

System Modeling Technology for Powerplant Control Systems

Makoto MURAKAMI*

Kazunori NOMURA*

Takashi HISAKI*

Yasuhiko KASAHARA*

ABSTRACT

Developing automobile control systems requires shortened development times even as it becomes more complex each year to enhance fuel efficiency, take environmental measures and meet high safety standards. The more complex and large-scale a system becomes, the greater the impact that upstream design precision during V-process development has on subsequent processes, and the greater the issue becomes of shortening development time. To resolve this issue, this research devised a technique that uses a use-case scenario and a Quality Function Deployment sheet (QFD sheet) to elicit requirements. In addition, it uses a total trial sheet (TTS) incorporating elements of Model Driven System Engineering (MDSE), a requirements analysis and system model design technique, to analyze requirements and design a system model, thereby increasing specification completeness at the start of development.

1. Introduction

A development style called a V-process, as shown in Fig. 1, is frequently used to develop control systems. This process lays out the series of actions, from examination of specifications to completion of product, in the form of a V. A product is designed and then verified; if there are failures, the specifications are amended and the product verified again, and as this cycle is repeated, development goes forward. Recently, techniques have been used in which the developer uses simulation tools like Matlab or Simulink to design specifications up to the point of verification and do software development^{(1), (2)}. This lets the developer detect design errors in advance, such as logical errors in the specifications or inconsistencies in specifications and implement codes, which reduces the burden during actual equipment verification and lets software development proceed efficiently. In power plant development heretofore, consistencies between specifications and implement code enabled a high degree of development efficiency. On the other hand, like the electronics field in which the enhancement of microcomputer performance has enabled a rapid increase in the number of software applications,

it is said that the greater the scale of the system under development, the greater the share of development that goes into eliciting and analyzing requirements and designing the system model⁽³⁾. This makes it important to increase not only consistency between specifications and implement code, but also between pre-implementing requirements and specifications.

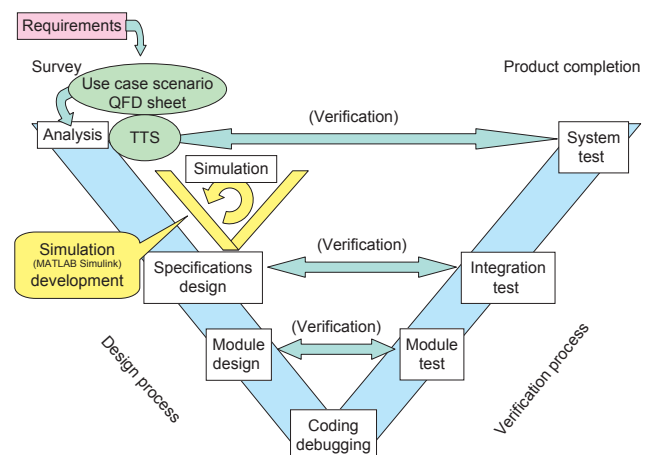


Fig. 1 V-process development

* Automobile R&D Center

Automobile control system development is expected to become more complex in the future, and a variety of initiatives have been tested^{(4), (5)}; increasing specification completeness at the start of development requires examining procedures that make requirement elicitation and analysis and system model design more efficient.

2. Development Goal

During power plant control system development, control specifications are written out to a point close to implement code in order to enhance development efficiency and quality. This has achieved shorter software development times and enhanced quality. On the other hand, it has become apparent that the grounds and particulars of such control specifications are hardly understood by anyone other than the designer, making it a challenge to convey the designer's technical intention to others.

To solve the above issues relating to the growing scale of systems hereafter and enhance development efficiency, it is important to elicit and analyze requirements and design system models efficiently at the design stage upstream in the V-process development and to document the process in a form understandable to those who are not directly involved in it, so this was set as the development goal.

The following techniques were devised as means for achieving the above aim.

- (1) A technique for eliciting requirements with a use-case scenario and QFD sheet
- (2) A requirements analysis and system model design technique using a TTS incorporating elements of MDSE

The relationship between techniques (1) and (2) is shown in Fig. 2.

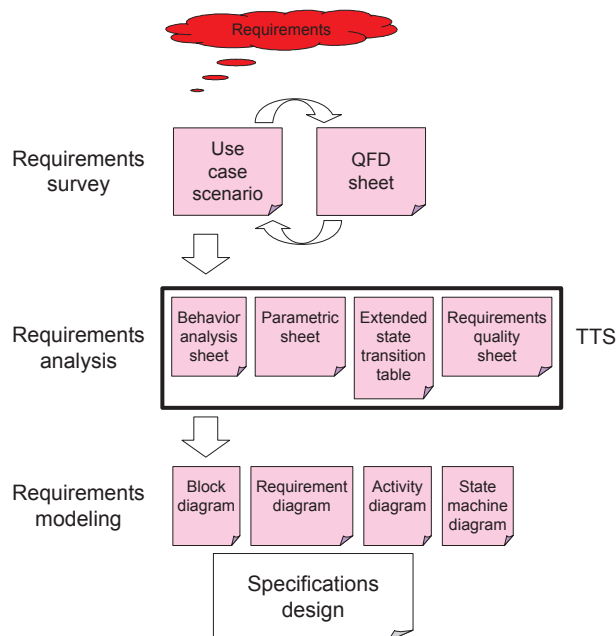


Fig. 2 Development flow

3. Requirements Elicitation Technique

To elicit the requirements at the upstream design stage, a use-case scenario is used, and a QFD sheet is drawn up to clarify the system configuration, constraints and requirement quality for those requirements.

3.1. Eliciting Requirements with Use-case Scenario

A use-case scenario⁽⁶⁾ sets up a scene to make it easy to elicit requirements and help the designer find requirements based on that scene. At this stage, a list is made of typical scenes in which the control is used, focusing on the when, where, what, who, why and how. A scenario is created that basically describes the listed scenes in text format, and first a main scenario is drawn up, that is, a scenario in which the system works normally. Figure 3 gives an example of such a description used for a power plant ECU control specification. Cases in which the condition of normal operation in the main scenario did not materialize are drawn up as extension scenarios. Additionally, exception scenarios are brought to light based on past troubles and designers' knowledge to find exceptional requirements. Figure 4 gives an example of an extension scenario and exception scenario used for a power plant ECU control specification. Thus, the scenario descriptions describe the system's functions as behaviors.

Inquiry work of scenario according to scene

Main actor: Compatibility check

Actors: Compatibility information, MIL, meter, ECM, TCM

Case 1: Cycle of usual drive

Case1-1: A compatibility check transmits compatibility information to other units, when it becomes Ignition-ON.

Case1-2: A compatibility check will perform a compatibility check if compatibility information about other units is received.

Case1-3: The compatibility check compares the compatibility information on other units with compatibility information on the subject unit, and judges a compatibility error.

MIL; Malfunction indicator lamp

ECM; Engine control module

TCM; Transmission control module

Fig. 3 Use case scenario

Case1: Cycle of usual drive

<Extended scenario>

Case1-1: *****

Case1-e1: *****

Case1-2: A compatibility check will perform a compatibility check if compatibility information about other units is received.

Case1-e2: When compatibility information about other units cannot be acquired, a compatibility check judges a compatibility error, and ends the compatibility check of this driving cycle.

Case1-3: *****

Case1-e3: *****

<Exception scenario>

Case1-ex1: After Ignition-ON, when compatibility check judged that the power supply was shut off, And compatibility check will perform a compatibility check again.

Fig. 4 Extension scenario and exception scenario

(4) Technical requirements selection

The requirement quality is elicited by checking to what quality characteristics the solutions to the chosen issues relate. An example is shown in Fig. 8.

(5) Categorization according to quality characteristics

A check is made of what quality characteristics the solutions to the elicited issues relate to. Also, the comprising

components from the QFD sheet are checked to clarify the system configuration. An example is shown in Fig. 9.

As above, the system configuration, constraints and requirement quality are clarified by QFD sheet. The clarified system configuration, constraints and requirement quality as well as the new scenario are reflected in the use-case scenario and managed as requirements documentation.

(4)								
Technical requirements								
Technical requirements	Relation post							
	Driver	Car	Engine	Service	ECM	TCM	Meter	Regulations
Add compatibility communicative function					○	○	○	○

Fig. 8 Relation post

(5)																												
Quality characteristics																												
Amenity	Quality	Vehicle status	Marketability															Destination	Safety									
			Start	Shift position	Water temperature	Weight	Price	Acceleration	Warranty	Environment	Security	Cost	Convenience	Meter	Fuel consumption	Running	Starting		Vibration	Sound insulation	Drivability	Noise	Japan	USA	Seat belt	Battery	User	Safety
	Guarantee																											
	Maintenance																											
																												</

Fig. 9 Quality characteristics

4. Requirements Analysis Technique

The use-case scenario, in which the system configuration, constraints, and requirement quality were reflected by eliciting requirements using the QFD sheet, is laid out on a TTS so that it can be analyzed efficiently and with high precision. A TTS sums up the requirements in table form in terms of behavior, configuration, constraints and requirement quality in order to organize requirements documentation that has been written in natural language.

The following provides details of the four perspectives and procedures for system model design.

4.1. Model Design by Behavior Analysis

“Behavior analysis” refers to eliciting and analyzing lines of text from the requirements documentation that describe the states and transitions of the system and actions in each state.

The analysis procedure is to start by selecting lines from the requirements documentation that express behavior, separate each line into component, prior state, trigger and action and write these onto the TTS behavioral constructs study sheet. If a document is written without subject or object, the person analyzing requirements has to conjecture the missing words based on earlier and subsequent documents and add them to the behavioral constructs study sheet. An example is shown in Fig. 10.

Also, in requirements documentation written in natural language, the same content is sometimes written in different words, and if this is left unchanged when the model is designed, the result could be a complex model with many similar states and triggers. This situation is controlled by term unification (i.e., organizing similar or duplicate names). An example of term unification is given in Fig. 11. Next, an extended state transition table is created based on the states, triggers and actions organized by term unification

(Fig. 12). States are placed in rows and triggers in columns, and actions and transition results are written in for each combination. The extended state transition table includes the information necessary to create a state machine diagram.

Creating a state machine diagram makes it possible to discover transitions that would not be easy to discern in requirements documentation written in natural language (Fig. 13).

Serial	Requirement															
	ID															
1	1	A compatibility check transmits compatibility information to other units, when it becomes IG-ON.														

Prior state				Trigger				Relationship (Prior state and trigger)	Action							
State ID	Block	Property	Condition	Trigger ID	Block	Property	Action		Action ID	Block	Object1		Object2		Action or state	
C0101	System		IG-OFF	T0101			IG-ON	C0101&T0101	A0101	Compatibility check	Block	Property	Block	Property	Operation mode	Verb
												Compatibility information	Other units			Transmits

Fig. 10 Total trial sheet (behavioral constructs study)

Serial	Prior state			
	State ID	Block	Property	Condition
14	C0901	System	State	Compatibility judge
15	C2801	System	State	Compatibility check
16	C3001	System	State	Compatibility analysis

Serial	Prior state			
	State ID	Block	Property	Condition
14	C0901			
15	C2801	System	State	Compatibility check
16	C3001			

Fig. 11 Example of term unification

				State ID	S10	S100	S200	S210	S220
				State name	Sleep	Initialization	Compatibility check		
							Data not received	Compatibility judging	Complete
Trigger									
ID	Block	Property	Action						
T0401	IG	State	ON		S100 TX0401				
T0402	ECM	State	Initialization process is completed			S200 TX0402			
T0501	ECM	State	Received compatibility information				S210 TX0501		
T0601	ECM	State	Judged compatibility (Pass)					S220 TX0601	
T0602	ECM	State	Judged compatibility (Fault)					S220 TX0602	
T3101	IG		OFF		S10 TX3101	S10 TX3101	S10 TX3101	S10 TX3101	S10 TX3101
Action ->						AC0401	AC0402	AC2801	

Fig. 12 Extended state transition table

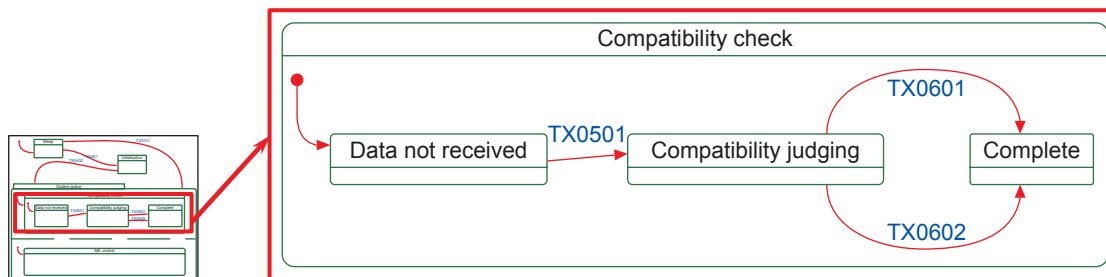


Fig. 13 State machine diagram

4.2. Model Design by Configuration Analysis

“Configuration analysis” refers to eliciting and analyzing lines of text from the requirements documentation that describe the configuration and relationships of components comprising the system and functions of those components.

“Components” refers to systems, parts and physical devices, and these have to be extracted both from a control system perspective and hardware perspective. In Section 4.1, components were picked out from a document expressing behavior. In this section, components are extracted from documents other than behavior documents (i.e., from “explanatory documents”) and term unification is performed. Based on term unification results, one checks whether each element depends on existing systems or techniques, creates a hierarchy based on level of dependence to existing systems (those with a low dependence level are ranked higher) and summarizes the results on a block diagram (Fig. 14).

4.3. Model Design by Constraints Analysis

“Constraints analysis” refers to eliciting and analyzing lines of text from the requirements documentation that describe specifications and constraints as determined numerically, such as various factors or elements and

target performance.

A constraints formula is examined from the parts of the requirements documentation in which specifications such as performance and limit values are expressed, where they are expressed numerically, and a parametric diagram is created by selecting the variables used in the constraints formula. As an example, Fig. 15 shows a parametric diagram that concerns ECM compatibility control. In this diagram, ECM and TCM have respective compatibility data within certain control values, and the diagram indicates that a compatibility judgment is made based on the relative size relationship.

4.4. Model Design by Requirement Quality Analysis

“Requirement quality analysis” refers to eliciting and analyzing sentences from the requirements documentation that describe quality requirements that cannot be expressed numerically.

Based on words that express degree, adverbs and adjectives in the sentences, as well as qualitative requirements of degree, state, etc. such as “optimal” or “automatically,” are identified and written on a requirement diagram (Fig. 16). In upstream design stages, work is done only up to the point of summarizing requirement quality

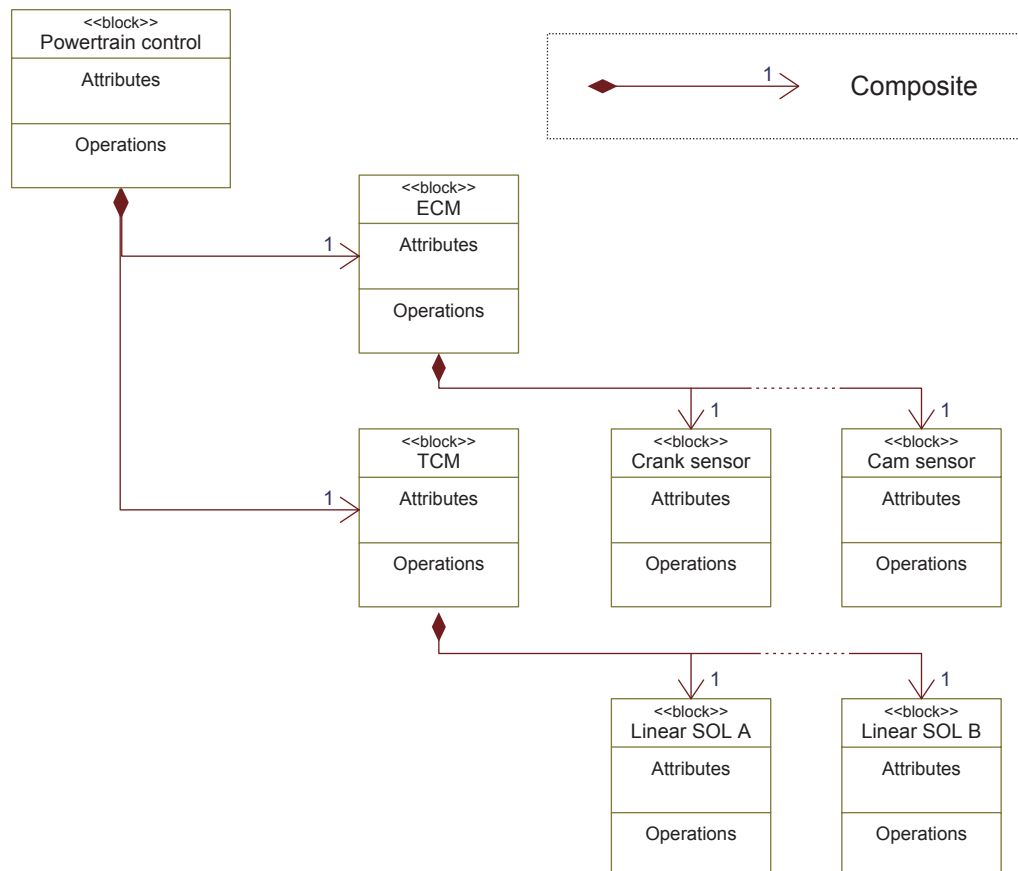


Fig. 14 Block diagram

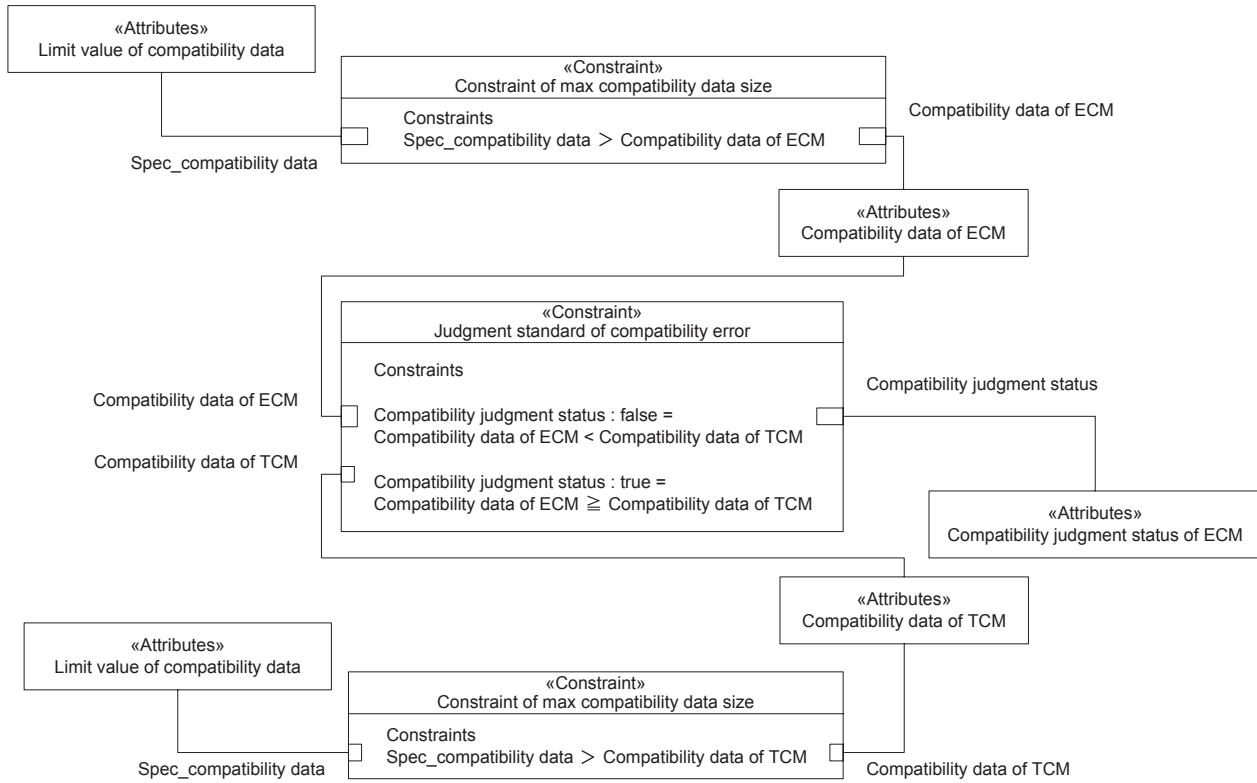


Fig. 15 Parametric diagram

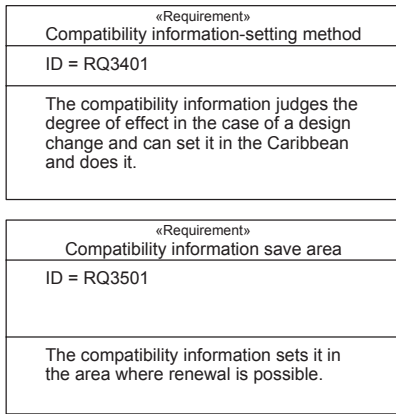


Fig. 16 Requirement diagram

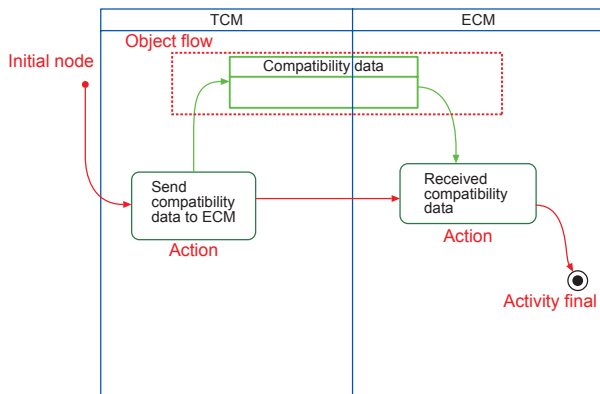


Fig. 17 Activity diagram

in a requirement diagram, and this is then developed into behaviors and constraints at the stage of examining specific ways of achieving it.

4.5. Behavior Detailing and Model Design

The state machine diagram created in Section 4.1. does not express the actions included in states. In order to clarify behavior, this section expands on functions using an activity diagram. Actions in each state are expressed using an activity diagram. In the activity diagram, information flowing out of the content of each action is anticipated and an object flow is added. Figure 17 gives an example of information sharing between components. Using an activity diagram to detail behavior can clarify data shared between components and help discover insufficiencies of data shared between components for the sake of continuity.

5. Conclusion

This paper has proposed a technique that can increase specification completeness at the start of development, help ensure a simple transition to downstream development with V-process development and keep work from being sent back upstream by using a use-case scenario and QFD sheet to pick out requirements and by using TTS incorporating elements of MDSE, a requirements analysis and system model design technique, to analyze requirements and conduct system model design.

References

- (1) Minowa, S., Hirota, T., Yokoyama, H., Ishiguro, T., Ohtsuka, S.: Autocoding of Power Plant Control Software, Honda R&D Technical Review, Vol. 16, No. 1, p. 243-248
- (2) Matsuyama, Y., Hagiwara, K., Seino, S.: Approach to Application of Executable Specifications in Development of Automatic Transmission Control Software, Honda R&D Technical Review, October, 2009, p. 120-129
- (3) <http://www.stevemcconnell.com/articles/art06.htm>
- (4) <http://www.autosar.org/>
- (5) Katai, M.: Denshi seigyo software kaihatsu no youkyu kanri prosess no jirei shoukai, IBM, AT International 2008 (in Japanese)
- (6) Cockburn, A., trans. by Yamagishi, K., Yazaki, H., Mizutani, M., Shinohara, A.: Writing Effective Use Cases, Shoueisha, p. 331 (2001) (in Japanese)

■ Author ■



Makoto MURAKAMI



Kazunori NOMURA



Takashi HISAKI



Yasuhiko KASAHARA