Modelling and verification methodology for control systems

Nesrine Darragi, El Miloudi El Koursi, Simon Collart-Dutilleul, Philippe Bon

Modelling and verification methodology for control systems

Nesrine Darragi[1], El-Miloudi El-Koursi, Simon Collart-Dutilleul, Philippe Bon

Univ Lille Nord de France, France
IFSTTAR, COSYS/ESTAS, Villeneuve d'Ascq, France

Abstract

The modelling process is dedicated to analysing requirements and designing systems in order to formalize text or other representations of requirement specifications. Verification is the qualification process that allows detection or prevention of defects in system design. A great variety of approaches of modelling and verification have been proposed in requirements engineering. Many of them have been widely used in the industry in recent years to . The diversity of techniques is the source of interoperability and portability weaknesses between projects and experts. This can lead to a large number of complex notations and methods that have to be understood. The expertise of many domains while necessary, is not always achievable.
The present work focuses on improving the performances of the modelling and the verification processes through the reuse of specific domain knowledge distributed among different similar projects. The methodology is for developing complex real-time control systems.

Keywords: Requirements Enginnering; Knowledge-based System; Modelling; Real-time Control System

Résumé

La modélisation est un processus dédié à l'analyse des exigences dans le but de formaliser les représentations informelles des exigences système. La vérification, en revanche, est un processus de qualification qui permet de détecter et prévenir les défaillances dans les modèles. Ces deux approches sont de plus en plus utilisées dans l'industrie pour maîtriser la complexité croissante des systèmes conçus.
Le papier propose une méthodologie générale de modélisation et de vérification pour les systèmes de contrôle à temps réel en se basant sur la ré-utilisabilité des connaissances des experts du domaine pour favoriser les échanges entre les projets similaires. Le papier présente une architecture d'une plate-forme dédiée à l'évaluation des systèmes basée sur un modèle générique de référence des systèmes de contrôle à temps-réel.

Mots-clé: L'ingénierie des exigences; Modélisation; Vérification; Systeme de Contrôle

[1] Corresponding author information here. Tel.: +33-3-20-43-84-07; fax: +33-320-43-83-98.
E-mail address: nesrine.darragi@ifsttar.fr.

Nomenclature

CORE	Controlled Requirements Expressions
DoD	american Department of Defense
ERTMS	European Rail Traffic Management System
ETCS	European Train Control System
KB	Knowledge Base
LOTOS	Language of Temporal Ordering Specification
RAMS	Reliability, Availability, Maintanbility and Safety/Security
RCS	Real-time Control System
SADT	Structured Analysis and Design Technique
SMV	Specification,Modelling and Verification
SQE	Software Quality Engineering
SRS	System Requirement Specification
UML	Unified Modeling Language
VDM	Vienna Definition Language
VORD	Viewpoint-Oriented Requirements Definition

1. Introduction

In the critical systems such as control systems for railways or aircraft, modeling and verification methods are a key issue to handle the complexity of those systems. To model a system design the scope has to be specified, which may not seem obvious to stakeholders particularly when requirements are not well or clearly-defined, since a model is a point of view of one or many aspects of the system.

The main difficulties in requirements engineering concern the identification of problem domain and solution domain. Without a clear understanding of the boundaries of these two domains, the distinction of requirements layers is difficult to solve. The first area concerns customers' requirements and their needs. It describes what a stakeholder wants to achieve and outlines its goals. The second area states what a system should do and how to do it. In other words, this domain describes the goals from the analyst and designer's points of view. Therefore, these goals are in different layers and levels. Only solution domain is concerned by this work, which do not treat the process of stakeholder requirement development. We assume that these specifications are written in natural language and other possible visual representations which may offer further illustration and more specific details.

The Real-time Control System (RCS) methodology proposed in this paper aims to improve human understanding of the design, since these systems require a minimum level of human interaction. The methodology objectives are to provide an understood module which could be integrated in systems with respect to domain constraints, and to ensure the efficency and a sufficient performance level of the system-to-be. We are interested in providing reusable and portable designs that could be extended in similar applications of RCS. For this reason, the methodology involves a set of integration rules in the early life cycle phases and dependability assessment.

Since it is difficult to elicit the expertise from experts, the knowledge acquisition does not seem obvious or systematic in knowledge management and it is considered as the most difficult step. Therefore, the use of techniques based on domain ontologies as specific domain knowledge is proposed in this paper. Besides the use of a specific vocabulary, consistent concepts and common notations, modelling allows specification structuring and system analyses. The choice of techniques depends on the nature of the system and its domain. For example, in the aircraft industry and aerodynamics and weight distribution, models are used. Timetable simulation and safety models are used in the railway industry. In order to model a RCS, it is necessary to use formal methods to conduct logical and rigorous testing of design correctness and to check requirements consistency.

Besides the use of a specific vocabulary, consistent concepts and common notations, modeling allows specification structuring and system analysis. Various representations are used to express requirements, such as statechart (Harel, 87), SADT (Ross, 77), CORE (Darke and Shanks, 97), VORD (Kotonya, 96) or techniques which are more object oriented like UML (OMG, 03) in addition to formal methods such as Z (Spivey, 89), B-Method (Abrial, 96), LOTOS (Bjorner, 87) and VDM (Jones, 96). The choice of techniques depends on the

nature of the system and its domain. For example, in the aircraft industry, aerodynamic and weight distribution models are used. Timetable simulation and safety models are used in the railway industry. In order to model a control system, it is necessary to use formal methods to conduct logical and rigorous testings of design correctness and to check requirements consistency.

Therefore, we propose to incorporate the verification process in every step in requirements engineering. Every level of requirements undergoes a specific method of testing in the qualification strategy. This work focuses on proposing a qualification strategy based on earlier and intensive tests. Our methodology provides a three-step process to model and verify requirement specifications.

This paper focuses on a qualification strategy based on earlier and intensive tests. Our methodology provides a three-step process to model and verify requirement specifications based on acquiring knowledge and negotiating stakeholder interests. A self-adaptive modelling system is proposed to simulate the behaviour of the system stakeholders and system components in order to generate a formal representation of its interactions. The latter may be improved by sourcing knowledge and capitalization base. A qualification strategy based on the requirements engineering V-model is introduced. Then the reference model, the general methodology and architecture of the RCS modelling and verification and are described.

2. Requirements qualification strategy

According to [DoD,1991] requirements engineering involves all life-cycle activities devoted to identification of user requirements, analysis of the requirements to derive additional requirements, documentation of the requirements as a specification, and validation of the documented requirements against user needs, as well as processes that support these activities. This definition lists most important activities and refers to the qualifying process which covers testing of models, design and final solution. Qualification is the evaluation of performance criteria. It combines verification and validation processes. The aim of the qualification is to increase the level of reliability in the quality of the system but also to improve the quality of other systems in the future. According to the processes of requirements engineering, the proposed procedure, in terms of the new platform, covers the verification of stakeholders, the system, subsystems and component requirements. Figure 1 shows the requirements in a V-Model (white boxes). The model illustrates the various stages of development and the relationship between requirements in different levels and testing. Each stage of development is characterized by a specific process based on the nature of requirements.

System testing can help to reduce risks, to prevent defects and to provide information for decision-making. Applying testing in the early stages of life requirements, is used to detect errors rather than facilitate the process of correcting and redefining the requirements. The cost of a correction in time and money of a system fault in the operation or maintenance phase is very expensive and costs the equivalent of correcting 1000 errors during the specification phase. Another advantage of early testing is the re-usability of tested requirements. Furthermore, the requirements can be reused to qualify similar products.

Figure 1 portrays the insertion of supplementary tests (grey boxes) to eliminate defects as early as possible. We elaborate a test planning to define the objectives of testing and a central planning to verify the progress of testing itself. The stakeholder requirement testing concerns the verification of informal specification. Similarly, system requirement testing includes an analysis process of controlled natural language in order to detect anomalies. A subsystem and component testing provides a very low level analysis. Qualification of components and requirements properties separately is not sufficient. We have to test the whole system at every level of integration. The refinement of requirements from coarse to fine granularity must to be verified too which is shown in figure 1. During these tests a bi-directional traceability is created between low and high level requirements. As figure 1 illustrates, the strategy focus on the solution domain. The scope of our methodology is limited to verifying system requirements in natural language and the design of system components and its verifications in preparing the system deployment.

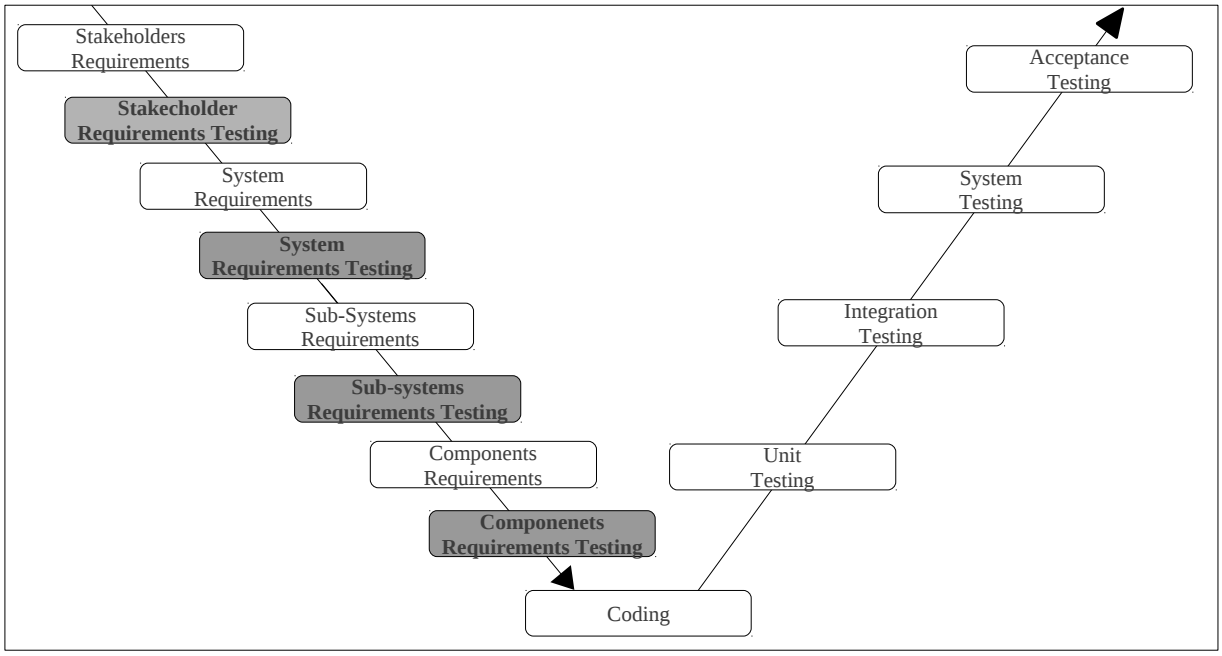

Fig. 1. Intensive requirements testing integrated in V-Model

3. The RCS methodology

To ensure the accuracy of the system to develop the first phase of the life cycle, the proposed methodology is based on three processes for specifying requirements, modelling designs and verifying models of RCS Requirements Specifications, SRS. RCS is defined to be embedded real-time systems designed to perform its functions using real-time knowledge. These computer systems monitor and control an environment connected to it through sensors, actuators and all other kinds of input-output interfaces (Fig. 2. (a)). The specificity of these systems is that they must meet timing constraints under certain assumptions. The predictability of the behavior of RCS must be guaranteed which is the most common denominator expected from these systems. Approaches adapted include the separation between hard and soft layers. Therefore, there are a consideration of two major divisions of architecture models. The other approach consists on modelling every aspect of the system by a specific model, the most suited model approach to the objective of the analysis and to the abstraction layers.

A preliminary definition of the key work requirement seems important before offring details of engineering trade-offs that must be considered in this work. According to (IEEE, 1998), requirement is a statement that identifies a product or process operational, functional, or design characteristics or constraints, which are unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality guidelines). The interpretation of the requirement specifications may lead to a misunderstanding between development engineers, system analysts and domain experts. In fact, between the real needs of the user, and what the expert expressed and all transformations by modeling techniques and what these techniques offer as an abstraction in a specific viewpoint, there is a risk of having different understandings of the requirements. The ambiguity and the incompleteness of requirements, which lead to the inaccuracy or incorrectness of the system, are mainly due to the difference between the viewpoints. Therefore it is important to share domain knowledge between different stakeholders during the development process. In requirement engineering, the use of ontology's aim is the knowledges standardization (Kaiya et al. ,2005)

3.1. The RCS architecture reference model

The methodology presented in this paper is based on the RCS architecture (Fig 2. (a)). The general structure is composed of three main functional modules which are sensor system, actuator system and the control system. The actuator and sensor interact with the environment via interfaces using networks characterized by its connectivities, latencies, bandwidths, reliabilities... The control system may be modeled as shown in (Fig. 2. (b)) by a hybrid architecture composed, in addition to the standard modules of sensing and perception, by states,

goals, actions, plans, communications and KB [Darragi et al. 2013]. Other modules of world modelling and fault tolerance may be used to extend the initial architecture.

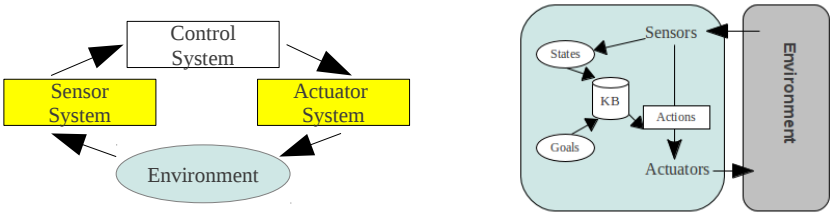

Fig. 2. (a) General structure of a RCS; (b) Simple conceptual view of a RCS

3.2. The RCS methodology processes

A three-step process of analysing and modelling requirements (Fig. 3) is proposed in this work . Each step is made up of one or more steps. There is a decomposition of RCS requirements into layers and levels of abstraction. The figure 3 describes the types of models carried out at each level of abstraction. Starting by “white boxes” in specification and modelling and coming to “black boxes” in simulation, the methodoly covers many aspects of the system behavior and environment.

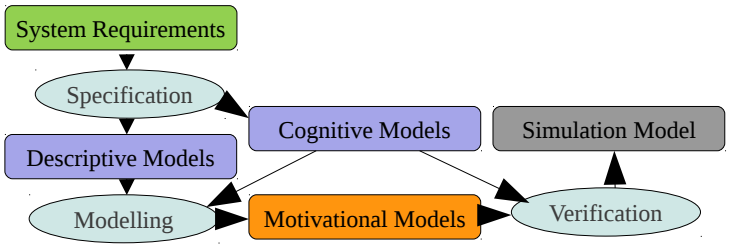

Fig. 3. RCS Methodology processes (SMV)

The first process of SMV is the specification. The formalization of informal requirement is ensured by methods for specificaiton.The goal of this step in the proposed approach is to discuss a structuring specification method largely build on cognitive techniques and shared domain knowledge for a sharp reasoning of each application domain and each case study.

The modelling RCS behavior aims to provide formal models of system requirements specifications and to verify and validate them. The first process is the requirement structuration shown by figure 4. This process provides, from a specification document provided by the client, structured specifications through templates in order to be modeled later. The template base and a context-free grammar are proposed to parse and to capture specifications. The structures provide higher hierarchical abstraction requirements. The aim of this step is to provide a high level description of the system and structured requirements which could be manipulated in future steps. This includes a verification of some requirements properties such as the consistency, the completeness or unambiguity. The objective is to provide a clear, concise and unambiguous specifications based on the use of domain knowledge, norms, standards and capitalization. The second step is the formalization. From the templates and graphs provided in the previous phase, a formal model is created. This allows for abstract specifications of system properties and behaviours to be provided. It allows an executable model of the system operation to be obtained. The last process is the verification. From the previous results, a process of data verification will be programmed. The advantage of this phase is that anomalies are detected as early as possible in the structuring and in the customer's needs, but also to test requirements. It will identify the execution inconsistencies and thus ensure corrections are made before deploying.

The methodology aims to give guidance on the use of already existing knowledge such as standards, safety norms, real-time constraints, incident reports, systems models, anomaly reports, regulations, technical concepts, relevant product designs and all other kinds of knowledge to construct, verify and validate the system model. Since the proposed approach is addressed for safety and performance analysis of RCS, a related safety knowledge is required. The same methodology could be applied to analyse and to verify a protocol development which is considered as a software system. The unique difference with the analysis of RCS is the knowledge base that will be used during the three-step process.

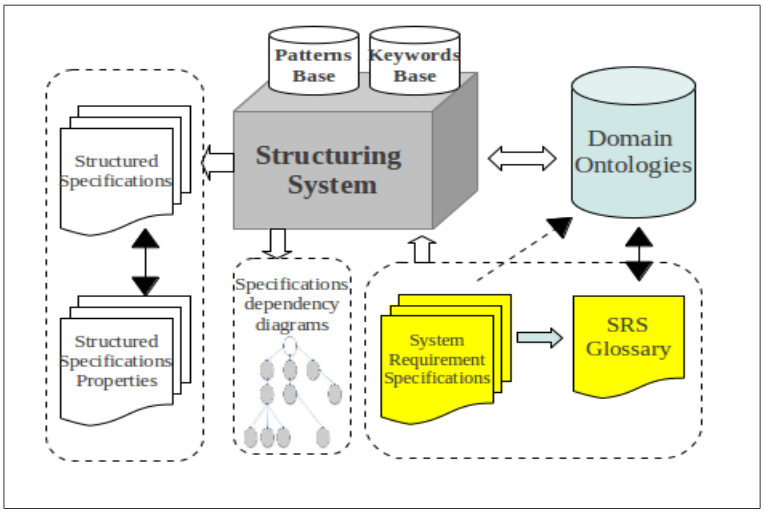

Fig. 4. Specification Domain Independant Framework Architecture

The modelling process aims to create a model of the system and its environment. (Fig. 5. (a)) depicts models needed to cover the modelling of RCS behavior in its environment and its interactions. The RCS is made up of several views shown by a meta-model in Fig. 5. (b). The goal model focuses on identifying problems and on exploring system solutions and alternatives.

- Behaviour Model: to prove the correctness of the system behavior and to assess to the real behavior of the system, a behavioral model with not too many assumptions is important.
- Goal Model: This model helps to elicit and complete requirements, to model goals of system entities, to detect conflicts between goals and to resolve conflicts.
- Object Model: This model shows the relationships between different concepts.
- Operational Model: ...performance yields the meta-relationship between agent and operation.
- Agent Model: It represents the behavior of various types of agents representing system components and stackeholders.
- Obstacle Model: To validate a critical system, its reliability has to be validated in addition to its safety and its security which means the necessity to establish the hazard and risk analysis. For the first type of analysis, the identification of possible threats, their effects, their rates and their severity are essential and obligatory.

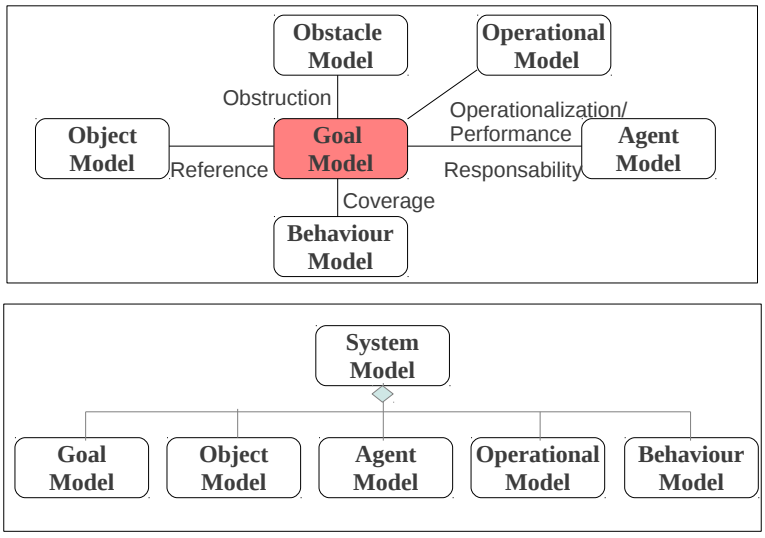

Fig. 5. (a) SMV representation models; (b) Meta-model of the overall representations

The third process is the verification which consists in investigating whether the RCS is correct and confirm to its specifications. Safety properties and lifeness progress are the key properties to be verified.
Two majors categories of verification techniques are defined by (Roache, 1998): Code verification and solution verification. Other classification of the code verification is proposed by (Trucano, 2002). He proposed to segregate this category to numerical code verification and software quality engineering SQE. The goal of this latter is to check the reliability of the software. Based on analysis, observations and comparaisons, code verification focus on spatial and temporal verification.

The solution verification consists on the quantitative estimation of the numerical accurancy of a given solution according to (Oberkampf et al, 2007). For the development of safety-critical applications the most used verification approaches are the theorem proving and the model checking. The first method consists of constructing a mathematical proof for a mathematical statement to give a proof and a convincing argument that the system is true. The problem with this kind of verification is in the case where the proof is not found. The interpretation is ambiguous. The statement could be true or false. The model checking is an automated approach to verify that a model of a finite state system satisfies a formal specification of requirements to the system. It means that there are two steps; creating a model of the system and formalizing the SRS to obtain formal specifications and then using a model checker to check whether the model actually satisfies the property specification. Modelling a system could be done in many ways and the same thing for the formalization of the specifications. For us, this is the major problem with this approach especially if we are not sure of informal requirements itself. Well-formed and formally verified SRS could generate in real-time execution unsafe scenarios where applied under some assumptions. Therefore, requirements should be verified to satisfy key properties called RAMS for Reliability, Availability, Maintability and Safety/Security. The verification of these properties and other desiderata (i.e recommendations or requirements) for specification has to be as early as possible which is presented in the first section of the qualification strategy.

Different steps of verification are planned in different layers and levels of abstraction. A checking process is incorporated in the specification process to test if a requirement is unique or if a requirement is atomic. This kind of tests is possible without knowing many information related to the set of requirements, the implementation or the context of the application. A second set of properties is defined to be verified during the modelling process which is mandatory for the qualification strategy. This step needs more knowledge about the entire SRS, integration rules, standards and domain specific constraints. For example the checking of unambiguity or the consistency of requirements is done against a set of requirements. The third step of verification is what we called here by the third process in the SMV. It concerns the third cluster of desiderata for specification such as the necessity, the completeness or the realistic nature of requirements. To achieve this step, there is a need for real-time constraints to prove that the system is fault-free from unsafe scenarios which are predicted in previous fault pre-processing.

All previous checking processes concern informal requirements which are formalized subsequently. The behavior of the system itself needs to be verified despite its requirement specifications. The next section describes the architecture of the general framework of SMV and the verification process.

3.3. The RCS methodology architecture

As shown in Fig. 6, the framework is composed by white boxes designing offline processes, black boxes to present online processes and grey boxes which are used by the framework for implementation purposes. The box of modelling control system behavior is the combination of the two first processes SMV: specification and modelling. The input of this latter is the informal requirements written in natural language and the outputs are a set of models (such as goal model, behavioral model, reliability model, availability model, performance model,...), data related to the specification (metadata used for traceability) and a model-based simulation for RCS.

The second white box concerns the pre-processing faults. This process is an offline application designated to analyse faulty scenarios based on behavioural model and reliability model. It is a requirement error estimation driven by assumptions. This technique uses domain knowledge and domain specific constraints to search “error pattern” which will be the input of the third white box, Fault prediction. These patterns are used to identify specification weaknesses and probable errors. The categories of faults concerned with this work at this stage is the composed and nested errors shown by the Fig. 7.

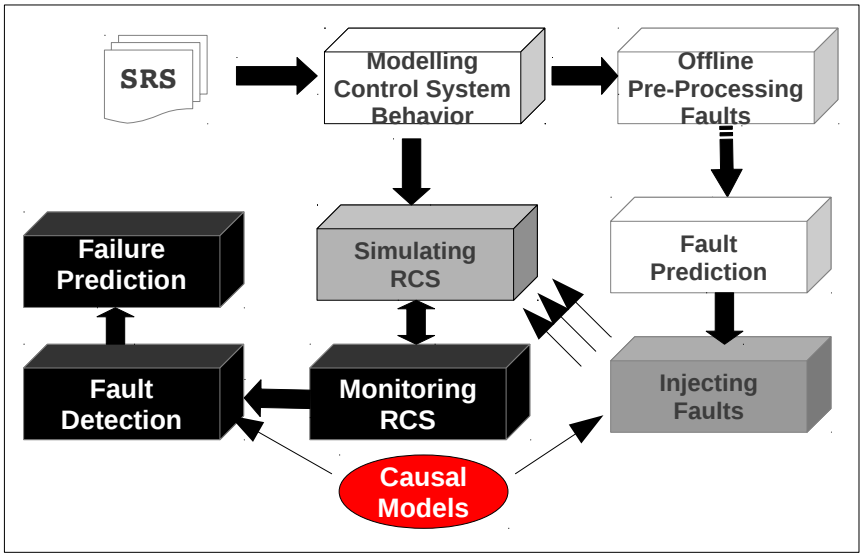

Fig. 6. General Framework Architecture of SMV

In general, fault propagation is from the elementary entity or the component to the subsytem and, finally, the system or equipment which crashes. The other assumption that we assumed in this work is that an accident is caused by many failures at different levels of abstraction.
Intrigued system behaviour under some conditions determined in previous steps is identified. It is the input of the injection faults process which operates in realtime (in this case, the simulator is the real-time environment).

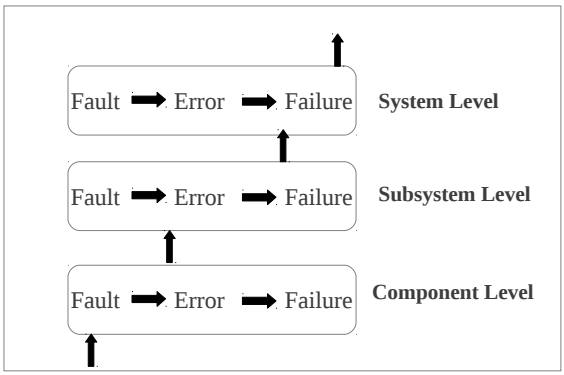

Fig. 7. Failure propagation

The monitoring system of the current RCS sniffs real-time messages exchanged between agents (entities and components) and all accessible simulation parameters. Based on “fault patterns” also called fault symptoms, the fault detection process analyses execution traces to identify unsafe states of the components. A causal model and component failure modes are used to determine the probability of the fault occurrence. This is beneficial to analyse quickly traces and to take a quick decision to trigger the process of the failure prediction as soon as possible. This allows us to accelerate the process to avoid the reaching of faulty states before the crash or the accident.

If a failure is detected, the probable faulty scenario is added to the fault analysis report to be identified as a possible cause of the crash. If there is no failure detection, the intrigued state which triggered the failure prediction process is quarantined. It may be considered as a potential fault that could cause a failure in the future. This is a way to guarantee the traceability of failure in the system.

4. Conclusion

This paper presents a general methodology of requirements modelling and verification. The aim is to assist the system design modelling and requirement qualification. We develop a methodology based on the knowledge -based systems to enable and support the reuse of already existing knowledge and expertise analyses. The architecture presented in this work is an independant-domain platform architecture based on the general methodology and designed to evaluate the safety and the reliability of RCS. It consists in using model-based approaches. The proposed methodology will be validated by using a distributed architecture of the European Rail Traffic Management System/ European Train Control System ERTMS/ETCS as a case study of RCS.

Aknowledgment

The authors would like to thank Ms Emily Holligan for providing language help. They also want to thank the anonymous reviewers for their efforts and their advice on improving this paper.

References

Jean-Raymond Abrial (1996). The B Book, Assigning Programs to Meanings. *Cambridge University Press, Cambridge.*

Dines Bjørner, C. A. R. Hoare, and H. Langmaack (1990). VDM and Z---Formal Methods in Software Development, *Kiel, volume 428 of Lecture Notes in Computer Science. Springer Verlag.*

Darragi N., Bon P. and Collart-Dutilleul S. (2013). Tropos for embedded real-time control system modeling and simulation. Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems.

Darke and Shanks (1997). Successfully completing case study research:combining rigour, relevance and pragmatism, *Info Systems J (1998) 8* , 273±289

DoD (1991). Software Measurement for DoD Systems: Recommendations for Initial Core Measures. T*echnical report, CMU/SEI-92-TR-19., Software Engineering Institute*.

Harel, D (1987). Statecharts: A visual Formalism for Complex Systems. *Science of Computer Programming* , pp. 231-274.

IEEE, 1998: 830-1998- IEEE Recommended Practice for Software Requirements Specifications.

Kotonya, G. & Sommerville, I. (1996). Requirements Engineering with Viewpoints. *Software Engineering Journal*, pp. 5-11.

OMG (2003). *The Unified Modeling Language Version 2*.

Ross (1977). Structured Analysis (SA): A Language for Communicating Ideas. *in: IEEE Trans. Software Eng. 3(1).* pp. 16–34.

J.M. Spivey (1998). The Z Notation: A Reference Manual. *Programming Research Group University of Oxford by Prentice Hall International (UK)*

Trucano, T.G, Pilch, M., Oberkampf, W.L, (2002). General Concepts for Experimental Validation of ASCI Code Applications. Sandia National Laboratories, SAND2002-0341.