

Using Model Transformation/Code Generation Technology to Migrate Legacy Software Assets to AUTOSAR

2011-01-1264

Published
04/12/2011

Rick Flores, David Rosa and Mohan Murugesan
General Motors Company

Copyright © 2011 SAE International

doi:[10.4271/2011-01-1264](https://doi.org/10.4271/2011-01-1264)

ABSTRACT

Maximizing reuse of existing software assets is a common goal for companies as they move to new projects and products. Typically, reuse is complicated by changes to the context in which the software executes; the software architecture. As in all development areas, software architects are seeking continuous improvement to their designs either through internal innovation or established external standards. However, changes to the software architecture (e.g. interfaces, services, scheduling etc.) can prove to be a driver of fundamental, large-scale change to software if reuse strategies are not planned into the product line. This paper describes a reuse strategy that maximizes the preservation of existing assets while focusing effort on the development of transformation rules that generate software targeting a new architectural context.

INTRODUCTION

The Model Transformation/ Code Generation (MT/CG) technology, a method for defining transformation **rules** to convert a set of assets from an initial form to a desired end state, has matured in recent years. A tool supporting this technology has the ability to import one or more meta-models from various sources allowing the user to define mapping rules. Combining this ability with automatic code generation provides a powerful capability to convert a model compliant to known conventions to code targeted at a different software architecture forming the foundation of an approach of converting legacy models to a new environment such as AUTOSAR.

If the transformation rules are owned by the company, the ability to migrate assets is bounded by its rules authoring

engineering resources rather than relying on availability of functional area domain experts to re-write software. It is assumed the number of engineers to maintain a set of rules is much less than the number necessary to re-write software applications.

Where some existing strategies for software reuse are capable of 1-1 transformations, for example direct code generation from behavioral models, the authors have found that MT/CG technologies allows a higher-level of software reuse by enabling more nonlinear transformations of models-to-code. This can include generating additional layer(s) of structural code for binding to entirely new software architectures.

Internal convergence to a single software organization and motivated by the automotive software industry's move toward a standard architecture, AUTOSAR, the General Motors software architecture team has begun migration activities away from proprietary architectures. The purpose of this paper is to describe the experiences of General Motors in applying MT/CG technology to solve a problem of a large volume of legacy software running on a software architecture targeted for obsolescence.

MOTIVATION

The next generation of General Motors electrical architectures requires the ability to re-distribute functionality across ECU nodes based on varying needs of the GM vehicle program portfolio. For example, some programs may include separately packaged ECUs for control of body and display functionality while others may combine the functionality into one ECU.

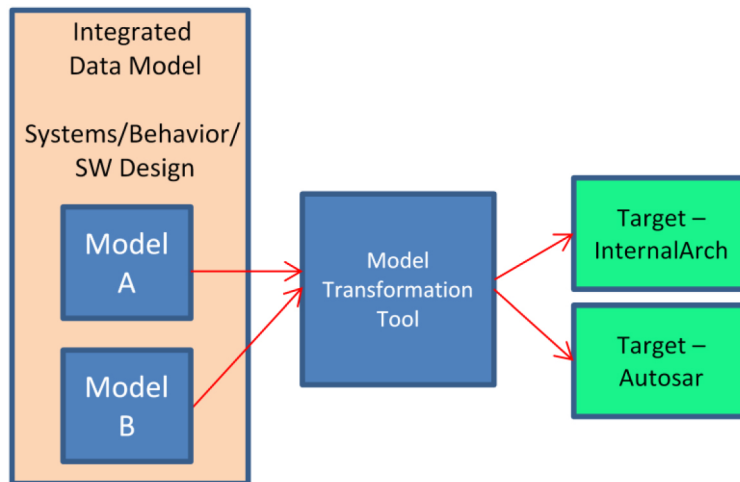


Figure 1. Targeting Distinct Architectures with Model Transformation Rules

This flexibility is realized by 1) running a common software platform on a set of integration nodes and 2) applications using a communication layer that abstracts them from knowledge of the source/target of their required/provided signal interfaces. This approach is characteristic of the AUTOSAR software architecture and enables the mixed IP (intellectual property) business model of software development.

From a software development perspective for deeply embedded controllers, General Motors currently has both externally sourced and internally developed parts. For internally developed ECUs, two internal legacy software architectures (referred to genetically as InternalArch1 and InternalArch2) have been targeted for migration toward the standard architecture, AUTOSAR. A path has been identified for migrating both infrastructure and content running on these legacy architectures.

In addition to InternalArch 1 and InternalArch2, there are other internal legacy software architectures and associated content which require migration toward the next generation. For these, GM must define a strategy to move toward one of InternalArch 1 / InternalArch2 architectures or move directly to AUTOSAR. This choice depends on how close/far they are to the existing architectures.

We will discuss a case study of domain application software written to run on an internal legacy software architecture *not* targeted for migration to AUTOSAR (referred to generically as LegacyArch). Although the LegacyArch aspires to the similar high-level goals of AUTOSAR, the mechanisms are different enough as to make the direct usage of the software on the standard architecture impossible. This lack of portability impedes reuse of the existing base of GM's intellectual property. The desire to retain the application content of the legacy software while moving to next

generation software architecture is the main motivation driving the definition of the migration strategy described in this paper.

CURRENT APPROACHES

The current generation of GM Software & Controls has used model-driven development extensively for algorithm and system development including behavioral, structural, and 'middleware' code generation.

A modeling convention based on UML and our own 'Domain-Specific Language' (DSL) extensions has been employed in our approach. The capability of adding extensions has been a key to allowing UML concepts to be merged with those of our own DSLs. The derivation of a domain-specific language from UML is through the concept of a UML Profile. A UML Profile provides the ability to document concepts of extending the language by creating additional behaviors and restricting the language by reducing behaviors.

The interfaces, behavior and the software design are captured in the context of the modeling convention. Once in this format, model transformation rules are applied to generate software conforming to a specific architecture. In this way, the direct dependence on architecture is eliminated enabling algorithm re-use across architectures.

MODEL TRANSFORMATION/CODE GENERATION TECHNOLOGY

BACKGROUND

The technology for transforming models of algorithms into code has existed for over a decade within the deeply embedded software space. Different modeling tool vendors have developed, marketed, and sold solutions that catered to

their modeling environment with widely varying degrees of success of actually delivering code that can be used in a target environment.

FROM CODE GENERATION TO MODEL TRANSFORMATION

Similar to the progression of web-based technologies such as Hyper Text Markup Language (HTML) and Extensible Markup Language (XML), the transition from code generation to model transformation is one of generalization.

Source code is nothing more than the representation of a model in text form that can be compiled, linked and programmed into a target environment and can therefore be classified as yet another model that has been transformed based on assumption about the target source code language and underlying software architecture on which the compiled code will run. Since the first applications of model transformation were focused on code generation, the information captured in model form represented nothing more than the algorithms to be transformed into source code. However, the benefits of modeling have rapidly grown beyond the scope of target electronic control unit (ECU) algorithms and are seeing increased use within other automotive design disciplines.

However, no useable single set of modeling conventions/guidelines has yet been developed that can adequately cover the needs of the full set of automotive design disciplines, and the need to exchange and correlate data between these disciplines is rising due to the exponential growth of complexity within automotive electrical architectures and the increasing scarcity of engineering resources.

The generalization of code generation technologies to model transformation and the application of such technologies are key to successfully moving the automotive design processes forward in order to cope with these future demands.

GM'S PATH TO MODEL TRANSFORMATION

Early Stages

- Algorithm modeling for requirements capture
- Algorithm modeling and code generation for simulation (in-vehicle and extra-vehicle)
- Algorithm modeling for test case generation (delivered to the vehicle validation group)

Recent Stages

- System, ECU and algorithm modeling and model transformation for target ECUs utilizing a single domain specific language (DSL)

- Modeling and code generation for ECU middleware
 - Tying algorithms to infrastructure/ECU services
 - Generating algorithm scheduling
 - Tying serial data handlers to algorithms
- Removing electrical architecture concerns from algorithm design
- Generating unit test environments for algorithms
- Initiated model driven collaboration between:
 - system and algorithm design disciplines
 - software design and ECU integration teams
 - algorithm design and ECU test teams

Future Stages

- Expanded model driven collaboration between the system, algorithm, software design, ECU integration and ECU test disciplines.
- Enhance to enable use of multiple domain specific languages (DSL) within the various engineering disciplines and use model transformation technologies to bridge between the varying DSLs.
- Move beyond application of the model transformation technologies to uni-directionally transform models into bi-directional applications of the technologies to keep models in varying forms in sync with each other.

Benefits Derived Thus Far

When combined with the software architecture agnostic modeling guidelines developed by the GM Software & Controls team, the model transformation/code generation technologies are allowing GM to take model assets originally developed to target the GM S&C proprietary InternalArch2 software architecture, and target the AUTOSAR software architecture with no impact to the algorithm models. All changes necessary to support targeting the AUTOSAR software architecture are absorbed by the model transformation/code generation tools.

While it would be nice to claim that this benefit is solely derived from using model transformation/code generation technologies, it is the fact that the GM S&C InternalArch2 software architecture closely resembles that of the AUTOSAR software architecture, combined with the GM S&C code generation / modeling transformation technologies that have enabled this migration.