Control Systems: Phenomena and Structuring Functional Requirement Documents

Sanaz Yeganefard, Michael Butler
Electronics and Computer Science
University of Southampton, Highfield
Southampton, UK, SO17 1BJ
Emails: sy2g08, mjb@ecs.soton.ac.uk

***Abstract*—Influenced by the Parnas and Madey's four-variable model and the concept of phenomena in problem frames, we desire to provide guidelines to facilitate refinement-based formal modelling. These guidelines are based on monitored, controlled, mode and commanded (MCMC) phenomena of a control system. Commanded phenomena reflect the role that an operator plays in system control. The mode phenomenon captures the states of the controller.**

Requirements of several case studies have been formally modelled using the MCMC phenomena. This helped to identify some of the ambiguities and advantages of the guidelines. In particular, we realised that the concept of commanded phenomena and its difference with monitored phenomena can cause confusion. Also, it was noticed that the mode is a special phenomenon, as it can be modified by operator requests or internally by the control system.

In this paper we clarify the concept of commanded phenomena and differentiate between monitored and commanded phenomena clearly. The concept of mode phenomenon is also introduced in details. As practical examples, the phenomena of two case studies, namely a cruise control system and a lane centering controller (LCC), are identified.

The MCMC phenomena are also used to structure the requirement document (RD) of a control system. This can help with the transition from an informal RD to a formal model. This approach is used to structure the RD of the LCC case study which is supported by our industrial partner.

***Keywords*-formal modelling guideline; structuring requirement; lane centering system;**

I. Introduction

Continual interactions of control systems with their evolving environment makes these systems complex and as with any complex system, constructing and structuring functional requirement documents (RD) and the design of control systems can be a difficult process. In addition, their requirements might be ambiguous and incomplete. However, these systems are usually used in life critical situations which means having a comprehensive and well-structured RD is essential. Also, to help with improvement of system reliability and safety, a systematic and rigorous design process is needed.

Specifying and developing systems using formal methods (mathematical based techniques) is known to improve understanding of a system [1]. Formal methods can help to identify missing and ambiguous requirements. Moreover, formal modelling supports formal verification of system properties. However, one of the difficulties of using rigorous formal modelling is formalising an informal RD.

Patterns and guidelines can be used to facilitate the transition between an informal RD and its formal representation. Influenced by Parnas' four-variable [2], we propose a set of guidelines to model a control system using its monitored, controlled, mode and commanded (MCMC) phenomena. Furthermore, we describe an approach to structuring the functional requirement document (RD) according to the MCMC phenomena. The term phenomena is influenced by Jackson's problem frames approach [3].

The focus of this paper is mostly on the definition and identification of the MCMC phenomena as well as structuring the RD of a control system accordingly. The approach presented here has evolved mainly from our previous works on:

- monitored, controlled and commanded (MCC) modelling guidelines [4],
- the MCC requirement structuring approach [5],
- the application of the above to several case studies.

The focus of the *MCC modelling guidelines* [4] is on formal modelling of control systems using monitored, controlled and commanded (MCC) phenomena. These guidelines were followed in formal modelling of case studies, such as a cruise control system [6] and a FEDAC system [7]. This resulted in the realisation of the lack of guidance on mapping requirements to the formal model and the identification of system phenomena.

The *MCC requirement structuring approach* [5] was proposed to fill in the gap between requirements and their formal representation. Here we proposed to structure requirements based on the MCC phenomena and then formalise the structured RD. This approach was followed to structure the requirements and formally model a lane departure warning system [5] and a cruise control system. These case studies showed that the process of formalisation can be simplified as the result of structuring RD.

Applying the MCC approaches to several case studies has helped us to notice the special phenomenon of *mode* which represents the state of a control system. One contribution of this paper is the detailed explanation of this special phenomenon, and the evolution of the MCC guidelines to MCMC, which includes the mode phenomenon.

Application of the guidelines to case studies has also shown that the identification of phenomena can be difficult. As our ultimate goal is to propose formal modelling guidelines, it is essential to have appropriate phenomena

which can also be easily identified. One main difficulty is the concept of commanded phenomena which needs further clarifications, as it can cause confusion. The other contribution of this paper is to explain how to identify the MCMC phenomena in detail and through examples. In particular we clarify why to use and what is meant by commanded phenomena.

Finally, we contribute by evolving the MCC requirement structuring approach [5] to an MCMC structuring approach, where the phenomena, their identification and the proposed structuring steps are unambiguous.

As a practical example, the phenomena of a real automotive lane centering controller (LCC) are identified and its RD is structured accordingly. This case study has been supported by industrial partners. In addition to this, the MCMC phenomena of a cruise control system are identified, although this system is not explained in detail and it is merely used to clarify the MCMC phenomena.

II. Phenomena in Problem Frames

The problem frame approach (PF) [3] distinguishes between the problem and the solution with the aim to focus on the problem domain in requirement analysis.

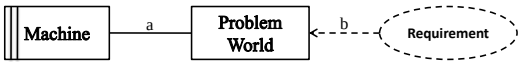

Figure 1. A simple problem diagram.

Figure 1 represents a simple problem frame. The *requirement* represents the system's goal. More specifically, it describes the behaviour expected from the *problem world*, where the problem is located. The satisfaction of the requirements are to be ensured by the *machine*, which is the software to be built. Every rectangle in Figure 1 represents a domain and their communication is shown using *shared phenomena*. For instance, *a* is the share phenomena which represents interactions between the machine and the problem world. PF defines six types of phenomena. In this paper two phenomena, namely *events* and *entities* are used. Also we refer to the entities as variable phenomena.

III. Summary of the MCMC Phenomena

Our ultimate aim is to provide guidelines on formal modelling of an informal RD. To do this an approach influenced by the four-variable model [2] is suggested. The initial step of the approach is to identify the system phenomena. These are *monitored*, *controlled* and *mode* phenomena for an *autonomous controller* which consists of a plant and a controller.

The other form of control systems is a *commanded controller* which involves an operator who can send commands to the controller (Figure 2). For such systems, we propose the identification of *monitored*, *controlled*, *mode* and *commanded* (MCMC) phenomena.

As mentioned a phenomenon can be of type variable or event. The definitions of MCMC variable and event

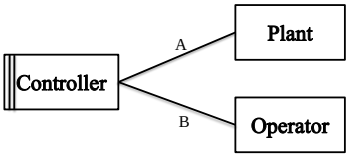

Figure 2. A commanded control system.

phenomena are given below, while they are discussed in detail in the remainder of this paper.

- Monitored phenomena:
  1) *Monitored variables* whose values are determined by the environment/plant.
  2) *Environment events* update monitored variables.
- Controlled phenomena:
  1) *Controlled variables* representing phenomena in the environment whose values are set by the controller.
  2) *Control events* update controlled variables.
- Mode phenomenon:
  1) *Mode variable* represents the controller mode.
  2) Mode events which cause mode change can be triggered by the operator or the controller:
     a) *Operator mode events*, such as switch on.
     b) *Controller mode events* update the mode based on the received monitored variables.
- Commanded phenomena:
  1) *Commanded variables* whose values are determined by the operator and that influence controlled and mode phenomena.
  2) *Commanded events* are operator requests to modify commanded variables.

The identification of the MCMC phenomena are based on firstly the system requirements and the domain study and secondly the given definitions of these phenomena. Notice that for the mode phenomenon we suggest the identification of its possible values. In other words, identifying the possible modes of the control system. As an example we briefly discuss this initial step for a cruise control system (CCS).

A CCS receives the *actual speed* of the car from the environment and the *target speed* from the driver. Its role is to minimise the difference between the actual and target speed using the received information. This is done by setting the *acceleration* of the car via a speed regulation mechanism [8]. Some of the requirements of the CCS are demonstrated in Table I [8].

The monitored, controlled and commanded phenomena (variable and event phenomena) of the CCS and its modes are shown in Table II.

IV. Commanded phenomena

In this section, the concept of commanded phenomena is explained in more detail. Firstly, we discuss why it is useful to have commanded phenomena. After that, the ambiguity between commanded and monitored phenomena is explained. This is clarified in Section IV-C.

Table I
SOME REQUIREMENTS OF A CCS (BASED ON [8]).

1	The cruise control system can be switched `ON` or `OFF` by the driver. When the cruise control has just been switched on, the target speed is not defined.
2	When CC is on, it will be `activated` as soon as the driver sets the target speed.
3	Once the cruise control is switched on and the target speed is defined, the speed regulation mechanism is automatically invoked.
4	Once the cruise control is started and the target speed is defined and it is `active`, if the driver uses the brake pedal, the speed regulation mechanism will be `suspended`.

Table II
MCMC PHENOMENA OF CCS.

Monitored Variable	**Environment event**
actual speed	Update actual speed
Controlled Variable	**Control Event**
acceleration	Update acceleration
Commanded Variable	**Command Event**
target speed	Set target speed
Values of Mode	**Mode Event**
`on, off, active, suspend`	Switch on (operator triggered)

A. How Commanded Phenomena Helps

The special role of the operator of a commanded controller is captured as *commanded* phenomena whose values are to be maintained and stored by the controller. The benefit of identifying commanded phenomena might be questionable for the readers as it might seem possible to capture operator commands as *monitored* phenomena. This view is mainly the result of perceiving the system from the controller (or the software to be implemented) point of view, where any input to the system is treated as a monitored quantity.

However, a system-level view where inputs are differentiated and operator actions are taken into account can provide a better understanding and a more accurate perception of the controller behaviour. Furthermore, requirements should be considered in the system-level view, as they present details of the controller as well as its environment which may also consists of operators. In addition, conducting several case studies has shown that engineers can benefit from the distinction between monitored and commanded phenomena, as they have differences:

1) Monitored phenomena are part of the environment which is usually *predictable*. However, commanded phenomena involve people which are *unpredictable*[1].
2) A controller receives values of monitored phenomena *periodically* and through means such as sensors. For instance, a CCS receives the car speed every x milliseconds through the speed sensors. However, operator commands are *sporadic* requests sent through a user interface, such as a button. In a CCS, a driver can set the target speed at any point after switching on the CCS through a set button.
3) Usually values of commanded phenomena have to be preserved by the controller, while values of monitored phenomena are received at every controlled cycle and responded to by the controller. This means that a controller is not required to maintain values of monitored phenomena when they have been responded to.

[1]PF distinguishes between these as causal and biddable domains [3].

The first difference shows that there should be minimal assumptions on operator behaviours, or even possibly none. For instance, it cannot be assumed that an operator will switch off the controller as soon as they are notified of a fault in the system. Therefore, to achieve a complete and well-behaved system (a system with no surprising results) the requirements and the design of the system should include situations where the operator does not follow the expected routine.

The second difference shows that most likely the user interface should be designed according to the possible operator requests. Whereas it is more likely that sensors which are used for transferring values of monitored phenomena will be configured or modified to fit their purposes, rather than designed. We recognise that some controllers may require specialised sensors which should be designed according to their specification. However, this discussion is avoided as requirements related to design of sensors is out of the scope of our current work.

The third difference is closely related to the second. As an example the target speed in the CCS can be increased or decreased by sporadic user requests. This means while the controller is active, it should maintain the value of the target speed so it can be increased or decreased according to the operator request. In contrast, monitored phenomena are sensed at the start of every control cycle and stored until they are responded to by the controller.

B. Commanded Phenomena are not Clearly Explained

As mentioned, in order to model a system using the MCMC guidelines, an engineer is required to use their judgment to identify the monitored, controlled and commanded phenomena as well as possible modes of the controller. While the identification and distinction between monitored, controlled and mode phenomena are usually straightforward, identifying commanded phenomena and sometimes distinguishing them from monitored phenomena can be more challenging. We use the example of the CCS to show this problem.

Commanded phenomena are defined as "shared phenomena between a controller and the operator". Based on this definition and using the requirement 2 and 4 of Table I, one can identify *target speed* and *brake* pedal as commanded phenomena of the CCS. However, after modelling and examining these phenomena, we realised that the behaviour of *brake* and *target speed* are different.

The difference between the brake and the target speed is that the latter is defined specifically to serve the CCS, while the former is a generic phenomenon. This is because the primary role of the brake pedal is to reduce the car speed, while its secondary role is to suspend the CCS. In other words, the brake is part of the plant (the given environment), whereas a user interface (such as buttons) is going to be implemented to allow interactions between the driver and the target speed.

This distinct characteristic can cause potential confusion. Since it is possible to treat the brake as either a commanded phenomenon, because it is a form of interaction between the operator and the controller, or a monitored phenomenon, because it is part of the environment. This confusion at the very first step of the modelling guidelines, can be great enough for some to avoid using the guidelines completely. In addition, the type of phenomena can affect the modelling process, particularly when the implementation details are modelled. In the next section we provide guidelines that help to clarify this distinction.

C. Distinguishing Monitored and Commanded Phenomena

The conducted case studies have shown that one way of differentiating between the monitored and commanded phenomena is to distinguish between their user interfaces (UI). We categorise the UI of a control system as:

- *Existing interface:* The UI whose primary role is to serve the given environment. The existence of such UI does **not** rely on the existence of the control system. For instance pedals in cars exist even without a CCS. Another example is the indicator lever whose primary role is to inform other road users of the driver's intention. However, a lane departure warning system receives data from the indicator to determine intentional lane departure.
- *Specialised interface:* The UI whose primary role is to serve the controller. This UI will exist only when its corresponding control system is implemented, such as an increase or a decrease button in a CCS which changes the value of the target speed.

Another difference between these two UI is in their connection with a control system. An *existing interface* may be connected to one or more controllers in a plant. Also this type of UI usually broadcasts user interactions as messages which can be received by any controller connected to this UI. For instance, a brake pedal can be connected to a CCS and an anti-lock braking system (ABS). In this case, if the brake is pressed, the message will be broadcast and then received by both systems. However, a *specialised interface* is usually designed for and directly connected to one control system.

The case studies that we have considered showed that in automotive systems an *existing interface* is connected to controller(s) via a Controller Area Network (CAN bus) which uses broadcasting mechanism to transfer data. However, a *specialised interface* is usually connected to its corresponding control system using the Local Interconnect Network (LIN bus).

We use the distinction between an existing and specialised interface to distinguish monitored and commanded phenomena. Phenomena whose values are determined by the plant/environment including *existing interface* are monitored phenomena. Whereas, phenomena whose values are determined by the operator via *specialised interfaces* are commanded phenomena. Based on this in the CCS, the brake pedal is a monitored phenomenon, while the target speed is a commanded phenomenon.

V. Mode Phenomenon

Experimenting with requirements and models of the conducted case studies showed that these systems usually have one particular phenomenon in common. This phenomenon which is called **mode** represents the overall state of the controller. Mode is also a special type of phenomenon, since both the *controller* and *operator interactions* can update its value. For instance, in the example of the CCS, the driver can switch the CCS `on` or `off`. In addition, the controller itself can update the mode to `suspend`, when the brake pedal is pressed or as soon as the actual speed goes below a certain threshold. Because of this characteristic, separating the mode from commanded and controlled phenomena can greatly clarify the process of identifying phenomena and modelling.

We define two types of mode event phenomena. These are *operator* and *controller mode event phenomena* which are triggered/performed by the operator or by the controller respectively. Differentiating between these two events is important, since their modelling processes are different. The decision of triggering an *operator mode event* takes place externally, whereas this decision is internal for a *controller mode event* (by the controller based on monitoring the environment). In addition, an *operator mode event* requires a UI, most likely a *specialised interface*, to be implemented to provide means of interactions for the operator. Therefore, these two events impose different requirements on the system. For instance, some issues of the operator mode events of switch on and off in a CCS that should be considered in the RD are:

- What kind of interface to define? A toggle button or two separate on and off buttons?
- What should happen if the conditions for switching the system on does not hold, but the driver requests to switch the CCS on?
- What is the priority of responding to the switch button in comparison to other buttons?

Another characteristic of the mode phenomenon is that it can be used, especially in modelling, to deal with faults and errors of the system. When an error is detected, the control system changes its mode, for instance to a `recovery` or a `fault` state. This way the controller avoids performing its normal role. An example of this is shown in Section VIII-A.

VI. Structuring RD Using MCMC Phenomena

As mentioned, the very first step of the MCMC modelling guidelines is to identify the *monitored*, *controlled* and *commanded* phenomena of the control system as well as its possible modes and transitions among them. This raised the subject of connecting phenomena of a control system to its RD. In [5] we explored this possibility by structuring requirements according to their monitored, controlled and commanded phenomena. However, the proposed structuring steps of [5] require changes.

- Firstly, the structuring steps are to be adjusted and clarified, since the definition of commanded and monitored phenomena have been clarified.
- Secondly, the phenomenon mode which is added to the new guidelines is to be represented in the RD structuring approach.

Notice that it is assumed that the textual requirements exist. The newly proposed approach for structuring a textual RD is to divide the requirements into monitored, controlled, mode and commanded sections, as shown in Figure 3. Requirements will be placed into the appropriate section, according to the phenomenon they represent. There is also a revision step to help an engineer to seek the most obvious MCMC phenomena in the initial structuring step and improve the structured RD incrementally.

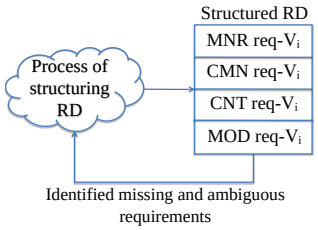

Figure 3. Structuring RD based on MCMC phenomena.

The structuring steps are based on the *variable phenomena*, in contrast to event phenomena. However, listing the commanded and mode event phenomena (especially operator mode events) is recommended, as they represent the *specialised interface* and can give us a better understanding of the requirements. The following shows the details of the proposed structuring steps into MCMC sections.

1) *List the system's monitored and controlled phenomena.* It suffices to list variable phenomena, such as `actual speed` in CCS.
2) *List the system's commanded phenomena.* It is useful to list both variable and event phenomena, respective examples are `target speed` and `increase target speed` in CCS.
3) *List the values of the mode phenomenon and their transitions.* One simple way is to use a state machine diagram. Examples of mode values are `on`, `off`, and `suspended` in CCS. To identify their transitions it is helpful to determine the operator and controller mode events phenomena, respective examples are `switch on` and `suspend`.
4) *Organise RD into four sections: monitored (MNR), commanded (CMN), controlled (CNT) and mode (MOD).* It is suggested to take every requirement and based on the list of phenomena determine the **main variable phenomenon** that the requirement refers to. The main variable phenomenon is the one which the requirement represents HOW it is being modified/updated.
   Requirements are then to be placed in the section which represents the type of their main variable phenomena. As an example the requirement "When CC is on, it will be activated as soon as the driver sets the target speed" in the CCS defines changes of the mode phenomenon (from `on` to `active`) under certain conditions. Thus, we place this requirement in the MOD section. Another example is the requirement "When the CCS is active and the actual speed is below the target speed then the acceleration should be increased according to the increase-speed control law". This defines a modification of the controlled variable *acceleration* so we place it in the CNT section.
5) *Add unique ID labels.* Every ID starts with the section the requirement belongs to (i.e. MNR, CMN, CNT or MOD), followed by a unique number.
6) *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 add the requirement to an appropriate section.

Notice that requirements of the monitored section represent the assumptions of the environment. This structuring approach is explained further through the LCC case study.

VII. Overview of Formalising Structured RD

In this section, an overview of our ultimate goal, which is formalising the structured RD, is briefly explained (though note that formalisation is not the main topic of this paper).

The formal modelling guidelines suggest that each variable and event phenomena be modelled with a corresponding formal variable and events. The formal language we use is Event-B [9] which consists of variables and guarded actions (called events). This language is mainly chosen because of the simplicity of its notation, its support for incremental refinement-based development and because of the availability of a modelling tool and provers.

Even-B also supports refinements, where new events can be added to the model by refining a skip event in the abstract model. This allows us to gradually add behaviours to the model and elaborate it with requirements which were hidden in abstract levels.

To model the structured RD, we suggest starting with the main behaviour of the control system which is represented as a controlled phenomenon. After modelling this phenomenon and its related requirements, we then model the remaining phenomena using refinement. For instance, the CCS was modelled in several refinement levels. We

firstly modelled the *acceleration* phenomenon. At this level it was also necessary to model the *actual* and *target speed*, as their values determine the acceleration. In the second level we modelled the *mode* phenomenon and its transitions. At this level it was also necessary to model the *brake pedal* and *switch on/off*, as they affect the value of the mode. Notice that models of monitored phenomena are loose as they represent the environment.

VIII. Applying MCMC to LCC

In this section, the case study of a lane centering controller (LCC) which is supported by our industrial partner is discussed. Section VIII-A gives an overview of this system. Section VIII-B describes the process of writing its requirements. In Section VIII-C the MCMC phenomena of the LCC are identified and its RD is structured based on the presented requirement structuring approach. Part of the formal model of the LCC is shown in Section VIII-D.

A. Overview of LCC

An LCC is responsible for "automated lane-centering" (Notice that we do not consider lane-change manoeuvres). This system aims to maintain a vehicle either on the centre line or on a line close to the centre (at an offset chosen by the driver) within a lane [10]. As shown in Figure 4, the path generator provides *predicted* and *target* paths. The role of the LCC is to maintain the angle between these two path as close to zero as possible by calculate the steering correction angle accordingly.

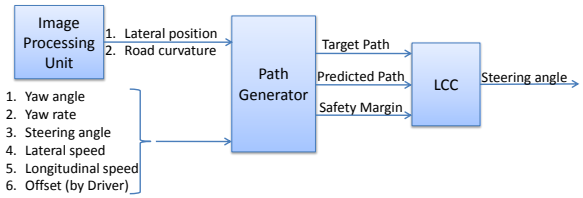

Figure 4. Overview of LCC inputs and output.

The predicted path is an estimate of the intended vehicle path for the near future. This is generated using the longitudinal speed, the lateral speed, the yaw angle, the yaw rate and the steering angle of the car. The predicted path also depends on the lateral position[2] of the car which is received from the image processing unit.

The target path[3] shows the path the vehicle should take to maintain its position within the lane. This path is determined base on the lateral position and the roadway curvature received from the image processing unit.

In addition, the target path can be influenced by the *offset* the driver wishes to have from the centre of the lane (central line). The offset is within a specific range, such as -2 to +2, where '-' represents the offset from the left of the central line and '+' represents the offset from the right. If the offset is set to 0, the central line will be used as the reference for target path. For instance Figure 5 shows that the offset is set to '-1'.

In addition to target path and predicted path, the path generator component provides the LCC with a *safety margin*. The safety margin represents the minimum time within which the path generator can provide the target and predicted path for the LCC. When the safety margin is x ($x > 0$), both paths can be determined for the next x milliseconds, while 0 means one (or both) path cannot be determined. This margin is necessary, as in the cases where the image processing unit fails to operate, the path generator can notify the LCC in advance by reducing the safety margin. In this case the LCC will issue warnings which means the driver should take over the responsibility of steering. Notice that this is an assistance system, and not a safety system, thus it is the driver's responsibility to resume steering as soon as they receive the warning.

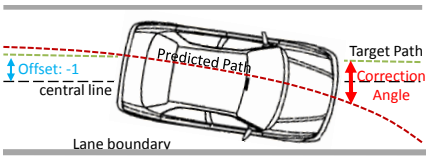

Figure 5. LCC actuates steering angle using target and predicted path.

Generally speaking an LCC is accompanied by an Adaptive Cruise Control (ACC) which is responsible for the longitudinal control of the car (the speed and the spacing from vehicles at the front), whereas the LCC controls the lateral car position (relevant to the lane boundaries) [12]. Here, we assume that the ACC exists and consider its communication with the LCC. For instance the activation of the LCC requires the ACC to be active. The LCC will issue warnings as soon as it detects faults and errors in ACC.

1) Interaction between User Interfaces and LCC: As mentioned, the UI is categorised into existing and specialised. The *existing UI* of the car which can override (or suspend) the LCC are:

- **Steering wheel** where the applied torque is above a certain threshold will override the LCC. The pressure on the steering wheel indicates that the driver is overriding the LCC, such as in an emergency. The defined torque threshold allows the LCC to detect this without any ambiguity.
- **Indicating** shows an intentional lane-change manoeuvre by the driver and will override the LCC.

The *specialised UI* of the LCC are defined using the requirements and with help of domain experts. These are:

- **Switching on or off:** The driver can switch the LCC on or off.
- **Setting offset:** The offset the driver wishes to have from the central line can be set by increasing or decreasing its value. Also there is a reset button which sets the offset to zero.
- **Resuming the LCC:** When the controller is overridden, such as when the indicator is used, the LCC can be resumed by the driver.

[2]Lateral position is sometimes referred to as "lateral displacement" [11].

[3]Target path is sometimes referred to as "desired path" [11].

B. Process of Writing LCC's RD

The RD of the LCC was evolved gradually. We first produced the requirements based on the *public domain*, such as [10] and [11], as well as our experience with a lane departure warning system (LDWS) [5]. This RD was changed as the result of feedback received from *domain experts*.

In comparison to other case studies, such as the CCS [6] and the LDWS [5], the LCC has less information available in the public domain. Thus the feedback we received influenced the RD considerably. As an example, the experience that we had with the LDWS was that this system warns the driver of an unintentional lane departure by using `lane boundaries` as its reference point. We used the same reference point for steering angle correction in the LCC. However, the feedback we received was that the reference point of the LCC is the `centre of the lane`.

In addition, some requirements were identified as a result of formally modelling the system in Event-B. For instance, the need for 'safety margin' which was discussed earlier was concluded during the modelling of the LCC.

C. Final RD of LCC

As mentioned the textual RD of the LCC was produced according to the public domain and the feedback from domain experts. In addition, this RD was evolved as a result of formal modelling. The requirements presented in this section are part of the final version of the LCC's RD.

To structure these requirements we take the MCMC structuring steps presented in Section VI. At first monitored variable phenomena of *target path*, *predicted path*, *safety margin*, *indicator*, *steer*, *status of ACC* and *error detection* were identified and listed. These phenomena and their corresponding requirements are shown in Table III.

As MNR8 shows we treat detection of faults and errors in sensors and actuators as monitored phenomena and we are not concerned with how they have been detected. The error handling and treatments of faults are discussed in the CNT section of the RD.

Table III
STRUCTURED REQUIREMENTS OF LCC - MONITORED SECTION.

Phen.	ID	Requirement Description
Target Path	MNR1	LCC shall receive the determined target path of the vehicle from the path generator.
Predicted Path	MNR2	LCC shall receive the determined predicted path of the vehicle from the path generator.
Safety Margin	MNR3	LCC shall receive the calculated safety margin from the path generator.
Indicator	MNR4	LCC monitors the indicator.
Steer	MNR5	LCC monitors forces applied to the steering by the driver.
	MNR6	LCC can monitor the steering only if the applied torque is above a minimal amount.
State of ACC	MNR7	LCC monitors the changes in the status of ACC.
Error detection	MNR8	LCC detects faults and errors of sensors and the steering actuator.

After this, *steering angle*, *display unit* and *warning* were identified and listed as the controlled variable phenomena, Table IV. While the steering angle represents the main output of the LCC, the display unit and warning phenomena are used to inform the driver of the LCC's mode.

Table IV
STRUCTURED REQUIREMENTS OF LCC - CONTROLLED SECTION

Phen.	ID	Requirement Description
Steering	CNT1	The role of LCC is to minimise the difference between the target and predicted paths by correcting the steering angle.
	CNT2	LCC will periodically correct the steering angle to an appropriate value within a specific range.
	CNT3	LCC will start its role as soon as it becomes `active`.
	CNT4	LCC will stop its role as soon as it is switched `off`, `standby`, `overridden`, not `available` or detects `error`.
Display	CNT5	The display unit may be green, yellow, red or off depending on the state of the system.
	CNT6	The display unit will be green when LCC is `active` and performs normally. It will turn yellow when LCC is `active` and is issuing warning. It will be red when LCC detects `errors`. It will be off when LCC is `off`, `standby`, `overridden` or not `available`.
Warning	CNT7	If LCC detects `errors` or if it is `active`, but the determined safety margin is below a certain threshold and above 0, LCC will issue warnings.
	CNT8	The warnings should stop as soon as LCC becomes `off`, `standby`, `overridden` or not `available`.

The only commanded variable phenomenon of the LCC is *offset*. Also, the events phenomena of INCREASE and DECREASE modify the *offset*. The requirements corresponding to these phenomena are shown in Table V.

Table V
STRUCTURED REQUIREMENTS OF LCC - COMMANDED SECTION

Phen.	ID	Requirement Description
Offset	CMN1	Driver can determine the offset they wish to have from the centre of the lane.
	CMN2	Driver chooses the value of offset through 2 buttons which INCREASE and DECREASE the offset by one unit every time they are pressed.
	CMN3	The offset is within a specific range.

To organise the mode section we identify and list the possible states of the LCC as well as the transitions between the states. An overview of the state machine is shown in Figure 6. As an example the transition from `active` to `error` state will be performed as soon as the LCC detects an error or fault in the system such as when the safety margin is 0.

Notice that the dashed arrow represents the assumption that the fault is to be resolved and the LCC is to be reset from the `error` to the `available` state manually. This manual reset is not discussed in this RD.

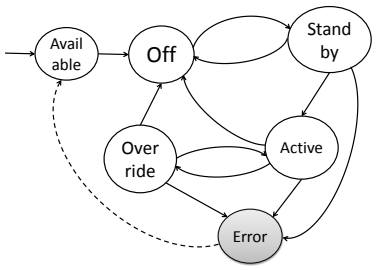

Figure 6. Possible mode changes in LCC.

We also list the operator mode events which can update the mode phenomenon. These are SWITCH ON, SWITCH OFF and RESUME. Requirements of the mode section are shown in Table VI.

Table VI
STRUCTURED REQUIREMENTS OF LCC - MODE SECTION

ID	Requirement Description
MOD1	LCC can be SWITCHED ON and OFF by the driver when it is `available`.
MOD2	If LCC is `available`, it will become `standby` as soon as it is SWITCHED ON, and will become `off` as soon as it is SWITCHED OFF.
MOD3	When LCC is `standby`, it will be `active` as soon as its starting conditions (not discussed in this paper) hold.
MOD4	When LCC is `active`, it will be `overridden` as soon as either the driver uses the vehicle's indicator or a steering torque above the threshold is detected.
MOD5	If LCC is `overridden`, it will be `active` when the RESUME button is pressed and the starting conditions hold.
MOD6	If the ACC is not in an active state, the LCC will not be `available`.
MOD7	LCC will be in an `error` state as soon as it detects errors or when the safety margin is 0.

D. Overview of Formalising LCC

The guidelines given in Section VII are used to model the LCC formally using the Event-B language. We also use refinement to introduce behaviours of the system gradually. In the most abstract level the main behaviour of the LCC (correcting steering angle - CNT1) is modelled. Thus a variable and an event are defined in the model, these are named as *steeringAngle* and *UpdateSteeringAngle* respectively.

However, the value of *steeringAngle* is determined according to the target and predicted paths (MNR1 and MNR2). Therefore, it is required to model these monitored phenomena at the same level. These are modelled as formal variables of *targetPath* and *predictPath* which can be updated by the environment events of *UpdateTargetPath* and *UpdatePredictedPath* respectively. In addition to these phenomena, we introduce the safety margin at this level. An overview of the event UpdateSteeringAngle is shown in Figure 7 using the Event-B formal language.

Notice that events in Event-B are atomic. Two clauses that are used here in events are **where** and **then**. The former represents guards which are predicates describing the conditions that must hold for the occurrence of the event. The latter represents the actions which determine how specific state variables change as a result of the occurrence of the event. Actions of an event will be performed simultaneously to preserve the atomic nature of the event.

```
event UpdateSteeringAngle
    where
      @grd1 safetyMargin > 0
    then
      @act1 steeringAngle ≔
     STEER_FUNC(targetPath ↦ predictPath)
  end
```

Figure 7. Updating steering angle event in abstract level.

After this, the rest of the RD was introduced to the model gradually using refinement steps. As an example, the *mode* phenomenon was introduced at the first level of refinement by defining a variable named *cnt_status* and mode events such as *SwitchOn* and *Activate* which update the mode. After this, the controlled variable phenomenon of *warning* was modelled. At this level we defined the formal variable *warning* and its corresponding events. One of these events, shown in Figure 8, represents the requirement that the "LCC will set the warning to true as soon as it detects that the safety margin is below the certain threshold" (CNT7).

```
event SetWarning_LowSafetyMargin
   where
     @grd1 cnt_status = Active
     @grd2 warning = FALSE
     @grd3 safetyMargin < SafetyThreshold
   then
     @act1 warning ≔ TRUE
 end
```

Figure 8. Warning is issued when safety margin is below threshold.

Several case studies have been modelled by following the presented phenomena, especially the monitored, controlled and commanded phenomena. Some of these case studies, such as a FEDAC system [7] and a sluice gate [13] were conducted as part of MSc projects by other users, and others, such as a cruise control system (CCS) [6], a lane departure warning system (LDWS) [5] and a lane centering controller (LCC) as part of this work by us.

IX. OVERVIEW OF RELATED WORK

In this section some of the relevant requirement engineering (RE) methods are discussed. Also a comparison of these methods and the MCMC guidelines are given.

A. Comparing MCMC with Phenomena and Frames in PF

A brief background on PF and its definition of phenomena was given in Section II. As discussed, PF is a general purpose RE approach. However, the MCMC is specific to control systems and its focus is on the

transition from an RD to a formal model. To achieve this transition, the MCMC guidelines uses the two individual phenomena *events* and *entities/variables* of PF to represent the dynamic behaviour of a system which is complex to model. It can also be beneficial to provide modelling guidelines for the other PF's phenomena. For instance *value phenomena*, can be modelled as **sets**, or **constants** of a formal model. In addition, requirements which represent phenomena of the kind relation can be modelled as **invariants** and **axioms**.

PF also provides five frames (patterns) for identifying different problem classes. One of the frames, called *commanded behaviour*, includes system operators. This frame helps to identify phenomena shared between the user and the machine (controller) and requirements related to the operator. However, the main aim of the MCMC approach is to guide formalisation of such requirements. Thus, in the MCMC approach the operator is to be understood from the system (requirement) point of view as well as the controller. This is one reason for defining *commanded variable phenomena* which are internal to the controller.

The other difference is that the MCMC approach explicitly distinguishes a specialised UI and an existing UI. This also helps to differentiate between *commanded* and *operator mode* event phenomena, which can have different communication with the controller. Such details are not mentioned in PF.

B. Four-variable model

The presented MCMC approach is influenced by the four-variable model [2]. Particularly, the concept of monitored and controlled phenomena are taken from the four-variable model.

The other two categories of the four-variable model are *input* and *output*. In [4] it is suggested that these are not used in abstract formal model; instead [4] provides patterns for introducing them in a step-wise manner using refinements. One other difference between the MCC guidelines and four-variable model is that variables defined in four-variable are continuous, while in MCC guidelines variables are discrete.

C. SCR

SCR (Software Cost Reduction) [14] 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, SCR uses *mode* classes (the monitored variable states), *terms* (quantities indirectly obtained from monitored variables), *conditions* (predicates of system states) and *events* (represent changes in system variables and mode).

Examining some of the examples of SCR [14], [15] shows that the interaction of an operator with the system are modelled as *monitored variables* and *terms*. The SCR does not define a separate quantity for operator commands. Our experience is that distinguishing monitored and commanded phenomenon facilitates formalisation and understanding of requirements as they serve distinct roles. SCR is more a specification method, while in this paper we focus on structuring the RD to specify a system formally.

The mode in SCR is different to the phenomena mode defined in the MCMC. The SCR represents mode as a "state machine defined on the monitored variables" [14]. This definition is preserved for some of the SCR case studies. However, in the example of a CCS [15] mode represents the states of the controller, which is not a monitored variable. In the MCMC approach, we tried to avoid ambiguity in definitions of phenomena. The example of the CCS in [15] also shows that it is necessary to explicitly differentiate between the controller's mode and monitored phenomena.

In SCR an engineer is required to identify the monitored, controlled, input and output phenomena of the system. However, we have a system-level view on the behaviour of the controller. Therefore, we focus on monitored and controlled phenomena initially and introduce input and output in refinement levels [5].

D. WRSPM

WRSPM [16] is a reference model for requirements and specifications of systems. WRSPM distinguishes between artifacts and phenomena which are shown in Figure 9. The artifacts are World, Requirements (how the customer needs the world to behave), Specification (information based on which a system satisfing requirement can be built), Program (implementation of the specification), and Machine (the basis provided for programming the system).

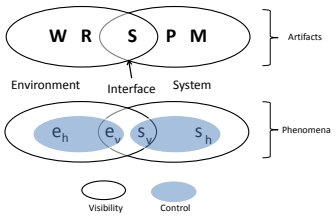

Figure 9. WRSPM reference model.

The phenomena of WRSPM define system states and events. Every phenomenon is controlled by either the *environment* (represented by *e*) or the *system* (represented by *s*), Figure 9. Phenomena which are controlled by the system are also visible to it. However not all the environment phenomena are visible to the system. The same is true for environment. Therefore, phenomena *e* and *s* are divided into hidden (e_h and s_h) and visible (e_v and s_v).

Similarly to the MCMC guidelines, the WRSPM also looks at the phenomena (states and transitions) of a system and its environment. The WRSPM is compared to the four-variable model in [16]. We adjust this comparison to include the MCMC's phenomena. Neither four-variable model nor MCMC guidelines discuss phenomena e_h (environment phenomena hidden from the system). This is because the objective of the MCMC approach is to model the system and its interactions with the environment and not the environment behaviour. *Monitored*, *commanded*

event and *operator mode event* of the MCMC approach represent phenomena e_v (environment phenomena visible to the system). Also, phenomena s_v (system phenomena visible to the environment) is shown by the *controlled*, *commanded variable*, *mode variable* and *controller mode event* phenomena in the MCMC approach. Phenomena s_h are not discussed in this paper, as they are not used in the requirement structuring.

X. Conclusion and Future Work

The main motivation of this work is to simplify the transition from an informal RD to its formal model. The proposed approach for identifying the MCMC phenomena and structuring requirements accordingly is the result of experimenting with various control system case studies, some of which are supported by industrial partners.

Our experience with several control system case studies has demonstrated that clear identification of the MCMC phenomenon and structuring of requirements based on these phenomenon has facilitated formalisation in Event-B. Furthermore, the RD structuring approach can help to improve system understanding.

In our future work we aim to introduce design details such as sensors, actuators and UI to the model. Also, the non-functional requirements are yet to be considered. Although we briefly discussed error handling in the LCC case study, error detection and timing requirements are to be explored further. Also, the MCMC is to be applied to more case studies, especially non-automotive case studies such as a train system for further improvements.

Acknowledgment

This work is supported by Global GM R&D, India Science Lab in Bangalore. We would also like to thank Manoranjan Satpathy, S. Ramesh and Silky Arora from the GM India Science Lab as our colleague Colin Snook for their helpful advice and support, especially with writing the requirements. Support from the EU FP7 DEPLOY Project (ICT-214158) is also acknowledged.

References

[1] J. M. Wing, "A specifier's introduction to formal methods," *Computer*, vol. 23, no. 9, pp. 8–24, 1990.

[2] D. L. Parnas and J. Madey, "Functional documents for computer systems," *Sci. Comput. Program.*, vol. 25, no. 1, pp. 41–61, 1995.

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

[4] 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.

[5] S. Yeganefard and M. Butler, "Structuring functional requirements of control systems to facilitate refinement-based formalisation," in *Proceedings of the 11th Workshop on Automated Verification of Critical Systems*, 2011.

[6] S. Yeganefard, M. Butler, and 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*. NASA, April 2010, pp. 182–191.

[7] S. Das, "Formal modelling of a FADEC system using Event-B," MSc Project Dissertation, 2010.

[8] J.-R. Abrial, "Cruise control requirement document," Internal report of the Deploy project, Tech. Rep., 2009.

[9] ——, *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.

[10] J.-W. Lee and B. Litkouhi, "Control and validation of automated lane centering and lane changing maneuver," *ASME Conference Proceedings*, vol. 2009, no. 48937, pp. 411–417, 2009.

[11] Jin-Woo Lee, "Model based predictive control for automated lane centering/changing control systems," http://www.faqs.org/patents/app/20100228420, September 2010, Patent number: 20100228420, cited 2012 Jan.

[12] S. Patwardhan, H.-S. Tan, and J. Guldner, "A general framework for automatic steering control: system analysis," in *American Control Conference, 1997. Proceedings of the 1997*, vol. 3, Jun 1997, pp. 1598 –1602 vol.3.

[13] M. Poppleton and G. Sulskus, "Patterns for modelling fault tolerant systems in Event-B," Rodin User and Developer Workshop, 27-29 February 2012, 2012, cited 2012 Feb.

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

[15] R. D. Jeffords and C. L. Heitmeyer, "A strategy for efficiently verifying requirements specifications using composition and invariants," *SIGSOFT Softw. Eng. Notes*, vol. 28, pp. 28–37, September 2003.

[16] C. Gunter, E. Gunter, M. Jackson, and P. Zave, "A reference model for requirements and specifications," *IEEE Softw.*, vol. 17, no. 3, pp. 37 –43, 2000.