Electronic Communications of the EASST
Volume 46 (2011)

Proceedings of the 11th International Workshop on Automated Verification of Critical Systems (AVoCS 2011)

Structuring Functional Requirements of Control Systems to Facilitate Refinement-based Formalisation

Sanaz Yeganefard and Michael Butler

15 pages

Guest Editors: Jens Bendisposto, Cliff Jones, Michael Leuschel, Alexander Romanovsky
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Structuring Functional Requirements of Control Systems to Facilitate Refinement-based Formalisation

Sanaz Yeganefard[1] **and Michael Butler**[2]

[1] sy2g08@ecs.soton.ac.uk
[2] mjb@ecs.soton.ac.uk
Electronics and Computer Science
University of Southampton, UK, SO17 1BJ

Abstract: Good requirements structure can greatly facilitate the construction of formal models of systems. This paper describes an approach to requirements structuring for control systems that aims to facilitate refinement-based formalisation. In addition to the well-known monitored and controlled phenomena used to analyse control systems, we also identify commanded phenomenon reflecting the special role that an operator plays in system control. These system phenomena guide the structure of the requirements analysis and documentation as well as the structure of the formal models.

We model systems using the Event-B formalism, making use of refinement to support layering of requirements. The structuring provided by the system phenomena and by the refinement layers supports clear traceability and validation between requirements and formal models. As a worked example, we structured the requirements of an automotive lane departure warning system using this approach. We found missing requirements through this process and we evolved the requirement document through domain experts' feedback and formal modelling.

Keywords: structuring requirement, requirement engineering, validation, formal verification, lane departure warning system

1 Introduction

Control systems are usually complex as they continually interact with and react to the evolving environment. Because of the complexity of these systems, constructing and structuring their functional requirement documents (RD) can be a time consuming process. In addition, their RD may not be clear and complete for developers of the system. However, since these systems are usually used in life critical situations it is essential to have a comprehensive RD to help with the improvement of safety and reliability of the system.

Formal methods are mathematical based techniques used for specification and development of systems as well as verifying their properties [Win90]. Modelling using formal methods is known to improve system understanding and thus help to find missing and ambiguous requirements. However, one difficulty of using formal modelling is formalising an informal RD.

In this paper we propose an approach to construct the RD of control systems incrementally to help with understanding the system requirements and to facilitate the process of for-

mal modelling. This approach consists of three stages and is based on monitored, commanded and controlled (MCC) phenomena introduced in [But09] as an extension of Parnas' 4-variable model [PM95].

In the first stage an RD is constructed and structured incrementally through iterations, as our understanding of the system improves (i.e. by considering the requirements in more depth). The second stage involves modelling the RD in a step-wise manner by using refinement. Here, requirements are layered and each layer is modelled in one refinement level. The third stage of this approach deals with any identified missing and ambiguous requirements by revising the RD and the model.

This approach also provides the means for validating a model against its RD in order to ensure that the model is an accurate representation of the system's requirements. This validation also facilitates the traceability between a model and its RD.

As a worked example, we structured the requirements of an automotive lane departure warning system (LDWS) using the proposed approach. Requirements of this system are evolved in three phases. In the first phase we produce and structure the RD of the LDWS based on information in the *public domain*. In the second phase, the generated RD is discussed with *domain experts*. In the third phase, the RD of LDWS is *formally modelled* using Event-B formal language. Also, as will be discussed any changes in requirements, i.e. identified missing and ambiguous requirements, are applied to both the RD and the Event-B model.

This paper is organised as follows: in Section 2 MCC phenomena are discussed. An overview of the proposed approach is given in Section 3. Section 4 introduces the lane departure warning system (LDWS) briefly. The RD of the LDWS and its formal model are represented in sections 5, 6 and 7. The validation of the model against some of its requirements is shown in Section 8. Section 9 and 10 discuss the related and future work. In Section 11 some of the advantages of the proposed approach are represented.

2 Guidelines for Modelling Control Systems

The guidelines outlined in [But09] can be used for formal modelling of control systems. The formal models consist of variables and guarded actions (events) and control systems consist of plants, controllers and in some cases operators who can send commands to the controller, shown in Figure 1[1].

The modelling steps suggested in this guidelines are based on the four-variable model of Parnas [PM95]. Variables shared between a plant and a controller, labelled as 'A' in Figure 1, are known as environment variables and are categorised into *monitored variables* whose values are determined by the plant and *controlled variables* whose values are set by the controller. There are also *environment events* and *control events* which update/modify monitored and controlled variables respectively. The other two variable categories of the four-variable model are *input* and *output*. In [But09] it is suggested that these are not used in abstract formal model; instead [But09] provides patterns for introducing them as refinements.

If a system involves operators, according to [But09] in addition to the phenomena introduced in the four-variable model, phenomena shared between controller and the operator can be iden-

[1] The diagram uses Jackson's Problem Frame notation [Jac01].

tified. These phenomena, labelled as 'B' in Figure 1, are represented by *command events* which are the commands given by an operator and *commanded variables* whose values are determined by command events and can affect the way other events behave.

In [YBR10] a cruise control system is modelled following MCC guidelines. In addition [YBR10] shows that modelling based on the MCC guideline helps to have a more structured process of modelling and refinement for a control system.

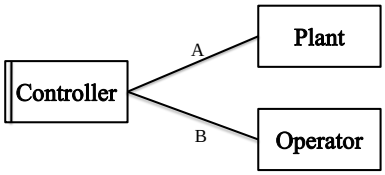


Figure 1: A control system.

3 Overview of the Proposed Approach

Modelling guidelines represented in [But09] requires the modeller to identify the MCC phenomena of a system before the commencement of the modelling process. This inspired us to propose an approach for structuring RD and modelling using MCC phenomena. This approach comprises of three stages, which are shown in Figure 2.

Notice that we use the term *phenomena* when we deal with (informal) requirements of a system and the term *variable* when we model the system formally. Also as our focus in this paper is not on requirement elicitation methods, it is assumed that the textual RD of the control system exists.

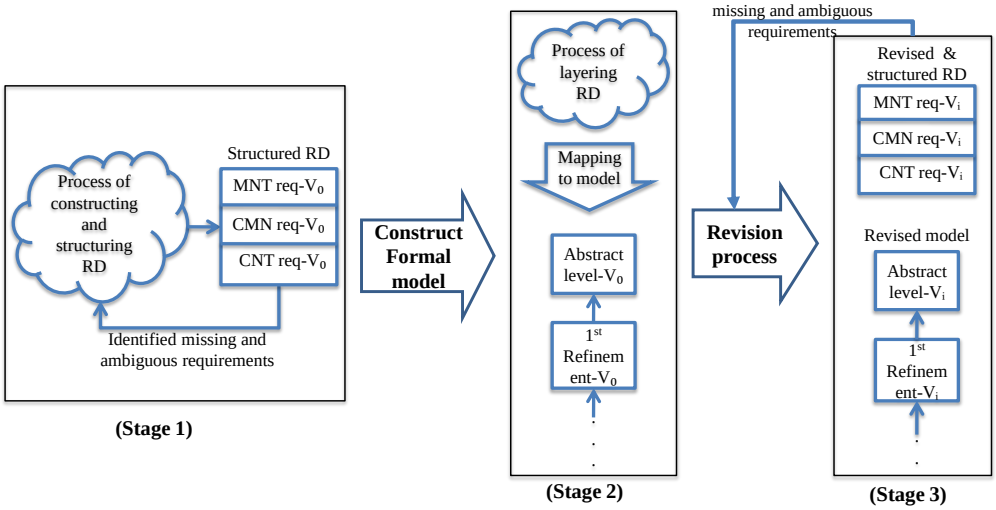


Figure 2: Overview of the three stages. Stage 1 - Structuring RD; Stage 2 - Layering structured RD and designing the initial formal model; Stage 3 - Revising the RD and the model.

3.1 Stage 1: Structuring Requirement Document

The following are the steps suggested for structuring the existing textual RD into monitored, commanded and controlled sections:

1. *Identify the system's MCC phenomena* based on its textual RD. An example is the monitored phenomenon *speed* in LDWS.
2. *Organise RD* into three monitored (MNR), commanded (CMN) and controlled (CNT) sections, each representing requirements of the corresponding phenomenon. If the requirement refers to
   - only one phenomenon, it will be moved to the relevant MCC section.
   - more than one phenomenon, but of the same type (e.g. they all are monitored phenomena), the requirement will be added to the corresponding section.
   - more than one phenomenon, but of different type, then it is designer's judgment which section is mostly appropriate.
3. *Add unique ID labels.* Every ID starts with the section that the requirement belongs to (i.e. MNR, CMN, CNT), followed by a unique number for that requirement.
4. *Revise RD* to accommodate any identified missing or ambiguous behaviour of the system. The revision step involves going back to Step 1 to identify any phenomena that the new requirement represents and then adding the requirement to the appropriate section.

The last step helps one to seek the most obvious MCC phenomena in the initial structuring of the RD and improve it incrementally through iterations. This iteration is shown in Figure 2-Stage 1. Section 5 represents structuring the RD of an LDWS using these steps.

3.2 Stage 2: Layering Requirements and Designing the Initial Models

In order to deal with the complexity of a control system, our aim is to use refinement to introduce system requirements in a step-wise manner. However, deciding on how to layer requirements and what to model in each levels is usually difficult.

We propose to overcome this problem by modelling one **feature** and the minimum number of requirements essential for this feature to be meaningful in one level of refinement. A feature is usually one of the MCC phenomena of the system. However, sometimes a phenomenon is interrelated to other phenomena and thus they should be modelled simultaneously. Examples of features for the LDWS are phenomena *warning* and *status*.

We also suggest to focus on the **main role or behaviour** of the system, which usually corresponds to a controlled phenomenon, in the most abstract level. If the system has more than one role, it is the modeller's judgment to choose the most important role to be initially modelled. This means the abstract model will focus on the role of the control system, while the rest of the requirements will be elaborated into the model through refinement levels. For instance, the main behaviour of an LDWS is to issue *warnings* and this is modelled in the abstract level. After that in the first refinement the phenomenon *status* is introduced. Section 6 describes this stage in more details through the LDWS example.

3.3 Stage 3: Revision of RD and Formal Model

Modelling a system formally can result in finding missing and ambiguous requirements of the system. We suggest to handle these requirements similarly to the revision step in Stage 1, where phenomena of new requirements are identified and based on them requirements are added to the corresponding MCC sections of the structured RD. However, in addition to revising the RD, it is necessary to update the formal model. This is because the RD and the model should be kept consistent to help with the process of validation and traceability.

If a new requirement is related to any of the previously modelled phenomena (modelled in Stage 2), this requirement can be modelled in the same refinement level as its phenomenon. For instance if the new requirement gives further information about the main behaviour of the system, which is modelled in abstract level, we update this level. However, if the newly identified requirement has no effects on any levels of the current formal model, for instance if the requirement introduces a new phenomenon, it can be introduced to the model in a new refinement level.

As shown in Figure 2, Stage 3 can be iterated meaning that as long as new missing or ambiguous requirements are identified, the RD and the model of the system should be revised. This stage is explained further in Section 7 using the example of LDWS.

4 An Overview of LDWS

LDWS is a driver assistance system which receives camera observations of the lane and uses this information to warn the driver of a lane departure, when the car is travelling above a certain speed. One way to detect the car departing the lane is by estimating the car's current position in the lane using lane detection algorithm on camera's observation [RME00].

In order to warn the driver before the vehicle crosses the lane, a virtual lane width which is inside the lane boundaries is assumed. This virtual lane, called the earliest warning lines (EWL), is determined by the LDWS based on the speed of the car. When the vehicle is within the earliest warning lines, the LDWS does not issue any warnings. This area is called the "no-warning zone", and the area pass EWL is the "warning zone" [Fed05], shown in Figure 3.

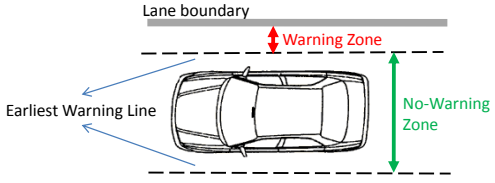


Figure 3: Warning and no-warning zone for an LDWS.

5 Stage 1: Constructing and Structuring RD of LDWS

In this section we discuss some of the requirements of the LDWS which are structured according to the first stage of the proposed approach, Section 3.1. The RD which is produced based on the information available in the public domain [Fed05, RME00, PMGB05] is outlined in Section 5.1.

This RD then is modified in Section 5.2 according to the feedback received from the domain experts.

5.1 First Version of Requirement Document

To structure the RD we first examined the public sources to identify the MCC phenomena. The identified **monitored phenomena** are *position* of the car relative to the centre of the lane, *lane width* and current car *speed* which also determines the EWL. The identified **commanded** and **controlled phenomena** are respectively *status* of the LDWS which is set through a switch button and *warning*. Requirements related to these phenomena are organised into MNT, CMN and CNT sections using appropriate identifiers for every requirement as shown in Table 1.

Table 1: First version of the structured RD of the LDWS.

Req ID	Requirement Description
MNR1	LDWS shall detect the earliest warning line (EWL) and vehicle position relative to visible lane boundaries based on the lane width and car width.
MNR2	LDWS shall track lane boundaries where lane markings are clearly visible in daylight (sunny/cloudy), night times and twilight (sunrise/sunset) lighting conditions.
MNR3	The width of the "warning zone" depends on the speed of the car. The higher the speed of the car the closer the earliest warning line (EWL) to the centre of the lane.
CMN1	LDWS can be switched on and off by the driver through a single button.
CNT1	LDWS would issue a warning when the vehicle has left the no warning zone (has crossed the EWL), and is entering the warning zone.
CNT2	When LDWS is on it would start its role provided that the car speed is greater or equal to a certain speed.

5.2 Second Version of Requirement Document

Discussing the first version of the LDWS's RD with domain experts from GM India Science Lab resulted in identifying some missing requirements. One of these requirements was that the driver can change the EWL by setting the offset they wish to have from the lane boundaries.

We retook the structuring steps and as the result the commanded phenomenon *offset* was identified. Thus, the three requirements related to offset, shown in Table 2, are added to CMN section. Notice that we refer to requirement MNR3 in CMN4 to show that these requirements are related.

6 Stage 2: Layering Requirements and First Model of LDWS

The initial formal model of LDWS was produced based on the second RD. We have used Event-B formal language which also provides supports for refinement. The process of modelling starts with identifying requirements necessary for constructing the abstract level. After building the abstract model, the remainder of the RD is introduced in refinement levels.

Table 2: Second RD - Requirements are added based on experts' feedback.

Req ID	Requirement Description
CMN2	The driver can set the offset they want to have from either side of the lane boundaries through two buttons which are responsible for increasing and decreasing the distance.
CMN3	The offset is always within a certain positive range.
CMN4	In addition to speed (MNR3) the width of the "warning zone" or earliest warning line depends on the offset from the lane boundaries. The greater the determined offset the closer the EWL to the centre of the lane.

6.1 Event-B and its Tool

For the purpose of this paper we only focus on two elements of an Event-B [Abr10] model. Firstly, *variables* and secondly *events*. An event consists of two elements, *guards* which are predicates defined for describing the conditions need to hold for event occurrence, and *actions* which determine the changes of state variables. An event becomes *enabled* if its guards hold. One of the advantages of Event-B is its open source tool, known as Rodin, which provides automatic proof and a wide range of plug-ins [ABH+10].

6.2 Requirements for Modelling the Abstract Level

As mentioned in Section 3.2, Stage 2 involves layering and modelling requirements based on the features of the system. In this stage, each feature is modelled in one level of refinement. Also, the main behaviour of the system (i.e. the main controlled phenomenon) is modelled in the most abstract level. Examining the structured RD of the LDWS shows that the requirement **CNT1** represents the main role of this system; "issuing warnings when the car has left the no warning zone". Thus, the controlled phenomenon *warning* is to be modelled at the abstract level.

After this we identify any interrelated phenomena, usually monitored and commanded requirements (MNR, CMN), which are vital for modelling *warning*. Thus, requirements related to crossing EWL should also be modelled at this level. These are firstly, the requirement **MNR1**, since the controller should receive the monitored phenomena *car position* and *lane width* in order to decide whether or not the EWL is crossed. Secondly, **MNR3** and **CMN4**, since the position of the EWL depends on the monitored phenomenon *speed* and the commanded phenomenon *offset*.

The next step is to identify requirements which describe the limitations and restrictions of the identified phenomena, or requirements which show how these phenomena affects/restricts the system. Based on this we need to model firstly **CNT2**, which describes the restriction imposed on *warning* by the system. The second is **CMN3**, as it gives details about restrictions of *offset*.

Finally we identify any requirements which represent state changes of the identified phenomena. This results in identifying **CMN2**, since it represents how *offset* can be modified. Notice that in Event-B requirements which are identified as restrictions of phenomena are usually modelled as guards for events or as types of the phenomena. Also, requirements which define state changes are usually modelled as actions of events.

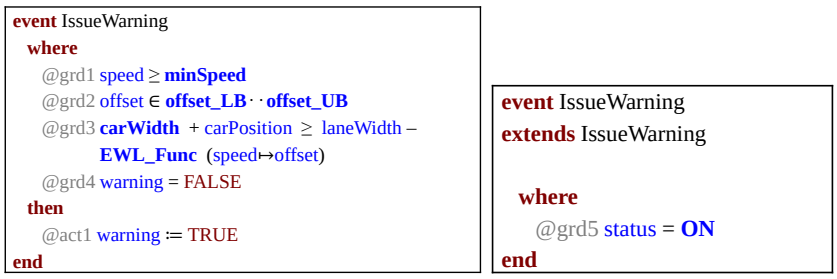
6.3 Modelling Abstract Level

In the previous section, requirements and phenomena for modelling the abstract level were identified. In this section we start modelling the abstract level by representing each of these phenomena as a system variable. Also according to [But09], the corresponding *environment*, *command* and *control* events for every variable is defined.

At this level, the monitored variables are *speed*, *laneWidth*, and *carPosition* (the difference between the car centre and the lane centre). The LDWS also needs to know the width of the car, modelled as the constant *carWidth*, in order to detect the value of *carPosition*. Environment events which are responsible for modifying monitored variables are defined to simply set the monitored variables non-deterministically through a parameter (MNR1 and MNR3).

The commanded variable modelled at this level is *offset* (CMN2). Since the value of *offset* is within a specific range (CMN3), two constants called *offset_LB* and *offset_UB* are defined to represent the lower and upper bounds of this variable. The command events which modify the value of *offset* are *IncreaseOffset* and *DecreaseOffset*. Also, the controlled variable *warning* is defined as a Boolean variable. The control event *IssueWarning*, shown in Figure 4a, is defined to set *warning* to TRUE when the car crosses the EWL (CNT1).

To model crossing EWL, we firstly define a function named *EWL_Func* which returns the distance of the EWL from the lane boundaries based on the car *speed* and the *offset* (MNR3 and CMN4). This function is defined as $EWL\_Func \in \mathbb{N} \times offset\_LB..offset\_UB \rightarrow \mathbb{N}$, meaning that for every possible tuple of *speed* (of type $\mathbb{N}$) and *offset* (within the range $offset\_LB..offset\_UB$) there is a value for the *EWL_Func*. We assume that the return value of this function, which represents the position of EWL, are provided. Based on this function *grd3* in Figure 4a is defined to model that event *IssueWarning* will be enabled when the car passes the EWL. In addition, CNT2 is modelled by defining a constant *minSpeed* and adding *grd1* in Figure 4a.



(a) Abstract level (b) Refinement level

Figure 4: Control event *IssueWarning*.

6.4 First Refinement

In this level of refinement we focus on the feature *status* which is a commanded phenomenon (CMN1). The process of identifying requirements corresponding to this phenomenon is similar to the process mentioned in Section 6.2 for the controlled phenomenon *warning*.

Since no other phenomenon is interrelated to *status*, at this level of refinement only *status*

is introduced. Also, requirement CNT2 is added to the model at this level, as it represents a restriction imposed by the phenomenon *status* on the controller.

This phenomenon is modelled by defining the variable *status* and command events *SwitchOn* and *SwitchOff* which set *status* variable from OFF to ON and vice versa (CMN1). Also, CNT2 is modelled by adding *grd5* to the control event *IssueWarning* as shown in Figure 4b.

7 Stage 3: Revision of RD and Model of LDWS

Modelling requirements formally helped us to find some of the missing and ambiguous requirements. So, based on Section 3.3, in Stage 3 we revise the RD and the model of the LDWS.

7.1 Missing Requirements and Revision of the RD

Two of the identified missing requirements are discussed in this section. The first is that we realised requirements related to the situations where a car *has crossed* the actual lane boundary and is travelling on the boundary should be differentiated from when the car has crossed the EWL and therefore is *about to cross* the boundary. As shown in Figure 5, this is mainly because of limitations of the camera's field of view which can result in detection of only one boundary.

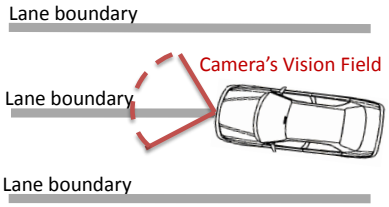


Figure 5: Limited field of vision for a camera when car is travelling on lane boundary.

We retake the structuring steps of Stage 1, Section 3.1, to add relevant requirements to the structured RD. Firstly, the LDWS should detect that the car has crossed the boundary. Thus, the requirement MNR4 is added to the RD. Secondly, the LDWS should issue warnings if crossing boundary is detected. This resulted in producing requirement CNT3, Table 3.

The other missing requirements are about the situations under which the LDWS should stop the warning process. Examining the LDWS showed that the process of issuing warnings should finish when the driver steers away from the lane boundaries and thus from the EWL. Here, the car is back within the lane and any issued warning should be stopped. Taking the structuring steps resulted in adding requirement CNT4 to the RD. Another situation where warning should stop is if the LDWS is switched off by the driver. This requirement is shown as CMN5 in Table 3.

7.2 Adjusting Abstract Model

To identify which of the newly found requirements of Table 3 need to be added to the abstract model, we retake the requirement selection process of Stage 2. As the phenomenon *warning* was modelled in the abstract level, we need to consider whether any of the new requirements are

Table 3: Third RD - Requirements are added based on Formal Modelling.

Req ID	Requirement Description
MNR4	LDWS shall detect when the car has crossed the lane boundaries and is travelling on the boundary.
CMN5	If LDWS is issuing the warning and the driver switches the system off, the warning signal would be stopped.
CNT3	In addition to (CNT1), LDWS issues warning when it detects that the car has crossed the lane boundaries and is travelling on the boundary (MNR4).
CNT4	LDWS would stop the warning signal when the system is performing (CNT2) and the warning signal has been issued but the driver steers away from the boundaries and therefore the car remains within the no warning zone.

related to this phenomenon. The examination of the new RD shows that requirements **CNT3** and **CNT4** need to be introduced at this level. In addition, **MNR4** should be modelled here, because the monitored phenomenon *crossing lane* is required for modelling CNT3.

Therefore, the abstract model is modified by defining *crossingLane* as a Boolean monitored variable which is TRUE if the car has crossed the lane boundary. CNT3 shows that there are two cases where the LDWS should decide on issuing warnings. Firstly when the car is *about* to cross the lane boundaries because it has crossed the EWL (CNT1). Secondly, when the car *has* crossed the lane boundary (CNT3). These are modelled in the two control events, *IssueWarning_CloseToBound* and *IssueWarning_CrossingLane* respectively, Figure 6. The requirement CNT4 is modelled by introducing the new control event *FinishWarning*. At this level of abstraction this event has a guard as $warning = TRUE$ and the action $warning := FALSE$.

```
event
IssueWarning_CrossingLane
 where
  @grd1 speed ≥ minSpeed
  @grd2 warning = FALSE
  @grd3 crossingLane = TRUE
  @grd4 status = ON
 then
  @act1 warning := TRUE
end
```

```
event IssueWarning_CloseToBound
 where
  @grd1 speed ≥ minSpeed
  @grd2 offset ∈ offset_LB · · offset_UB
  @grd3 carWidth + carPosition ≥
      laneWidth – EWL_Func (speed ↦ offset)
  @grd4 warning = FALSE
  @grd5 crossingLane = FALSE
  @grd6 status = ON
 then
  @act1 warning := TRUE
end
```

Figure 6: Control events *IssueWarning_CloseToBound* and *IssueWarning_CrossingLane*

7.3 Adjusting First level of refinement

As mentioned, requirements of the commanded phenomenon *status* were introduced in the first refinement level. Examining the new requirements of Table 3 show that CMN5 needs to be modelled at this level. This is done by defining two *SwitchOff* events. The event *SwitchOff1* sets

variables *status* to OFF and *warning* to FALSE, if it was previously TRUE, while *SwitchOff2* sets *status* to OFF and has the guard *warning* = *FALSE*.

8 Validation of the Model

Validation of the model against its RD can be done by adding a *validation column* to the right hand side of the structured requirement tables. Every requirement is then validated by adding the elements of the model, such as an event or a variable, which represent that requirement to its validation column. Also as quick reference, we refer to the level in which the requirement was modelled just after the ID of the requirement. Table 4 shows validation of some of the LDWS requirements against the model. For instance, requirement CNT1 is modelled in the abstract level using the following elements:

1. the controlled variable *warning*;
2. the control event *IssueWarning_CloseToBound*;
3. and showing that the car has entered the warning zone through the guard $carWidth + carPosition \geq laneWidth - EWL\_Func(speed \mapsto offset)$ in the control event *IssueWarning_CloseToBound*.

It is important to mention that the process of validation should take place at the end of every modelling step. This means as well as modelling RD, validation should be done incrementally. Thus, if a requirement is modified, the model and the validation column both should be updated to keep them consistent with the RD. Also, not always the entire RD is modelled, which means the validation column may only contain the reason for not modelling the requirement rather than the elements of the model. This is the case for the requirement MNR2, Table 4.

9 Related Work

In this section we look at some related works and compare them to our proposed approach.

9.1 Concretization and Formalization of Requirements

[FHP+05] has provided some guidelines for concretization and formalization of requirements of embedded systems. In formalization part, four steps have been suggested. *Identification* to produce an RD; *Normalization* to construct a glossary of terms that requirements use; *Structuring* to organise RD based on their "contents in a taxonomy", such as car speed; *Formalization* to formalise the structure, behaviour, interaction and data of the system.

In [FHP+05] grouping requirements is based on "different aspects", while we represent guidelines based on MCC phenomena for an engineer. In addition, our proposed approach allows one to revise the structured RD and construct a formal model incrementally. This means identified missing/ambiguous requirements can be addressed in the forthcoming iterations, while [FHP+05] does not tackle this issue. Also, we proposed a way for layering requirements which facilitates the use of refinement-based modelling, while this is not specified in [FHP+05].

Table 4: Validation of the requirements against the model.

Req ID	Requirement Description	Validation Column
MNR1 ABST	LDWS shall detect the EWL and vehicle position relative to visible lane boundaries based on the lane width and car width.	Monitored variable: *carPosition* & *laneWidth*; Constant: *carWidth*; Environment event: *UpdateCarPosition* & *UpdateLaneWidth*.
MNR2	LDWS shall track lane boundaries where lane markings are clearly visible in daylight (sunny/cloudy), night times and twilight (sunrise/sunset) lighting conditions.	Not Considered, since we are not concerned with requirements related to the performance of the camera and image processing unit.
CMN1 Refine	LDWS can be switched on and off by the driver through a single button.	Set: STATUS= {ON,OFF}; Commended variable: *status*; Command event: *SwitchOn* & *SwitchOff*.
CMN2 ABST	The driver can set the offset they want to have from either side of the lane boundaries through two buttons which are responsible for increasing and decreasing the distance.	Commanded variable: *offset*; Command event: *DecreaseOffset* & *IncreaseOffset*.
CNT1 ABST	LDWS would issue a warning when the vehicle has left the no warning zone (has crossed the EWL), and is entering the warning zone.	Controlled variable: *warning*; Control event: *IssueWarning_CloseToBound*; Guard: $carWidth + carPosition \geq laneWidth - EWL\_Func(speed \mapsto offset)$.

9.2 HJJ

In the HJJ approach [HJJ03] the specification of control systems is initially based on the system view rather than the software view. In this respect our approach has a similarity to HJJ. In the HJJ approach the focus is to model the environment and requirements while also the properties that the control system relies on are captured as "rely condition".

In HJJ all requirements of a system are dealt with in one step of specification, while our approach uses refinement. Also, we differentiate between monitored, commanded and controlled phenomena which assists with structuring the RD and mapping it to a formal model.

9.3 Requirement Tracing based on WRSPM

[JHLR10] introduces an approach for tracing requirements to an Event-B formal model. This approach is based on WRSPM [GGJZ00] (World, Requirement, Specification, Program and Machine) which distinguishes between phenomena, system's state space, and artifacts which represent constraints. The method of [JHLR10] for traceability involves taking a requirement and identifying phenomena and artifacts of the environment and system for that requirement. The

identified phenomena and artifacts are then modelled and traceability information is provided.

Both our approach and [JHLR10] are concerned with formal modelling of informal requirements and providing traceability. While [JHLR10] is more focused on traceability between an RD and the formal model, the approach represented in this paper is more on structuring and formal modelling. Also, we provide some guiding steps for layering requirements for a refinement-based modelling, while this is not specified in [JHLR10].

9.4 SCR

Our approach shares features with the SCR (Software Cost Reduction) method [HJL96]. SCR is a formal method for specification of control systems with a tabular notation.

Like our approach, SCR is based on the four-variable model. In addition to these four variables, the SCR uses mode classes (the system states), conditions (predicates of system states), and events (represent changes in system variables and mode). The SCR method does not have commanded phenomena though these can be represented as monitored variables.

Our experience is that distinguishing monitored and commanded phenomenon facilitates requirements elicitation as they serve distinct roles. SCR is more a specification method, while in this paper we focus on structuring the RD as well as specification and traceability.

In SCR an engineer is required to identify the monitored, controlled, input and output variables of the system. However, we have a system-level view on the behaviour of the controller. Therefore, we focus on monitored, controlled and commanded variables in the more abstract models and introduce input and output variables in refinement levels.

10 Future Work and Limitations

This approach can be improved and developed further by experimenting its application in other case studies. Also, the current approach focuses on the functional RD and we are yet to deal with some of the challenges presented by non-functional requirements. In addition to improving the approach, part of our future work involves developing a more complete formal model of the LDWS. Examples of requirements which can be considered in the future work are timing and fault tolerance requirements. Our other future work involves evolution of the passive LDWS to an active lane centring system and examining the evolution of the requirement document in this case.

11 Conclusion

In this paper we discussed the evolution of the requirement document of the LDWS through domain expert's feedback and modelling using Event-B. The MCC modelling guidelines [But09] inspired us to structure the requirement document based on monitored, controlled and commanded phenomena. Also in this paper, some criteria for layering the requirements and mapping informal requirements to a formal model were provided. We followed the approach proposed in the paper to structure and model the RD of an LDWS. Some of the advantages provided by following the proposed approach are:

- Improving the requirement document by gathering and structuring the requirements incrementally in iterations.
- Facilitating the process of validation of the model against the requirement document and therefore helping with traceability between the model and requirements.
- Traceability enables us to maintain the requirement document and the model consistent in an easier and more manageable style.

The process of formal modelling of the LDWS also helped to identify missing requirements. We structured the newly identified requirements and revised the RD to accommodate them. These changes were also applied to the model. This shows that in order to achieve a more accurate requirement document and formal model the process of structuring requirements and modelling needs to be iterated.

We believe that the proposed approach can facilitate formal modelling of control systems and it can be used for modelling a structured RD using any refinement-based formal language. Furthermore, it is possible to use the MCC approach for organising requirement documents without modelling them formally.

Acknowledgment

This work is supported by GM India Science Lab and partly by the EU research project ICT 214158 DEPLOY (Industrial deployment of system engineering methods providing high dependability and productivity) www.deploy-project.eu.

We would like to thank Manoranjan Satpathy and S. Ramesh (GM India Science Lab) and also Abdolbaghi Rezazadeh and Reza Sarshogh for their helpful advice.

Bibliography

[ABH+10] J.-R. Abrial, M. Butler, S. Hallerstede, T. Hoang, F. Mehta, L. Voisin. Rodin: an open toolset for modelling and reasoning in Event-B. *International Journal on Software Tools for Technology Transfer (STTT)* 12:447–466, 2010.

[Abr10] J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.

[But09] M. Butler. Chapter 8: Modelling Guidelines for Discrete Control Systems, Deploy Deliverable D15, D6.1 Advances in Methods Public Document. http://www.deploy-project.eu/pdf/D15-D6.1-Advances-in-Methodological-WPs..pdf, 2009. cited 2011 Jan.

[Fed05] Federal Motor Carrier Safety Administration. Concept of Operations and Voluntary Operational Requirements for Lane Departure Warning Systems (LDWS) On-board Commercial Motor Vehicles. http://www.fmcsa.dot.gov/facts-research/research-technology/report/lane-departure-warning-systems.htm, 2005. cited 2011 Jan.

[FHP+05] A. Fleischmann, J. Hartmann, C. Pfaller, M. Rappl, S. Rittmann, D. Wild. Concretization and Formalization of Requirements for Automotive Embedded Software Systems Development. In *The Tenth Australian Workshop on Requirements Engineering (AWRE)*. Pp. 60–65. 2005.

[GGJZ00] C. Gunter, E. Gunter, M. Jackson, P. Zave. A reference model for requirements and specifications. *Software, IEEE* 17(3):37 –43, 2000.

[HJJ03] I. Hayes, M. Jackson, C. Jones. Determining the specification of a control system from that of its environment. In *FME 2003: Formal Methods*. Lecture Notes in Computer Science 2805, pp. 154–169. Springer Verlag, 2003.

[HJL96] C. L. Heitmeyer, R. D. Jeffords, B. G. Labaw. Automated consistency checking of requirements specifications. *ACM Trans. Softw. Eng. Methodol.* 5:231–261, July 1996.

[Jac01] M. Jackson. *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[JHLR10] M. Jastram, S. Hallerstede, M. Leuschel, A. Russo. An Approach of Requirements Tracing in Formal Refinement. In *Verified Software: Theories, Tools, Experiments*. Lecture Notes in Computer Science 6217, pp. 97–111. Springer Berlin / Heidelberg, 2010.

[PM95] D. L. Parnas, J. Madey. Functional Documents for Computer Systems. *Sci. Comput. Program.* 25(1):41–61, 1995.

[PMGB05] A. Polychronopoulos, N. Mhler, S. Ghosh, A. Beutner. System Design of a Situation Adaptive Lane Keeping Support System, the SAFELANE System. In *Advanced Microsystems for Automotive Applications 2005*. Pp. 169–183. Springer Berlin Heidelberg, 2005.

[RME00] R. Risack, N. Mohler, W. Enkelmann. A video-based lane keeping assistant. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. Pp. 356 –361. 2000.

[Win90] J. M. Wing. A Specifier's Introduction to Formal Methods. *Computer* 23(9):8–24, 1990.

[YBR10] S. Yeganefard, M. Butler, A. Rezazadeh. Evaluation of a Guideline by Formal Modelling of Cruise Control System in Event-B. In *Proceedings of the Second NASA Formal Methods Symposium (NFM 2010), NASA/CP-2010-216215*. Pp. 182–191. April 2010.