

# A Component-Based Model-Driven Approach with traceability of concerns: Railway RBC Handover Case Study

Marc Sango

*PhD candidate, University of Lille 1 – IFSTTAR/COSYS/LEOST, INRIA/Spirals, Villeneuve d’Ascq, France  
marc.sango@ifsttar.fr*

---

## **Abstract**

In this paper, we introduce a domain-specific component-based approach with traceability of concerns. It is based on the separation of concerns, which is a long well known fundamental principle of software engineering but some time neglected in practice. Our approach is included in the V-based development process prescribed by the CENELEC EN-50128 standard of railway control and protection systems. The core of our approach is a meta-model including traceability meta-model, which can be used to enable traceability of concerns within or across component-based system abstraction levels. We use the meta-model to design a small software part of RBC handover case study of ERTMS/ETCS. To demonstrate the consistency of this case study model and the impact of verification results on traceability links, we use a model-driven transformation approach to validate some requirements. The validation is formally achieved by using the UPPAAL model checker tool.

**Keywords:** Traceability; Model, Verification; RBC handover; ERTMS/ETCS

---

## **Table of contents**

<b>1</b>	<b>INTRODUCTION</b> .....	<b>2</b>
<b>2</b>	<b>BACKGROUND ON TRACEABILITY</b> .....	<b>3</b>
<b>3</b>	<b>OUR COMPONENT-BASED MODEL-DRIVEN APPROACH</b> .....	<b>4</b>
3.1	THE OVERVIEW .....	4
3.2	THE META-MODEL .....	5
<b>4</b>	<b>CASE STUDY</b> .....	<b>7</b>
4.1	A BRIEF PRESENTATION OF ERTMS/ETCS .....	7
4.2	RBC HANDOVER CASE STUDY .....	8
4.3	TIME DELAY REQUIREMENTS .....	9
4.4	MODELS: SARA MODEL AND TIAO MODEL .....	9
4.5	ANALYSIS OF MODEL VERIFICATION RESULTS.....	11
<b>5</b>	<b>RELATED WORK AND DISCUSSION</b> .....	<b>12</b>
<b>6</b>	<b>CONCLUSION AND FUTURE WORK</b> .....	<b>13</b>

# 1 Introduction

Traceability and verification of requirements are key activities in the development process of safety-critical software systems, such as railway control and protection software, because these systems emphasize high qualities in term of Safety Integrity Level (EN-50128, 2011). For this reason certification agencies conduct reviews on the entire range of system development process to get the traceability information. To establish relationships between elements specified as requirements and elements implemented in the final system, software developers and testers are usually asked to provide accurate traceability information in order to guarantee that critical requirements are properly designed and implemented. In addition, information obtained from the traceability management can be used in different activities, such as change impact assessment (Zhao et al., 2002). Unfortunately, traceability information becomes obsolete very quickly during software development without the automatic or semi-automatic support to maintain traceability links among software artefacts (Mäder et al., 2009).

The advent of Model Driven Engineering (MDE) (Schmidt, 2006), which principles are to enhance the role of models and to increase the level of automation, provides a new landscape for dealing with traceability of concerns in software development process (Santiago et al., 2012). Indeed, the trace maintenance can be seen mainly as links between the elements of those models at different abstraction levels, from the user requirements, the architecture design specification and the source code. In the literature, there are several established traceability techniques (Galvao and Goknil, 2007; Konigs et al., 2012; Wen et al., 2014). For example, Wen et al. propose a traceability model by comparing multiple versions of software, by visualizing the evolution of architecture components in a graphical form, and by tracing the design evolution back to the requirement evolution (Wen et al., 2014). Their traceability model is designed specifically for the behaviour engineering approach, which first constructs a formal specification from requirements and then derives the design model.

In contrast, our approach is based on software Component-Based Model-Driven Development (CBMDD) (Chen et al., 2009), which provides an intermediate language for representing requirements in an architectural fashion. CBMDD is based on MDE and Component-Based Software Engineering (CBSE) (Szyperski et al., 2002), which has been acknowledged as a fruitful technology to improve the separation of concerns and to enhance reusability. Particularly, the use of domain-specific software component model allows to abstract domain-specific concerns, whether functional or non-functional, to address special industrial needs (Lau and Taweel, 2009). Although there are many general-purpose and domain-specific component-based approaches (Crnkovic et al., 2011; Pop et al., 2014), very few papers address the traceability challenge in software component (Gao et al., 2000; Chanda et al.

2012). As CBMDD spreads to specific critical domains, like aeronautical, automotive or railway control systems (Chess Project, 2012), the traceability challenge must require more attention in this development approach in order to guarantee that critical requirements are properly implemented as proposed in the European INESS Project (INESS Project, 2007) and national projects, such as the French scientific project named PERFECT for Performing Enhanced Rail Formal Engineering Constraints Traceability.

Although the take-up of CBMDD has been particularly evident in the aeronautical and automotive engineering domains, the railway sector lagged behind, partly due to out-dated standards and lack of awareness (Favaro and Sartori, 2014). But the situation is now changing rapidly, and the environment in recent works (e.g., Chess Project, 2012) is representative of the European awareness and push toward implementation of model-based approaches in railway engineering (Favaro and Sartori, 2014). In this way, in our previous works, we have proposed a domain-specific component-based model named SARA, dedicated to SAFETY-critical RAILway control applications (Sango et al., 2014a). Then, we have translated this model into Time Automata model over Inputs and Outputs actions (TAIO) (Sango et al., 2014b). Beyond this proposal, in this article, we demonstrate the traceability value of our approach and provide a more objective evaluation.

The remainder of the paper is structured as follows. In Section 2, we leverage the literature reviews and briefly present a background on traceability. In Section 3, we give an overview of our approach and our conceptual meta-models for traceability. Then, in Section 4, through an RBC handover case study, we provide a process to apply the meta-models defined. We also analyse the requirement change impact in the modelling and verification process of our case study. Finally, we discuss our approach in Section 5 before concluding and drawing future directions of our research in Section 6.

## 2 Background on traceability

Our interest is the traceability of concerns based on component-based model driven approaches. Traceability has been discussed in various domains. The IEEE provides the following definition of traceability (IEEE-Glossary, 1990): “Traceability is the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship one another”. This definition is adopted in the railway standard of software for railway control and protection system (EN-50128, 2011).

In requirements engineering, a lot of works has been done for tracing requirements from the stakeholders and in the design process (Gotel and Finkelstein, 1994; Ramesh et al., 1997). Gotel and Finkelstein define requirements traceability as “the ability to describe and follow

the life of a requirement, in both a forward and backward direction, i.e. from its origins, through its specification and development, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases” (Gotel and Finkelstein, 1994). With requirement traceability, we want to understand the impact of software requirement evolution on software design artefacts, particularly in the context of the European system for railway signalling called ERTMS/ETCS system (ERTMS/ETCS, 2014). These system requirements have been specified to replace the many incompatible different Automatic Train Control (ATC) systems used in different countries at different times. However, the specification is not yet stable as illustrate in the different versions of the ERMS/ETCS documents.

As already mentioned in the introduction, without the automatic or semi-automatic support to maintain traceability links among software artefacts, traceability information becomes obsolete very quickly during software development (Mäder et al., 2009). For this reason, in (Galvao and Goknil, 2007), a traceability taxonomy is presented for model driven engineering, which principles are to enhance the role of models and to increase the level of automation (Aizenbud-Reshef et al., 2006). Based on this survey, we combine the main advantages of component-based model-driven development (Chen et al., 2009), i.e., domain abstraction, automated refinement and separation of concerns to drive a traceability model based on our domain-specific component-based model-driven approach.

## **3 Our component-based model-driven approach**

### **3.1 The Overview**

Figure 1 shows an overview of our component-based model-driven development process. It is included to the “V development Lifecycle” (V-Lifecycle) prescribed by the CENELEC standard of railway safety-critical software for control and protection systems (EN-50128, 2011). In this V-Lifecycle, we highlight the part of V-Lifecycle that strictly concerns the software development activities, i.e., the surrounded part of the figure. It is based on our component-based development and evolution process named SARA. The SARA approach is a simple domain-specific component based approach, which is firstly dedicated for the modelling and verification of SAfety-critical RAILway control applications (Sango et al., 2014a, Sango et al., 2014b). As the V-Lifecycle, the SARA process includes two activities:

1. The component-based software design activities (the left branch of the surrounded part). From the phase of software requirement specifications, we want to separate functional requirements from safety and temporal requirements in software component-based architecture in order to facilitate the traceability of requirement concerns.

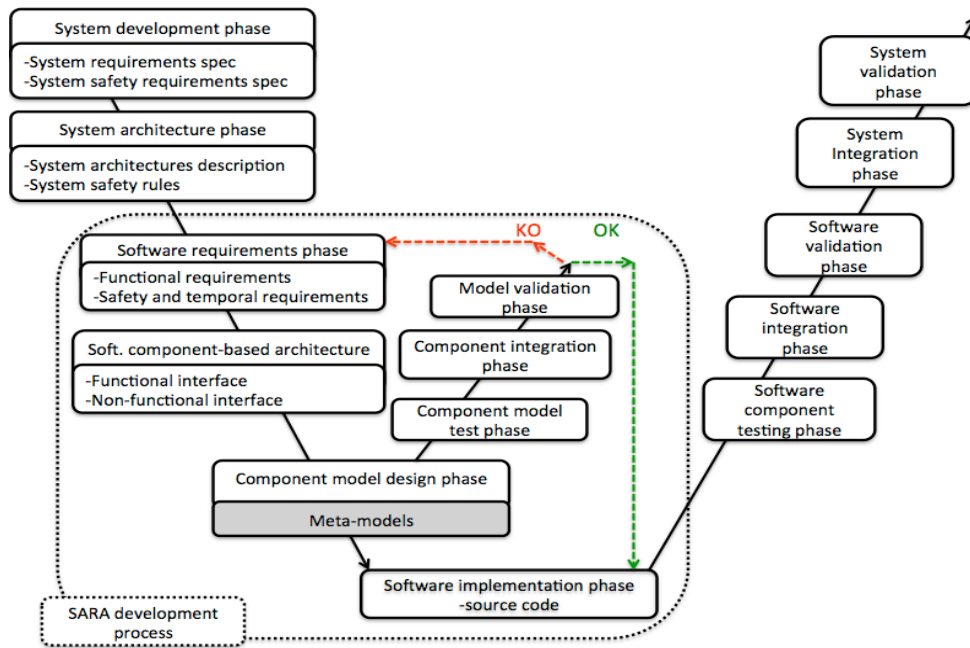


Figure 1. Our SARA component-based development process included in the V-Lifecycle

2. The verification activities performed at the model level (the right branch of the surrounded part). These verification activities performed at the model level are separated to other verification activities performed at the source code and the system level. They include both a test of individual components and their impact on the overall composite system. With these activities, we want to link back verification results (“ok” or “ko”) to initial requirements or intermediated models to ensure that specifications are properly implemented.

The SARA process is based on the core meta-model, which is illustrated with grey colour in the overview Figure 1. This meta-model is used for modelling software requirement concerns, component elements and the traceability links among elements in different abstractions levels, from design level to implementation level and verification level.

### 3.2 The meta-model

Figure 2 shows our component-based meta-model with traceability of concerns. The *ConcernModel* consists of *ConcernGroup*, *ComponentModel* and *TraceabilityModel*. A *ConcernGroup* assembles a set of concerns to be modelled. We focus here on ERTMS/ETCS requirement concerns. Each requirement is identified by a unique identifier (Id) and can be described in different ways (natural language, diagram like UML, mathematical expression, etc.). Requirements are either functional requirements or non-functional requirements. A concern-based taxonomy of requirements and traceability between requirements are not detailed here but we are based on taxonomy and relations among system requirements

(Dubois et al., 2010). As our main focus in this paper is the traceability of requirements through multiple abstraction levels of software component-based development process, we focus our discussion on some specific non-functional requirements for our case study. In this context, we focus on safety requirements subjected to real-time constraints (named TSRequirement in Figure 2), which are separated from other non-functional requirements (named OtherNonFunctional in Figure 2).

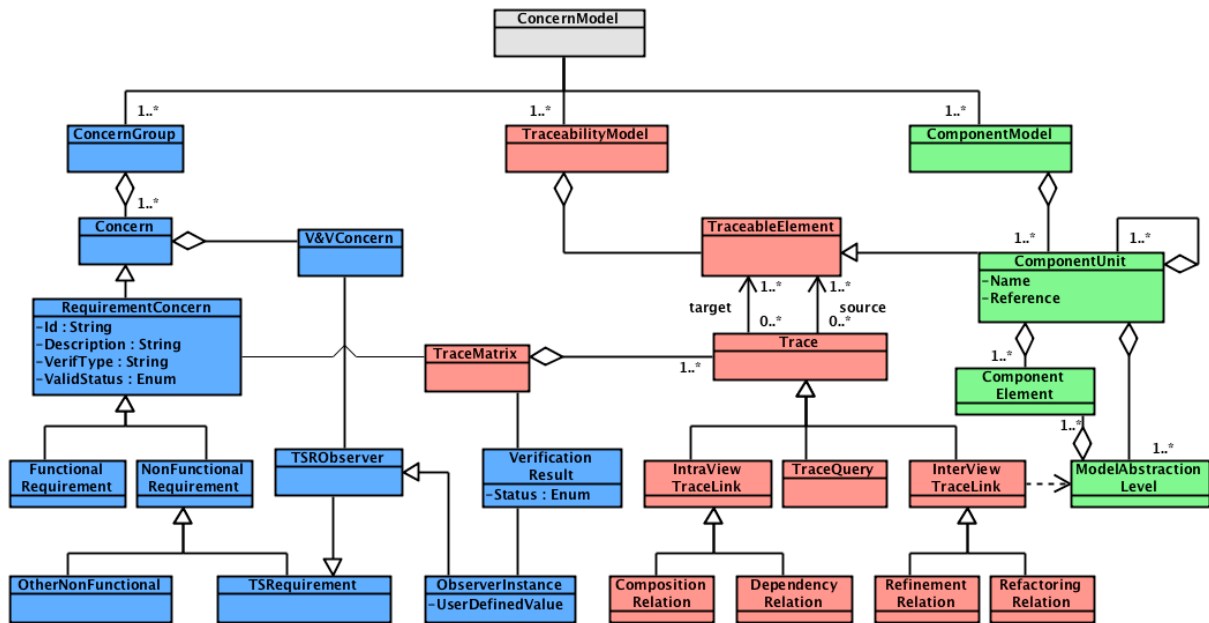


Figure 2. Our conceptual traceability meta-model

A *ComponentModel* models requirements. A model is viewed as a collection of component units. A component unit refers to an artefact (e.g., UML classes or codes) in the software life cycle. As a consequence, a component unit consists of one or more abstraction levels throughout the software model life cycle (e.g., model design level, model analysis level and model implementation level). To refer to component units, each component unit includes the attributes: name and reference. As illustrated in Figure 2, a component unit also consists of one or more component elements. A Component element can be component entities, component connections and component operations, which are not detailed in this paper. A detail on the component model can be found in (Sango et al., 2014b).

A *TraceabilityModel* links component units to requirements and vice versa. The linked component units and requirements form a set of traceable elements. Traceable elements are modelled explicitly by a trace, which links one or more source elements to one or more target elements. A Trace link is either an intra view trace link or an inter view trace link as shown in the Figure 2. Indeed, in our traceability context, important information are transformation relationships within or across models in the different abstraction levels. A model could be modified or transformed to another model. As a consequence, an intra view trace link can be a

composition relation or dependency relation, while, an inter view trace link can be refinement relation or refactoring relation through the model abstraction levels. Furthermore, to represent explicitly trace links, an explicit trace query is defined within or across views. A Trace can be stored and managed in the traceability matrix, see the meta-model in Figure 2. For this, we characterize each requirement with a verification type (VerifType) and a validation status (ValidStatus). The verification type is a characteristic used to indicate whether the verification used is model or code verification, a test or a formal proof. While, the validation status characterizes the status of the verification process. Initially, the status is unchecked and can be changed to passed, failed or inconclusive, depending on the verification process result.

## 4 Case study

We evaluate the applicability of our approach through one relevant and concrete case study of ERTMS/ETCS specification.

### 4.1 A brief presentation of ERTMS/ETCS

The case study presented in this paper is parts of ERTMS/ETCS (ERTMS/ETCS, 2014). The ERTMS is the European Rail Traffic Management System and the ETCS is the European Train Control Sub-system. Figure 3 shows the system architecture of ETCS and its interfaces with the Global System for Mobile communications for Railway (GSM-R) system and other signalling systems, such as specific national systems, interlocking and control centre.

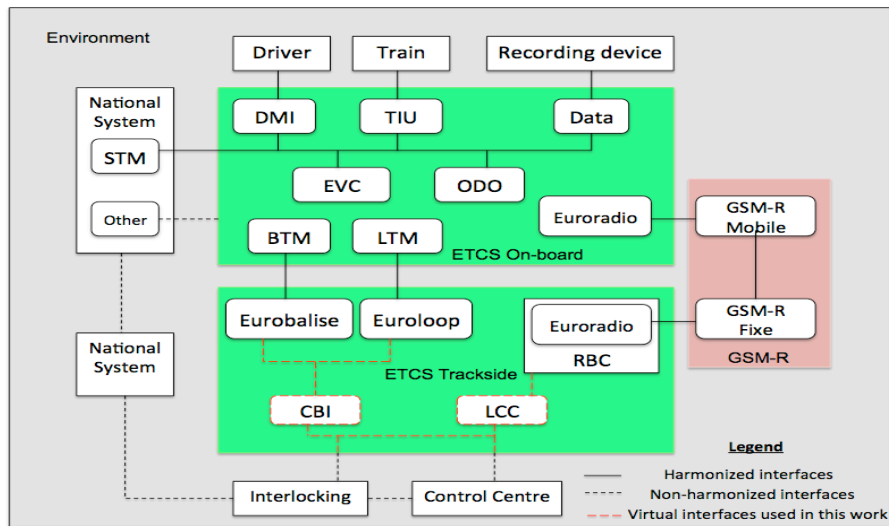


Figure 3. System architecture of the ERTMS/ETCS and its interfaces

The ETCS consists of two parts: the On-board part and the Trackside part. The detail of each part can be found in the specification. As shown in Figure 3, two types of interface specifications have been identified. The first category is the harmonized interfaces, such the interface between the Eurobalise and Eurobalise reader, which is called the Balise

Transmission Module (BTM). This category ensures that systems from different suppliers, which meet this specification, can work together without any further precautions. The second category is the non-harmonized interfaces, such as the interface between Radio Block Centre (RBC) and interlocking. This category will depend on the type of interlocking or national systems to be used on an ERTMS project or experimentation. For our experimentation, we add a virtual Computer-Based Interlocking (CBI), as shown in Figure 3, which communicates with interlocking, RBC, Eurobalise and Euroloop to provide the track occupancy information for the RBC handover case study.

## 4.2 RBC handover case study

The ERTMS/ETCS specification defines five application levels to express the possible operating scenarios between Trackside and On-board train sub-systems. These operating scenarios, such as start of mission, RBC handover, level crossing and so on, describe the behaviours and the expectative running modes of train control system. Here, we took the RBC handover scenario as a first case study. The handover procedure is called in order to transfer the train communication from the Handing Over RBC (HO RBC for short) to an Accepting RBC (AC RBC for short) and to avoid communication termination when the train gets outside the range of the HO RBC, as illustrated in Figure 4.

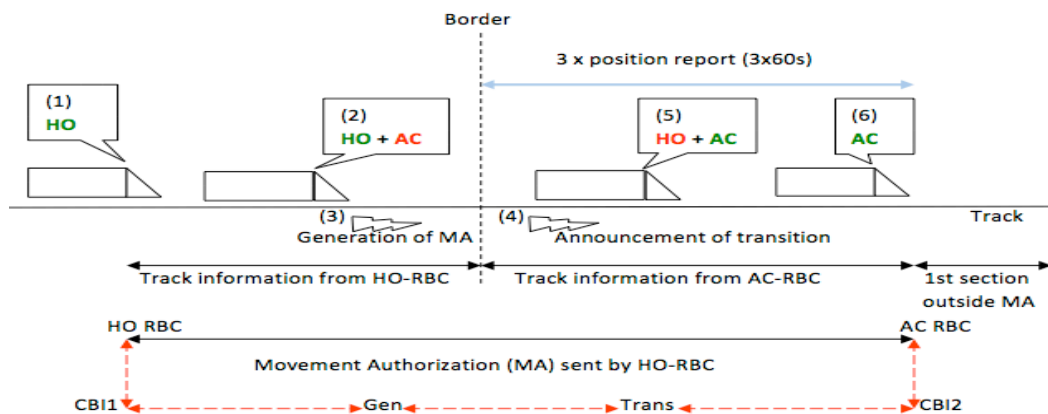


Figure 4. Example of RBC-RBC handover topography

In the specification, which can be found in paragraph §5.15 of (ERTMS/ETCS, 2014), there are two kinds of communication methods when the handover is announced: (i) two communication sessions can be handled simultaneously between the train and the both HO RBC and AC RBC, and (ii) only one communication session can be handled between the train and the HO RBC or AC RBC, exclusively. Here, we consider the second method. Figure 4 shows the main functional steps needed for running from one RBC area to another one: (1) pre-announcement of the transition by the HO RBC; (2) HO RBC requests the track information from AC RBC; (3) generation of Movement Authorization (MA) including the border information and track information from both HO RBC and AC RBC; (4)

announcement of the transition (5) transfer of train supervision to the AC RBC.

### 4.3 Time delay requirements

We focus on non-deterministic time delay performance. For this, we report in Figure 4 the delay scheme to deliver the track occupancy information. In the view of HO RBC, the minimum delivering time delay is  $d_{\min} = AC - CBI2 - Tran - Gen - CBI1 - HO$ , and the maximum delivering time delay is  $d_{\max} = d_{\min} + 3 * d_{\text{position\_report}}$ . In fact, “in case a communication session is established and no request to terminate the session is received from the HO RBC within a fixed waiting time after sending the position report, the position report shall be repeated with the fixed waiting time after each repetition” (§5.15.4 of ERTMS/ETCS, 2014). For our evaluation, we allow 3 possible repetitions of position report and each position report takes 60 s, as illustrated in Figure 4. This non-deterministic time delay from  $d_{\min}$  to  $d_{\max}$  is exactly what we concerned, and we want to check whether the response of the designed system can meet the time limit of the handover process at any time. At any time, the active handover process requirements considered as the following.

- R1: If an RBC receives track occupancy information during the current MA, it must send to the On-board system a Conditional EMergency stop (CEM) message immediately and it do not need acknowledge;
- R2: If an RBC receives track occupancy information of first section outside the current MA, the RBC determines with time delay  $d_{\text{wait}}$  if the train must be stopped before this first section to respect safety margin. If not, RBC will send Shorten MA (SMA) message. This message needs to be answered by the train in the delay  $d_{\text{ack}}$ ;
- R3: If an RBC receives track occupancy information of other section (except the first section) outside the current MA, the RBC will send SMA to train and this message need be answered in  $d_{\text{ack}}$ ;
- R4: In addition, if an RBC has not received the SMA response message ACK, but receives another track occupancy state information, it will no longer send CEM order to the train, but it will send an upgrade order Unconditional EMergency stop (UEM).

### 4.4 Models: SARA model and TIAO model

The conceptual meta-model of our approach presented in Section 3.2 is implemented with DTD/XML presentation for human-readability and for machine-readability. In addition, DTD/XML implementation can be easily translated in any other implementation language and vice versa. For example, the Ada programming language, which is one of the recommended languages in the development of railway safety-critical applications, can be translated into XML file by using ada-to-xml converter. Figure 5 shows the model abstraction levels of our RBC handover use case, implemented in the RBC-model-views.xml file.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE ModelAbstractionLevel PUBLIC "ModelAbstractionLevel" "component-model.dtd">
3 <ModelAbstractionLevel id="RBC_USE_1" ref="" name="RBC handover use case model">
4   <CompView id="Sara_View_ID" ref="RBC-Sara-model-view.xml" type="model design view"></CompView>
5   <CompView id="Uppaal_View_ID" ref="RBC-Uppaal-verification-view.xml" type="model analysis view"></CompView>
6   <CompView id="Ada_View_ID" ref="RBC-Ada-implementation-view.xml" type="model implementation view"></CompView>
7 </ModelAbstractionLevel>

```

Figure 5. The RBC handover model: the 3-layer abstraction views

It is a 3-layer abstraction levels: SARA component modelling view for the model design level (line 4 of Figure 5), its UPPAAL TAIO analysis view for the model analysis level (line 5 of Figure 5) and its ADA implementation view for the model implementation level (line 6 of Figure 5). By always keeping in mind the separation of concerns, each layer is defined in separate files: *RBC-sara-model-view.xml*, *RBC-Uppaal-verification-view.xml* and *LC-Ada-implementation-view.xml* as shown in Figure 5. In this paper, we focus only on the design layer and the analyse layer for the traceability of the system design specifications to the verification results and vice versa.

Based on the general architecture of Figure 3, which includes RBC component, the SARA model of RBC handover contains three main components: CBI, EVC and RBC. Here, we hide the implementation detail. In the left-hand of Figure 6, we only show its graphical representation in accordance with the requirement specifications presented in Section 4.3.

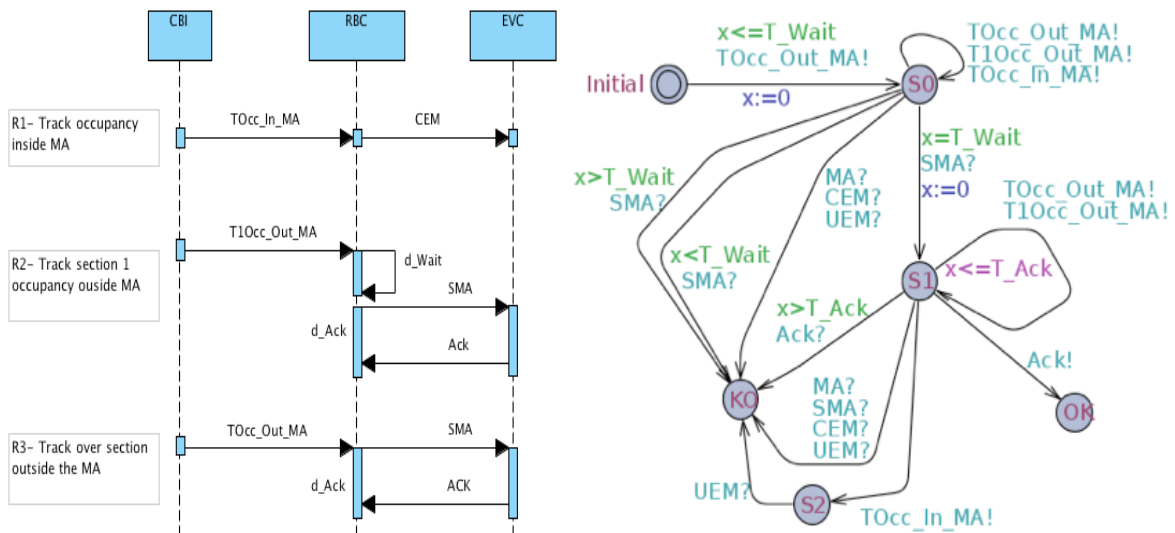


Figure 6. (Left) The RBC handover scenario; (Right) The RBC handover UPPAAL TAIO model

Then, based on our transformation process presented in (Sango et al., 2014b), the SARA model is translated into the TIAO model. The generated TIAO model is composed with the observer of requirement to be verified. Here, we focus on requirement R2 of Section 4.3. One part of the composition result is given in the right-hand of Figure 6, where, we focus on some specific states relevant for the test realised for this paper.

We realize manually 10 independent testing with UPPAAL-TRON framework (Larsen et al., 2005). By using the trace query between elements of design and analysis view of Figure 5, the results of testing are linked to requirement specification (R1, R2, R3 and R4) defined in Section 4.3. The left-hand of Figure 7 shows a traceability matrix in Excel file. The verification fails for the requirement R2. The detail of this requirement verification is shown in the right-hand of Figure 7.

	A	B	C	D	E	F
1	Requirement definition			Validation results		
2	REQ ID	Observer pattern	Expected output	Output	Status	Comment
3	R1	CEM must hold even if ACK never occurs.	CEM	CEM	PASSED	Validation with UPPAAL tool
4	R2	Wait a max delay before UEM	ACK(165,172)	UEM(145,146)	FAILED	Validation with UPPAAL tool
5	R3	Response ACK pattern	ACK	ACK	PASSED	Validation with UPPAAL tool
6	R4				UNCHECKED	No value supplied

RBC final state: IDLE.

Observable actions:  
input : T1Occ\_Out\_MA@(148;171)  
output : ACK

Internal actions annotations:  
PDelay\_Of\_TOcc@(145;148)  
QDelay\_Of\_TOcc@(146;150)  
TDelay\_Of\_TOcc@(165;169)  
PPDelay\_Of\_TOcc@(166;172)

Delay requirement: T\_Delay\_Max@200  
Expected output: ACK@(165;172)

---

Got unacceptable output: UEM@(145;146)

Figure 7. (Left) The requirement validation results; (Right) Focus on requirement R2

This means that with input action during the time window (148; 171), the expected state is OK and the output action must be ACK, but we get a KO state with unexpected output UEM. In deed, at each testing, we specified the four common delay strategy for track occupancy information (TOcc): proceeding delay (PDelay\_Of\_TOcc), queuing delay (QDelay\_Of\_TOcc), transmission delay (TDelay\_Of\_TOcc) and propagation delay (PPDelay\_Of\_TOcc). But during all the process, the maximum time unit for track occupancy information is set at 200 seconds (T\_Delay\_Max@200) in order to observe the influence of trans-boundary interlocking message delay.

#### 4.5 Analysis of model verification results

This evaluation showed that the time delay parameters in our initial RBC handover specification of Section 4.3 were inconsistent with the designed system. The cause of failure may be various. For example, the interlocking can transfer track occupancy information with a delay so long that exceeds the allowed time limit. Based on the analysing of the last good states and the message delivering time sequence set before the failure, we concluded in the system under verification automata (right-hand of Figure 6) the following failure path:

*T1Occ\_Out\_MA!TWait:SMA?Tack:TOcc\_In\_MA!UEM?*

In the subsequent experiments, we modified the time parameters in the requirement specification model by widening the upper and lower limit value of the delay, and the failure was no longer occurred. Note that our experimentation is realized based on an RBC scenario, which sounds theoretical because we only focus on software control part of RBC handover architecture shown in Figure 4. A concrete implementation must also involve hardware concerns. As a consequence, some statistics are strongly recommended to determine the suitable parameter values (e.g., *TWait*). In addition, when dealing with the traceability management and the requirement change impact, a rigorous safety risk assessment is required to meet the high Safety Integrity Level (SIL) (Beugin et al., 2007). For example, in France, when railway functionality is replaced, modified or updated, the well-known safety principle, named GAME for “*Globalement Au Moins Equivalent*” in French, meaning “*globally at least as good*”, has to be fulfilled.

## 5 Related work and discussion

We already introduced some related works (Aizenbud-Reshef et al., 2006; Galvao and Goknil, 2007) in Section 2. However some other specific related works (Ledang et al., 2008; Anquetil et al., 2010; Wen et al., 2014) have not yet been discussed. For example, in (Ledang et al., 2008), a traceability management in the context of Accord-UML, a MARTE-based methodology, is proposed and tested in the context of an automobile system design (Dubois et al., 2010). The traceability facilities provided by the SysML language enable to manage traceability links between requirements concerns and model elements. These works have been one of ours key sources of inspiration. While the technique to link requirement concerns among them and to link requirement concerns to model elements is the same, our approach enable in addition traceability links between model elements in the same design artefact. We have also investigated the feedback of verification results and design artefact evolution to the requirement evolution. The idea is to define a relationship between requirements and verification test cases that can determine whether a system fulfils requirements in our model transformation and verification approach (Sango et al., 2014b).

The problem of traceability has also been addressed in product lines engineering (Anquetil et al., 2010). In this paper, the main traceability tools (CaliberRM, DOORS, GEARS, SCADE suite, etc.) were evaluated in terms of five criteria: (i) traceability links, (ii) traceability queries, (iii) traceability views, (iv) extensibility, and (v) support for Software Product Line Development (SPLD) and Model-Driven Development (MDD). Their conclusions show that none of the investigated tools had built-in support for the three-mentioned development approaches. In addition, a vast majority of tools is closed, so they cannot be adapted to deal with the issues raised by these development approaches.

In (Wen et al., 2014), an evolutionary traceability model is proposed. They compare multiple versions of software and report the feedback of design evolution to the requirement evolution. Their traceability model is designed specifically for the behaviour engineering approach. In contrast, our approach is based on domain-specific software component-based model-driven development (Chen et al., 2009), which provides a language for representing requirements in an architectural fashion. In addition, as our approach is based on domain engineering, which is the first phase of product lines engineering (Alayed et al., 2013), our approach can be enhanced to go towards a model-driven traceability framework for software product lines.

## 6 Conclusion and future work

The main challenge we face in this paper is how to handle the traceability of requirements in the component-based development process in order to improve the analysis of requirements change impact on software modelling and verification. To face this challenge, we have presented a traceability meta-model, which highlights the separation of concerns provided by component-based development process. The meta-model can be used for modelling the requirement concerns, the component elements and the traceability links within or across different abstraction levels, e.g., component model design level, model analysis levels and model implementation level. The first purpose of our approach is to help to maintain traceability links consistency while software artefacts are evolving. For this, we propose to represent explicitly traceability links in the early development stage. The second purpose is to guarantee that the models used at different stages in the development process correctly fulfil the initial requirement expression. For this, we translate our high-level model to the low-level timed automata model for which we use the appropriate model checker tool for verification tasks. Depending on verification results, the traceability model allows going back in the model or in the specification to update or to change component or requirement parameters.

To evaluate our approach, we experiment the RBC handover use case of the ERTMS/ETCS specification. Although the approach has been shown applicable in this use case and the railroad level crossing use case, not presented in this paper, our approach can be enhanced in different directions. As a consequence, we are following several directions for future work. The first direction is the application of our approach to several use case scenarios to demonstrate the efficiency and the scalability of our approach. The second direction is to enable the automatic connection between our transformation approach and traceability mechanisms in order to update automatically the target models in the case of changes in the source models. In this direction, we also want to take advantage of model driven techniques and traceability of concerns in tooling in order to have validation feedbacks on source models. Concretely, in future work, we will focus on an integrated tool-chain demonstrator that will highlight several main results of this research work.

## References

- Aizenbud-Reshef, N., Nolan, B. T., Rubin, J., and Shaham-Gafni, Y. (2006). Model traceability. *IBM Syst. J.*, 45(3):515-526.
- Alayed, A., Lau, K.-K., Stepan, P., and Tran, C. (2013). Towards component-based domain engineering. In 39th EUROMICRO Conference on, pages 106–113.
- Anquetil, N., Kulesza, U., Mitschke, R., Moreira, A., Royer, J.-C., Rummler, A., and Sousa, A. (2010). A model-driven traceability framework for software product lines. *Softw. Syst. Model.*, 9(4):427–451.
- Beugin, J., Renaux, D., and Cauffriez, L. (2007). A SIL quantification approach based on an operating situation model for safety evaluation in complex guided transportation systems. *Reliability Engineering and System Safety*, 92(12):1686-1700. Special Issue on ESREL 2005.
- Chanda, J., Sengupta, S., Kanjilal, A., and Bhattacharya, S. (2012). Traceability between service component and class: A model based approach. *SIGSOFT Softw. Eng. Notes*, 37(6):1-5.
- Chen, Z., Liu, Z., Ravn, A. P., Stolz, V., and Zhan, N. (2009). Refinement and verification in component-based model-driven design. *Sci. Comput. Program.*, 74(4):168-196.
- Crnkovic, I., Sentilles, S., Vulgarakis, A., and Chaudron, M. R. V. (2011). A classification framework for software component models. *IEEE Trans. Software Eng.*, 37(5):593-615.
- Chess Project (2012). <http://www.chess-project.org/>.
- Dubois, H., Peraldi-Frati, M., and Lakhil, F. (2010). A model for requirements traceability in a heterogeneous model-based design process: Application to automotive embedded systems. In *Engineering of Complex Computer Systems (ICECCS)*, pages 233–242.
- EN-50128 (2011). *Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems*.
- ERTMS/ETCS (2014). *European rail traffic management system/european train control system. Release Notes to System Requirements Specification (SUBSET 026) Version 3.4.0.* [http://www.era.europa.eu/Document-Register/Pages/SystemRequirementsSpecification\(Recommendation\).aspx](http://www.era.europa.eu/Document-Register/Pages/SystemRequirementsSpecification(Recommendation).aspx).
- Favaro, J. and Sartori, G. (2014). Model-based software design in rail - an European perspective. *EURAILmag*, 30:224–227.
- Galvao, I. and Goknil, A. (2007). Survey of traceability approaches in model-driven engineering. In *Enterprise Distributed Object Computing Conference, 2007*, pages 313-313.
- Gao, J., Zhu, E., Shim, S., and Chang, L. (2000). Monitoring software components and component-based software. In *Computer Software and Applications Conference, 2000. COMPSAC 2000. The 24th Annual International*, pages 403-412.
- Gotel, O. C. Z. and Finkelstein, A. C. W. (1994). An analysis of the requirements traceability problem. In *Proceedings of the First IEEE International Conference on Requirements Engineering*, p. 94–101.
- INESS Project (2007): *INTEgrated European Signalling System*. <http://www.iness.eu/>

- Konigs, S. F., Beier, G., Figge, A., and Stark, R. (2012). Traceability in systems engineering - review of industrial practices, state-of-the-art technologies and new research solutions. *Advanced Eng. Informatics*, 26(4):924–940.
- Larsen, K. G., Mikucionis, M., Nielsen, B., and Skou, A. (2005). Testing real-time embedded software using uppaal-tron: An industrial case study. In *Proceedings of the 5th ACM International Conference on Embedded Software*, pages 299–306. ACM.
- Lau, K. and Taweel, F. M. (2009). Domain-specific software component models. In *Component-Based Software Engineering (CBSE 2009)*, June 24-26, 2009, Proceedings, pages 19-35.
- Lau, K.-K. and Wang, Z. (2007). Software component models. *Software Engineering, IEEE Transactions on*, 33(10):709–724.
- Ledang, H., Dubois, H., and Gérard, S. (2008). Towards a traceability model in a marte-based methodology for real-time embedded systems. *ISSE*, 4(3):189–193.
- Mäder, P., Gotel, O., and Philippow, I. (2009). Enabling automated traceability maintenance through the upkeep of traceability relations. In *5th European Conference on Model Driven Architecture - Foundations and Applications*. The Netherlands, June 23-26, 2009. Proceedings, pages 174-189.
- Pop, T., Hntynka, P., Hoek, P., Malohlava, M., and Bures, T. (2014). Comparison of component frameworks for real-time embedded systems. *Knowledge and Information Systems*, 40(1):127-170.
- Ramesh, B. and Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE Trans. Softw. Eng.*, 27(1):58–93.
- Ramesh, B., Stubbs, C., Powers, T., and Edwards, M. (1997). Requirements traceability: Theory and practice. *Ann. Softw. Eng.*, 3:397–415.
- Sango, M., Gransart, C., and Duchien, L. (2014a). Safety component-based approach and its application to ERTMS/ETCS on-board train control system. In *Transport Research Arena (TRA2014)*, Paris, France.
- Sango, M., Duchien, L., and Gransart, C. (2014b). Component-Based Modeling and Observer-Based Verification for Railway Safety-Critical Applications. In *The 11th International Symposium on FACS*, volume 8997 of *Lecture Notes in Computer Science*, Bertinoro, Italie.
- Santiago, I., Jiménez, A., Vara, J. M., De Castro, V., Bollati, V. A., and Marcos, E. (2012). Model-driven engineering as a new landscape for traceability management: A systematic literature review. *Inf. Softw. Tech.*, 54(12):1340-1356.
- Schmidt, D. C. (2006). Model-driven engineering. *IEEE Computer*, 39(2):25–31.
- Szyperski, C., Gruntz, D., and Murer, S. (2002). *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition.
- Wen, L., Tuffey, D., and Dromey, R. G. (2014). Formalizing the transition from requirements' change to design change using an evolutionary traceability model. *ISSE*, 10(3):181-202.
- Zhao, J., Yang, H., Xiang, L., and Xu, B. (2002). Change impact analysis to support architectural evolution. *Journal of Software Maintenance*, 14(5):317-333.