Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Improving Model-based Design for Automotive Control Systems Development

Akira Ohata, *Toyota Motor Corporation*,

Kenneth R. Butts, *Toyota Motor Engineering and Manufacturing, North America*

Abstract— **A Model-Based Development (MBD) Framework is proposed in this brief-paper. The paper motivates the need for integrating automotive control system development activities and describes engineering roles in such an environment. It also outlines the features of an integrated development environment designed to support these roles.**

I. Introduction

It is an interesting era for automotive control system designers. On one hand, control system designers drive innovation in technologies that provide reduced fuel consumption, reduced emissions, improved vehicle dynamics, and active safety. Thus it appears control engineers will play an important role for the foreseeable future. On the other hand, control system development is very time consuming and, thus, in conflict with the industry's constant push to shorter development cycles. Consider that engine control development is on the critical path in automotive development and thus can constrain vehicle offerings. A challenge, then, for automotive control system designers is to deliver ever more complex and mission-critical systems with dramatically increased productivity.

We have identified model-based development for concurrent plant and controller engineering as a measure to deliver more complex systems with quality and improved productivity [1], [2], [3]. The purpose of this brief-paper is to focus more specifically on the control system design domain and identify critical areas for control-system-development-environment research and development. This paper includes a presentation of the major development roles (Section II) in a model-based automotive control system development process. Section III describes features that should be considered when constructing an integrated development environment for the roles described in Section II. We conclude with an illustrative verification and validation use-case for the integrated development environment in Section IV.

II. Development Roles

The control system development process we consider is shown in Figure 1. This is a generalized and abstract rendition of the 'V' process that is familiar to automotive engineers. Here we map model-based development roles onto the process. The basic principle is to use models at each stage of 'left-side' development to a) confirm prior to advancing to the next stage and b) establish test scenarios (and expected behavior) for 'right-side' verification and validation [4].

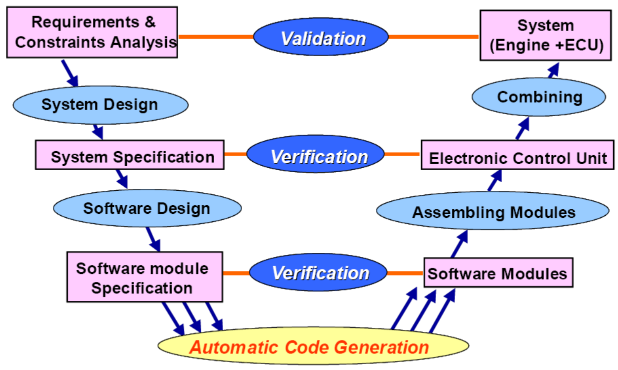

Fig.1. Automotive Control System Development Process [3]

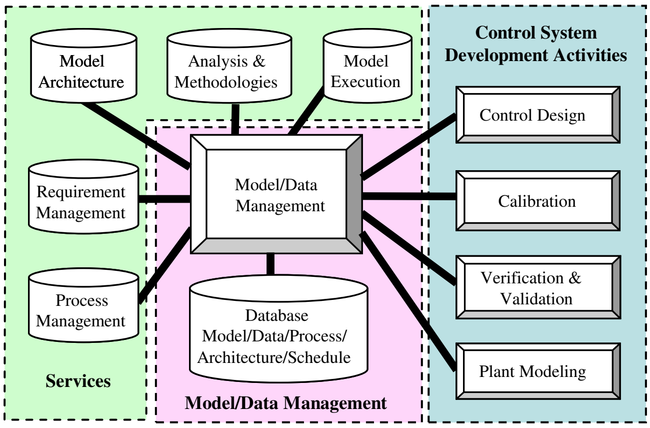

Fig. 2 Model Based Development Framework

There are four major activities in control system development. Each activity is supported by many services in the MBD Framework. The model and data instances are managed as shown in Fig.2.

A. *Control Design*

The control engineer establishes the externally driven product and project requirements, confirms traceability between requirements and design elements, assists with change impact analysis, and confirms that all requirements are validated or verified. He sets the architecture of the control system, specifies plant model behavioral phenomena needed for design and confirmation, and cascades system requirements to sensor, actuator, and control algorithm developers. The system engineer will make extensive use of open and closed-loop simulation and analysis. He uses model-based and heuristic control synthesis techniques to create an "executable specification" model of the desired software module behavior.

A. Plant Modeling

The plant modeler develops models of the physical plant to be controlled. Each model should be 'purpose-built' to support the engineering task at hand: requirements analysis, control design, or verification and validation.

B. Calibration

The calibration engineer sets the tuning parameters within the control system to meet performance criteria and confirms (validates) the system from a customer perspective. Note that even though calibration is traditionally executed at the last stage of control system development, some level of calibration is required in a model-based process in order proceed from 'left-side' design stage to stage.

C. Verification and Validation

The Verification and Validation engineer designs and executes test-plans to confirm the development at each stage of process. These test-plans will be model-based wherever possible because model-based testing admits more exhaustive testing using coverage-based and extended scenario techniques, scenario and results reuse, automation, and formal verification.

III. Features for a Control System Development Environment

Automotive control system development is a large and inter-disciplinary effort, thus an effective control system development environment should provide project collaboration support as well as domain specific engineering and project management methods. Moreover, the environment should provide integrated access to past project work products to facilitate consistency and reuse across the product-line. In this section we outline useful development environment features in seven categories. Features marked with (D) indicate that, in our judgement, significant development is needed while features marked with (I) require mainly framework integration.

A. Process and Project Management

Process and Project Management features to be addressed include:

- Requirements Management and Analysis (I)
- Process Definition (I)
- Process Integration (I)
- Project Planning and Scheduling (I)
- Project Status Reporting (I)

B. Plant Modeling

Plant Modeling features to be addressed include:

- Conservative Modeling Methods (D)
- Statistical Modeling Methods (I)
- Model Simplification Methods (D)
- Parameter Identification Methods (I)
- Integration of Physical and Statistical Models (D)
- Solvers for Acausal Model formulations (D)
- Model Confirmation & Validation Methods (D)

C. Control Design

Control Design features to be addressed include:

- Control architecture approach (I)
- Model-based Control Synthesis Methods (I)
- Control simplification methods (D)
- Controller Quality Evaluation Methods (D)
- Controller Specification Methods (I)
- Data Dictionary (I)
- Automatic Code Generation Methods (I)

D. Calibration

Calibration features to be addressed include:

- Test facility specification and management (I)
- Test and Measurement Automation Methods (D)
- Optimization Methods (I)
- Empirical Modeling Methods (I)
- Design-of-Experiments Methods (I)
- Calibration Quality Evaluation Methods (D)

E. Verification and Validation

Verification and Validation features to be addressed include:

- Open-Loop Verification Services (D)
- Closed-Loop Verification Services (D)
- System Assertion Definition (D)
- Verification and Validation Manager (D)
- Verification and Validation Platform (D)

F. Model and Data Management

Model and Data Management features to be addressed include:

- Global Repository Services (I)
- Versioning Services (I)
- Model Architectural Services (D)
- Configuration Management Services (I)
- Model Compilation Services (D)
- Search and Query Services (I)

G. Model Execution

Model Execution features to be addressed include:

- Model- (I), Software- (D), Processor- (D), and Hardware- in-the-Loop (I) Services
- Reusable Test-plans (D)
- Distributed Computation Services (D)
- Automated Execution Methods (D)
- Simulation Monitoring and Evaluation Services (I)

IV. Sample Development Environment use case: Verification and Validation

In the following, we partially describe, via use case format, how an engineer could use the integrated framework to verify a newly designed control algorithm.

In our architecture, we call the development activities mentioned earlier 'domains'. A domain consists of people, processes, methods, work products, and templates / standards. Referring to Figure 3, the Domain Manager is responsible for coordinating the flow of information within the domain and at the domain interface to the core framework. This component diagram also shows the primary services to be accessed by the V &V domain.

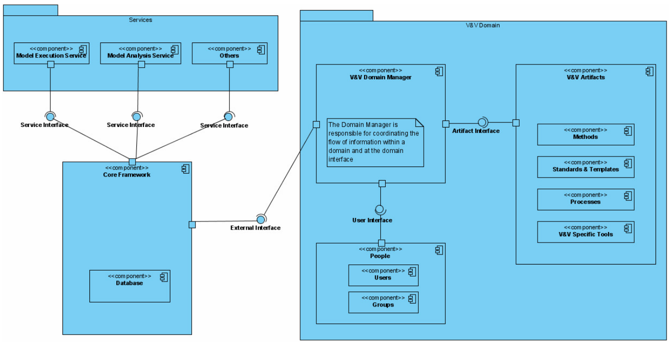

Fig. 3 V & V Framework Component Diagram

Figure shows the top-level use case diagram for the V & V domain. The use case describes the activities of the Customer, Team Lead, Field Specialist Engineer, and V & V Engineers.

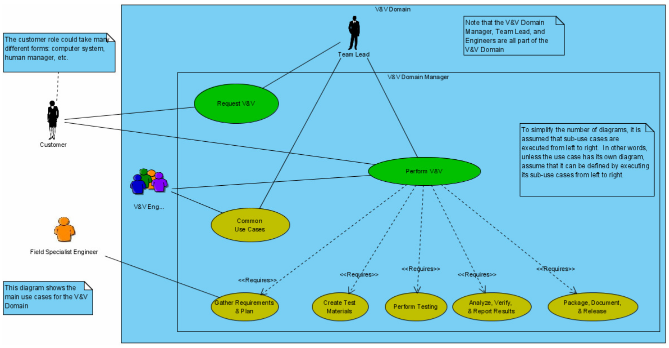

Fig. 4 Top-level Use Case: V & V Domain

Once the V&V Request has been accepted, then it must be performed. This 'Perform V&V' use case is still high level, and is comprised five sub-use cases. These five use cases are:

1. Gather Requirements and Plan
2. Create Test Materials
3. Perform Testing
4. Analyze, Verify, and Report Results
5. Package, Document, and Release

Note that these steps could, in-turn, have further sub-use cases. In general, the order of the sub-use cases (left to right) reflects the ordering that the use cases should be completed. One of these sub-use cases, 'Perform Testing', is shown in Figure 5. For this step, engineers execute the V&V activities while confirming that the results are valid (not that the work product passes, but that the test results are not flawed.)

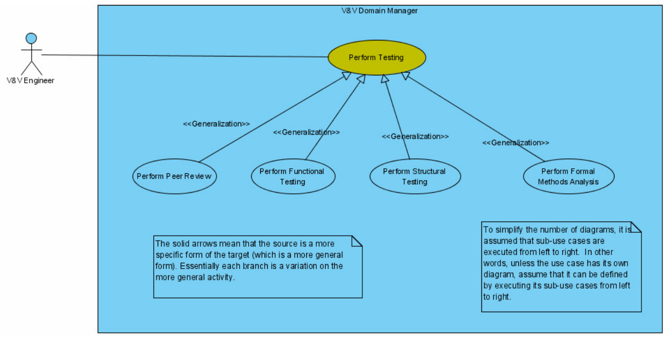

Fig. 5 Perform Testing Use Case

Finally, in Figure 6, we reach the leaf nodes in this portion of the use case structure. At this point we can introduce the sequence diagram to show actual transactions between actors and elements in the framework. Figures 7 and 8 are the sequence diagrams for 'Execute Tests' and 'Check Diagnostics' respectively.

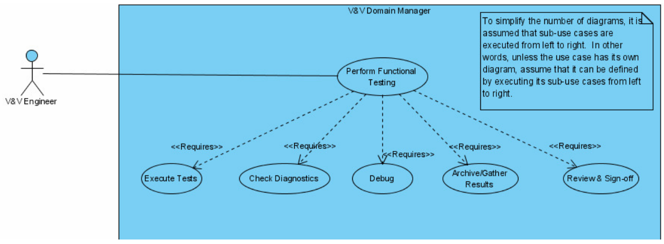

Fig. 6 Perform Functional Testing Use Case

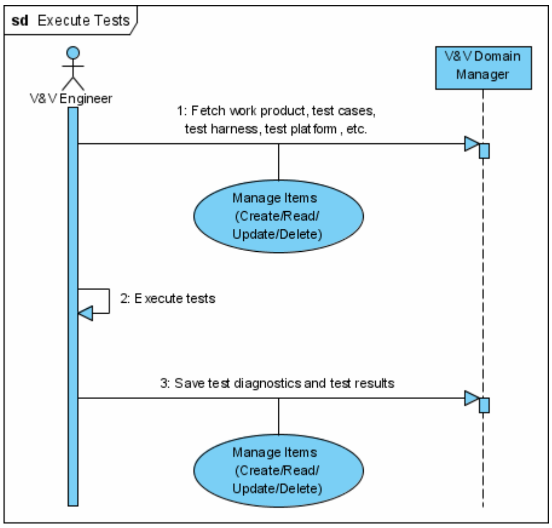

Fig. 7 Execute Tests Sequence Diagram

We notice that the sequence diagrams reference use cases multiple times. Consider the 'Manage Items' use case: this action contains interaction with the core framework database that allows multiple users to share their work in a collaborative, and geographically distributed development environment. In general, this action involves creating, reading (i.e,. fetching a copy of something), updating (committing and tracking the modifications), and deleting items. 'Update Project Status' is another common use case that can be seen in Figure 8. This action updates the project status for coordination and monitoring purposes thereby further facilitating distributed project development and project management. Thus, we can see an example of how the V&V engineer provides information to the Team Lead who may be interacting with the framework via the 'Process Management' service.

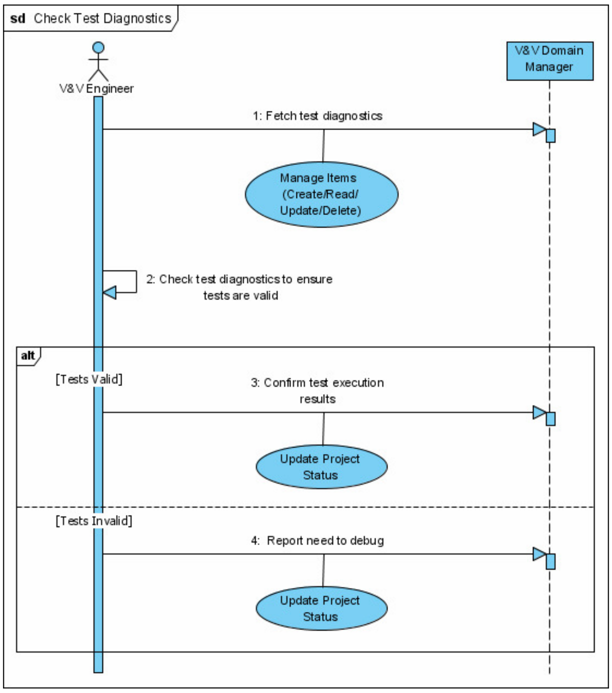

Fig. 8 Check Diagnostics Sequence Diagram

As we elaborate the various use cases, we can collect all of the reusable (typically low-level) use cases into a special container called 'Common'. Figure 9 shows all of the common use cases that we have collected for the V&V Domain. Inspection reveals that some (e.g. 'Use COTS Tool') are domain specific and others are more general (e.g. 'Arrange Meeting'). The more general use cases will likely be collected into a 'Framework Common' container. Regardless, the common container indicates opportunities for data / service integration and automation across the framework.

Fig. 9 V&V Common Use Cases

Acknowledgements

The authors would like to acknowledge Emmeskay, Inc. who are helping establish the user requirements for the integrated development environment described in this paper. In particular, Section IV is taken almost verbatim from their work.

References

[1] T. Ueda, A. Ohata, "Trends of Future Powertrain Development and the Evolution of Powertrain Control Systems," in *Proc. 2004 SAE –Convergence Conference,* October, 2004.

[2] A. Ohata, K. Butts, "Towards a Concurrent Engine System Design Methodology," *Proc. American Control Conference*, Portland, OR, June 2005.

[3] T. Ueda, "Innovative Development Methodology Based on the Toyota Way," *Internationales Wiener Motorensymposium*, 2007.

[4] K. Butts, J. Cook, C, Davey, J. Friedman, P. Menter, S. Raman, N. Sivashankar, P. Smith, and S. Toeppe, "Automotive Powertrain Controller Development using CACSD," Chapter15, Perspectives in Control Engineering Technolgies, Applications, and New Directions, T. Samad, editor, Wiley-IEEE Press, 2000.