

# Converting Legacy Embedded Control Software to Executable Specifications

Koichi Ueda  
TOYOTA MOTOR CORPORATION  
Yoshitaka Uematsu  
DENSO CORPORATION  
Michael Baloh  
EMMESKAY INC.

## **ABSTRACT**

This paper describes recent efforts of TOYOTA in Model-Based Development (MBD). In order to shift to MBD entirely, one important step is the translation of documented legacy control algorithms into executable specifications like Simulink® models. TOYOTA and DENSO have already deployed a proprietary Auto-Code Generation (ACG) environment for advanced and mass production development. By increasing the use of model-based executable specifications, the ACG technology is increasingly used in control and ECU software development. Furthermore, improvements in productivity are expected through a seamless environment that includes other tools such as rapid prototyping ECU, Hardware-In-the-Loop Simulation (HILS) and Software-In-the-Loop Simulation (SILS). Therefore, given the scope of the problem, TOYOTA has developed the conversion process with DENSO and EMMESKAY.

This paper focuses on some key aspects of the conversion process that are important to ensure efficient and reliable translation. It discusses the importance of a structured process, aids for model implementation, and the criteria to measure the resulting model quality.

## **1. INTRODUCTION**

In the automotive industry, control system has increasingly become more complex. On the other hand, the development period must be reduced. To resolve the issue, many organizations recognize the need to move toward Model-Based Development (MBD). It is important to use plant and controller models as executable specifications. Models implemented in executable specification languages like Simulink® and Stateflow® have many benefits. For example, control design by a closed loop simulation with plant models, auto-code generation and efficient verification/validation.

Figure 1 shows MBD process. While TOYOTA is developing new control algorithms through this process, TOYOTA still has a substantial control system knowledge base in the form of documented legacy specifications. These algorithms have been in production for a number of years. Hence, they are proven in the field and are well understood.

In order to increase the adoption of MBD, TOYOTA started a project with DENSO and EMMESKAY to convert legacy embedded control software into Simulink® models. The generic idea of the conversion is shown in Figure 2.

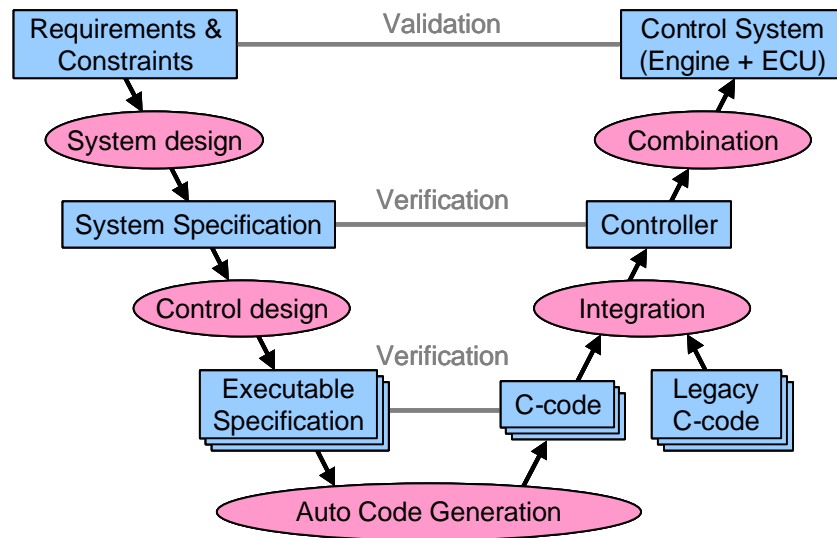


Figure 1: MBD Process (V-model)

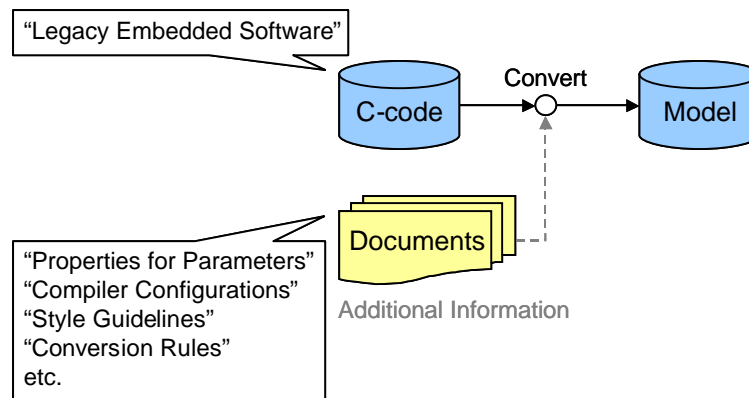


Figure 2: Conversion Concept

## 2. THE IMPORTANCE OF PROCESS

Although the conversion process may appear to be a routine translation from one language to another, there are many process nuances and product constraints that make this task difficult. For instance, translating fixed-point arithmetic code into target independent models; isolating highly inter-dependent code for unit testing; interpreting the wide usage of compiler directives; and creating a signal data dictionary. In order to have a very time-efficient and productive process which maintains a high level of translation accuracy between the legacy C-code and the converted models, there are fundamentally two methods to ensure translation accuracy - implementing “correct by construction” techniques and trapping errors. Process efficiency is attained by leveraging automation throughout the process.

The first step in this project is the development of a structured translation process which clearly lays out the path from start to finish. A process definition is extremely important since the actual translation task is enormous and it is executed by many people. Without having a structured process, it may be difficult to predict and measure the quality of the work products, the complexity of the tasks and the time required to complete the tasks.

Figure 3 shows a process designed to accommodate the translation of units of source code. There are several steps in the process which require specialized skills. Instead of a single engineer working on all the steps sequentially, different engineers working simultaneously and contributing to the work product is effective. So, the translation process is similar to an

“Assembly Line” where “components” are completed by different engineers at different steps of the process and they all come together to deliver the product. This “Assembly Line” implementation makes the translation process very efficient while delivering good quality.

Translation efficiency is enhanced by high levels of process automation. Automation is leveraged heavily throughout the process so that fewer errors are introduced by manual steps and the work product out of every step is automatically checked for bugs and inconsistencies. So, in addition to maximizing efficiency, automation enables a high level of Quality Control.

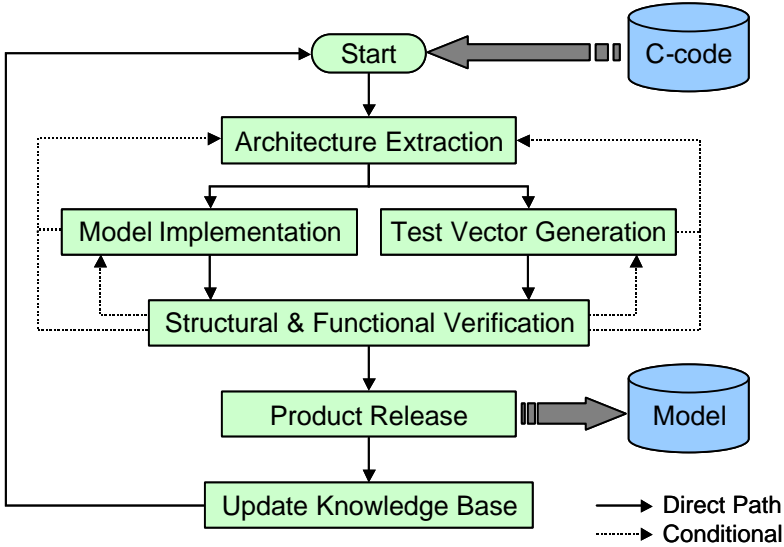


Figure 3: Conversion Process Diagram Comprised of Several Activities

**3. FUNDAMENTAL PROCESS ACTIVITIES**

**A. Architecture Definition and Extraction**

Architecture definition serves as the basis of conversion. It provides a top-level viewpoint required to partition the code into units or subsystems. Partitioning extracts valuable functional and structural dependencies as well as the unit interface. Achieving this requires incremental inclusion of external C-Code. The source C-code may have complex dependencies and this may make it unreasonable to include definitions of all externally called functions. So, some externally defined functions should be treated as Unknown Functions with no contents while translating at the unit level. Once defined, C-code units can be translated and tested independently in a distributed development environment. Consequently, a large translation task can be efficiently executed as several independent sub-tasks. The extracted architecture is the backbone to all process automation. The test vector generation leverages the architecture information to achieve coverage. For this reason, the quality of the final model greatly depends on the correctness of the extracted architecture. With this in mind, strict quality control should be implemented on the architecture definition using peer reviews and automated consistency checks.

**B. Model Implementation**

The model implementation must take the objectives of translation accuracy and efficiency into account. Techniques must either facilitate in building models that are “correct by construction” or highlight errors. The translation of the internals of the source code to Simulink® constructs involves substantial time and effort. The use of custom libraries and revision control aid the conversion. Different parts of the code unit can be independently modeled as library components and then the unit can be integrated and tested. To have a “correct by construction” model requires mapping rules for each code construct to a modeling construct. These rules ensure that (1) there is a uniform approach to translation by different team members, (2)

constraints of tools (e.g. TOYOTA-DENSO auto-coder) are satisfied upfront during model implementation, (3) the model is easy to read and comprehend, and, (4) the model satisfies TOYOTA specific standards. TOYOTA and DENSO already have original style guides based on the MAAB (The MathWorks Automotive Advisory Board) Style Guide. The adherence of the model to the style guides can be checked using Model Style Checkers. Such style checking tools detect and correct modeling errors and ensure a good model quality. Some examples of the mapping rules can be realized in a custom blockset of model primitives.

Fixed-point arithmetic and logic greatly complicate code translation. There may be constructs in C-code that need to be ignored, or, those that have to be mapped to a specific modeling construct. On the other hand, there are special constructs which have to be implemented to reasonably mimic the fixed-point arithmetic. As an example, equality operations on fixed-point signals whose real-world values are not discrete demand special consideration. In this case, two signals can be considered to be equal if they are within a tolerance of each other. So a direct comparison of signals in the floating-point model is prohibited and a custom “Equality Relational Operator Blockset” is used.

### **C. Quality Control**

This section will discuss the problem of judging the quality of the Model.

#### *I. DLL Generation*

To ensure the quality of the source to model conversion process, a complete set of testable specifications are required. In the case of converting C-code to a Simulink® model, compiling the legacy code into a MATLAB DLL is simple and objective. Even so, there are several challenges that must be addressed. For example:

1. Preprocessor directives allow for multiple configurations of the legacy code,
2. The DLL may be implemented in fixed-point arithmetic while the model is implemented in floating-point.

The presence of preprocessor directives may require multiple DLLs to have full MCDC source coverage. In practice, even small code units may require numerous DLLs. For this reason, a complete tool-set that automatically and rapidly instruments the legacy code for coverage, discovers compile configurations, and compiles the code with a fixed point to floating point wrapper has been developed.

#### *II. Test Vector Generation*

Test cases represent a behavioral specification of the source code. There are many types of coverage that help quantify behavior. Examples are MCDC, look-up table, operator, or data coverage. Each coverage method helps find test cases that quantify the behavior of the code unit. Using MATLAB tools, we have created test vectors automatically based on DLL coverage. The test vector generation continues until coverage requirements are met. In our case, 100% code coverage of original methods is required. Together, the test vectors and the DLL form a numeric function specification for the model.

#### *III. Verification Analysis*

Regarding comparing the responses of the code and model when subjected to the test vectors, there are several issues. For example, the resolution of fixed-point algorithms is limited by the scaling value assigned to least significant bit (LSB), unobservable signals, dead code, or overflows to the verification process. Any viable approach for converting legacy code to a model domain must consider these problems.

#### *IV. Peer Review*

The quality of the model cannot be ensured through verification technologies alone. The model quality is deemed poor if it does not meet all of the style guidelines. A style check of the model has to be performed to ensure that it adheres to all guidelines. Since some of the style guidelines are not objective enough for automatic checking and some errors are unobservable,

a manual peer review process is useful. Each model is subjected to two or more levels of peer review. The peer reviewer visually checks the model for functional accuracy, style guide violations, or, readability and usability issues. A style-checking tool can be used to check for violations. The peer review process is time consuming but it improves the quality of the model conversion.

#### **4. SUMMARY**

This paper describes recent activities at TOYOTA toward MBD. In particular, a process to efficiently and accurately translate documented legacy specifications into executable specifications (i.e. convert existing embedded C-code into Simulink® models) has been developed. Adherence to a structured process is one of the key enablers for the successful execution of the conversion project.

The verification method presented in this paper is crucial in establishing the quality and correctness of the translated models. Although effective, it is expected that it will continue to improve in the future into an efficient and practical V&V tools/environment.

The result of this project will bring about great changes to the powertrain control and embedded software development style in TOYOTA and DENSO.

#### **ACKNOWLEDGMENT**

It is a tribute to all parties concerned that the project went according to plan. The authors would like to thank them of TOYOTA MOTOR CORPORATION, DENSO CORPORATION, EMMESKAY INC. and TOYOTA TECHNICAL DEVELOPMENT CORPORATION.