both the supplier and the OEM.

The burning question is: How can suppliers stay competitive in terms of system cost, resource utilisation and resource allocation when they are required to implement similar vehicle application functions across different OEMs with differing technology platforms and vehicle variants ?

MODEL-BASED DEVELOPMENT

Software engineering has seen many developments over the past few decades, from the first high-level languages such as FORTRAN and BASIC, to 4GLs and Frameworks, to the recent growth in the use of graphical development environments, object-based technologies and middleware platforms. The underlying trend in all of these developments has been the use of increasingly higher levels of abstraction to specify and develop software systems. The increasing complexity of software systems and the decreasing time-to-market require software engineers to look to more abstract development techniques to improve productivity and manage complexity. This is evident in the automotive software development domain, where software systems are usually developed by geographically-dispersed teams to operate in a complex electro-mechanical environment.

Automotive software engineers have been using model-based development techniques for some time to develop and test vehicle software functions. Model-based development is used primarily to design vehicle networks and control algorithms. Proprietary tools exist for configuring network communication modules based on message communication matrices and for optimizing software function allocation to Electronic Control Units (ECUs) based on models of application data flows and the physical network architecture. Hardware-in-the-Loop (HIL) simulation is widely used to verify control

[1] MDA, Model Driven Architecture, UML and Unified Modeling language are trademarks of the Object Management Group.

algorithms against models of the vehicle systems before the target system is generated, usually by some kind of auto code-generator. In such systems the control algorithms are specified with modeling tools such as Matlab/Simulink using dataflow and state diagram notations. The model can be functionally verified on a PC against models of the vehicle-driver environment. Only when the models are verified is the physical target system code generated from the model.

System development using models at a high level of abstraction offers the following benefits:

- Faster design iterations;
- A reduction in design, development and implementation costs;
- Earlier verification of designs with a resulting decrease in system integration problems;
- Better traceability from requirements to implementation, since the model is both the specification and the implementation.

Figure 1 illustrates the current use of models in the development process.

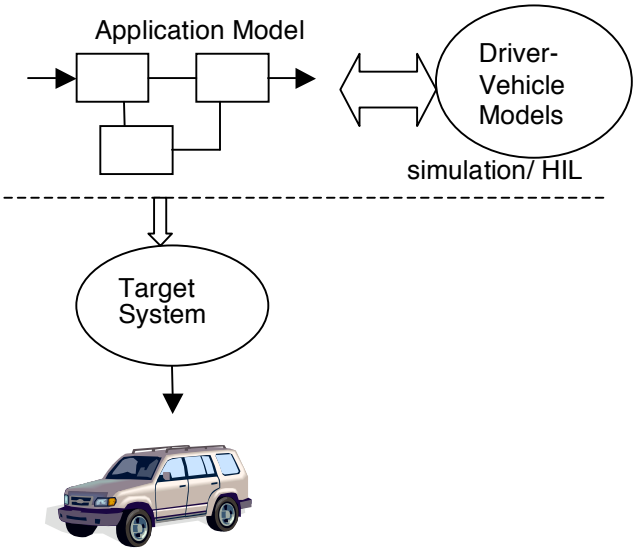

Figure 1. Models in the development process

MODEL DRIVEN ARCHITECTURE

Model Driven Architecture (MDA) is an initiative by the Object Management Group (OMG) [1] that takes model-based development to the next level. One of the problems with the models used in today's development tools is that even though they are at a relatively high level of abstraction they still contain much implementation detail. For example, software function models usually contain details of how sensors and actuators are interfaced, how signals are transmitted on vehicle networks, how functions are deployed to processors and so on. All of this embedded implementation detail makes it very difficult to port the models to different target platforms. Developers typically use modular design and encapsulation principles to limit the platform-specific details to a small portion of the model. The MDA solution to this problem is to divide an application into a set of models of different levels of abstraction. MDA defines the following model types, from the most abstract to the most concrete:

- Computation Independent Model (CIM);
- Platform Independent Model (PIM);
- Platform Specific Model (PSM);
- Code Model.

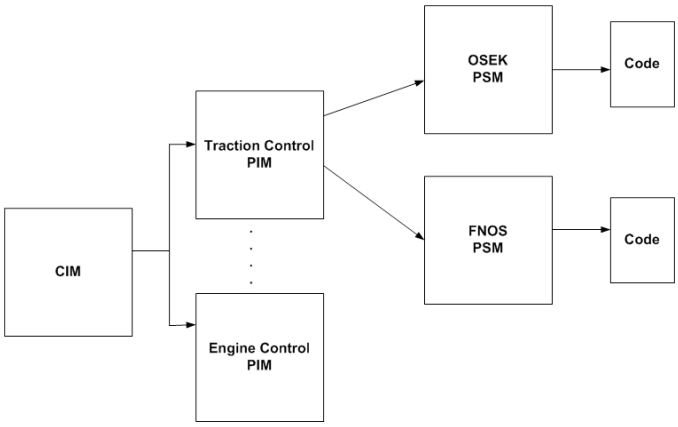

Figure 2. MDA models and their relationship

COMPUTATION INDEPENDENT MODEL

The CIM operates at the application domain level and describes the interactions between processes and elements in the application environment. The CIM is usually expressed in a domain-specific language and avoids specialized knowledge of procedures. An automotive example would be the use of differential equations and other mathematical means to describe control algorithms.

PLATFORM INDEPENDENT MODEL

The PIM describes the processes and structure of the system without detailing the implementation platform. It is thus independent of operating systems, networks,

programming languages and hardware. PIMs are usually expressed as Unified Modeling Language (UML) models. The PIM is the main focus of system development projects. Removing implementation-specific concerns from the PIM has the following advantages:

- It is easier to verify the functional correctness of a model which is not burdened by platform-specific semantics;
- Different systems can be integrated easier at the PIM level and the combined models subsequently mapped to a specific implementation platform;
- It is easier to target the PIM to different implementation platforms while ensuring that the overall system functionality remains identical.

It is important to note that MDA allows the system to be represented by a set of models at the PIM level. These models may be related in a hierarchical manner (by refinement) or they may not be directly related, for example when each model represents a different system *viewpoint*.

PLATFORM SPECIFIC MODEL

The PSM represents the platform-specific implementation of the system. The PSM is derived directly from the PIM using a transformation process which is explained in a later section. A separate PSM exists for each mapping of a PIM to a specific platform. MDA allows the PSM to consist of multiple models, representing different implementation concerns, such as

- Concurrency;
- Data storage formats;
- Time handling;
- Event handling;
- Exception handling.

The PSM can also be organized as a number of tiered models when appropriate, for example when separating the implementation concerns of different layers of a network protocol.

CODE MODEL

The code model is the final deployable object code that is executed on the target platform.

MODEL MAPPING

The aims of MDA can only be achieved if the platform-specific details are kept completely separate from the Platform Independent Models. While it is possible to map a PIM to a PSM manually, the real benefits of MDA are achieved only when the mapping process is automated. The Object Management Group has defined a number of supporting technologies that will eventually allow MDA to be automated.

- UML 2.0 provides a semantically rich modeling language for describing systems and software;
- UML Profiles are a means of extending the UML notation to model domain-specific concepts.
- MOF (MetaObject Facility) is a fundamental language for defining other modeling languages (metamodels). UML has been defined in terms of MOF;
- XMI (XML Metadata Interchange) is a specification for importing/exporting models between tools;
- CWM (Common Warehouse Metamodel) is a specification for storing models in a repository.

MDA tools are currently in their infancy, with some existing tool vendors providing limited support for some of the MDA concepts. In a fully automated MDA environment the mapping from abstract models to more specific models will be guided by a separate set of mapping models. This process is illustrated in Figure 3.

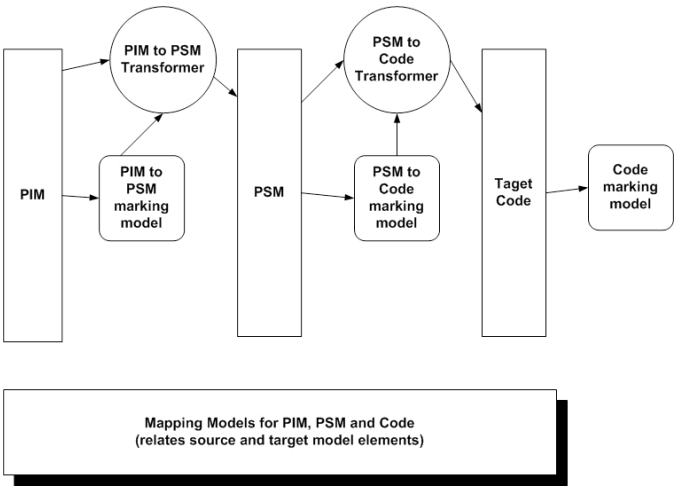

Figure 3. MDA Mapping Process

Even though the PIM is platform-independent, this is a relative term. The PIM must at some stage provide guidelines on how each functional concept is to be implemented. These design decisions are really related to the platform-specific concerns described in the PSM and are required as input to the mapping process. For example, a function in the PIM may be implemented on a platform in a number of ways; as a task, as an interrupt

service routine or as a re-entrant ROM routine. The concepts of Task, Interrupt and Routine would be described by the PSM model, but to select the appropriate implementation the PIM must be supplemented with *marks* to indicate the desired design decisions. The designer enhances the PIM with marks related to the PSM, but these marks are stored in a separate *Marking Model*. This keeps the PIM uncluttered by any implementation detail and allows different marking models to be applied when generating different PSMs for various target platforms.

The mapping process is driven by a Transformer that converts source models to target models based on the markings indicated in the marking model. A separate Mapping Model relates the source model elements to the generated target model elements so that there is full traceability between the models.

What makes the whole process very flexible and efficient is the fact that all of the models, namely the PIM, PSM, Marking and Mapping models, are expressed in UML or some MOF-derived language. Even the transformations from PIM to PSM can be expressed as models to provide the ultimate in flexibility.

USING MDA IN THE AUTOMOTIVE INDUSTRY

The ideas of MDA can be applied to software development in the automotive industry. Vehicle system functionality could be described using Platform Independent Models. These would be primarily UML models describing the behavior of the required software functions and the desired system structure. These would have to be rigorously defined models that could be executed for the purposes of functional verification.

Once the Platform Independent Model is verified it can then be marked for transformation to a Platform Specific Model such as an OSEK implementation. This would involve the creation of a marking model to indicate certain key design decisions related to OSEK implementation. One approach to modeling PSM concepts is to use a UML Profile for each target platform. These profiles would define stereotypes for key platform elements such as Task, Alarm etc., together with tagged values that are used to configure the element. The marking process then consists of labeling items in the PIM with the stereotypes and tagged values of the PSM UML Profile. As mentioned earlier, these markings can be stored in a related model to keep the PIM free from implementation detail. A partial example of using a UML profile is shown in Figure 4.

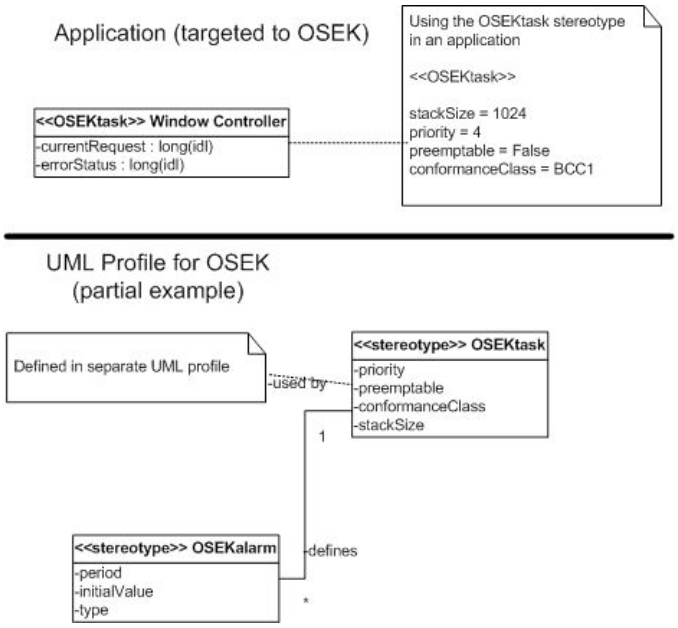

Figure 4. Example of using a UML Profile

The transformation process can then be guided by the markings on the PIM so as to generate a PSM that describes the implementation of the system on the specified target platform. Remember that the PSM is another UML model, but at a lower level of abstraction than the PIM. It has details of the design decisions required to implement the system on the target platform. The final step is to transform the PSM to executable code using a code generator.

While reverse engineering is not a goal of MDA, the ability to reverse engineer code changes back to both the PSM and PIM would be important in an automotive environment where the target code is usually debugged in a test vehicle away from any MDA tools. The mapping models take on an important role in keeping the models synchronized in such cases.

Automotive suppliers delivering systems to multiple OEMs can leverage MDA by defining their system functionality using a PIM and specifying a different PSM for each OEM target platform. That way the same functionality is delivered to each OEM platform with a clear separation of OEM platform issues in each PSM.

MODEL-DRIVEN INTEGRATION

The successful integration of software functions in a vehicle is still a challenge due to the tremendous variations in ECU platforms, networking protocols and network management strategies. System integration activities typically concentrate on integrating the various layers of the system in a bottom-up fashion. For example, CAN Physical Layer problems are sorted out first before network management and finally functional integration problems are addressed. By using an MDA approach functional integration can take place much

earlier in the development cycle, as soon as the PIMs for each separate function are available. Likewise the PSMs can be integrated at the PSM level to resolve any platform-specific integration issues. It is also possible to define mappings between different PSMs to abstract interfacing details.

MDA AND AUTOSAR

AUTOSAR [3] is an effort by OEMs and suppliers to standardize the electrical and electronic architectures of vehicles. It proposes to define a standard set of interfaces and functions to cover most vehicle applications. While the AUTOSAR specification has not yet been finalized, it is reasonable to assume that the main concepts and interfaces of AUTOSAR could be defined by a UML Profile. This would then pave the way for UML models of vehicle functions to be enhanced with AUTOSAR profile stereotypes and tagged values that could guide a transformation to PSM and executable target code. Suppliers would then be free to concentrate on core functionality rather than AUTOSAR implementation details. AUTOSAR is intended to execute on many hardware platforms so the PSM models would contain two tiers: an upper tier PSM specifying an AUTOSAR implementation, and a lower tier refining the model to a specific microprocessor. The widespread adoption of AUTOSAR would provide a great incentive to tool developers to incorporate MDA concepts, allowing different AUTOSAR platforms and configurations to be targeted in a platform-independent manner. Perhaps the greatest benefits to be had are from the early verification and integration of the Platform Independent Models in the development process.

In addition to AUTOSAR, PSMs could be created for other automotive platforms such as OSEK, FNOS etc., allowing suppliers to reach a global automotive market with a single set of proven software functions.

CONCLUSION

This paper presented an overview of the Model Driven Architecture concept and described some of the possible benefits to the automotive industry from adopting MDA. MDA is at an early stage of development and is presented as a vision rather than a process or set of tools. Many of today's software development tools exhibit some aspects of MDA, but none really take MDA to its ultimate level: the complete separation of platform-specific concerns from application models. MDA provides a set of robust core technology specifications and a vision that brings it all together. With the current software crisis in the automotive industry, now might be a good time for OEMs, suppliers and tool developers to work together towards realizing the MDA dream.

REFERENCES

1. www.omg.org. Model Driven Architecture.
2. UML 2.0 in a Nutshell. Dan Pilone. O'Reilly Media, 2005.
3. www,autosar.org. AUTOSAR consortium.

CONTACT

Brendan Jackman B.Sc. M.Tech.

Brendan is the founder and leader of the Automotive Control Group at Waterford Institute of Technology, where he supervises postgraduate students working on automotive software development, diagnostics and vehicle networking research.. Brendan also lectures in Automotive Software Development to undergraduates on the B.Sc. in Applied Computing Degree at Waterford Institute of Technology. Brendan has extensive experience in the implementation of real-time control systems, having worked previously with Digital Equipment Corporation, Ireland and Logica BV in The Netherlands.

Email: bjackman@wit.ie

Website: http://www.wit.ie/automotive

Shepherd Sanyanga (BEng BSc MSc PhD CEng EurIng MIEE)

Shepherd has worked for United Nations in Africa in infrastructure development programs, worked for some years in the Aerospace Industry developing military electronic systems and then worked for a number of Tier One Automotive Suppliers (Lucas Electronics (UK), Sagem (France) TRW (UK). He is currently involved in a joint project between Ford Motor Company (Europe), Johnson Controls (France) and Takosan (Turkey). In his spare time he is also an external examiner on a masters degree automotive programme in a university in Ireland.

Email: ssanyan2@ford.com