OETCS TK-01-01

openETCS

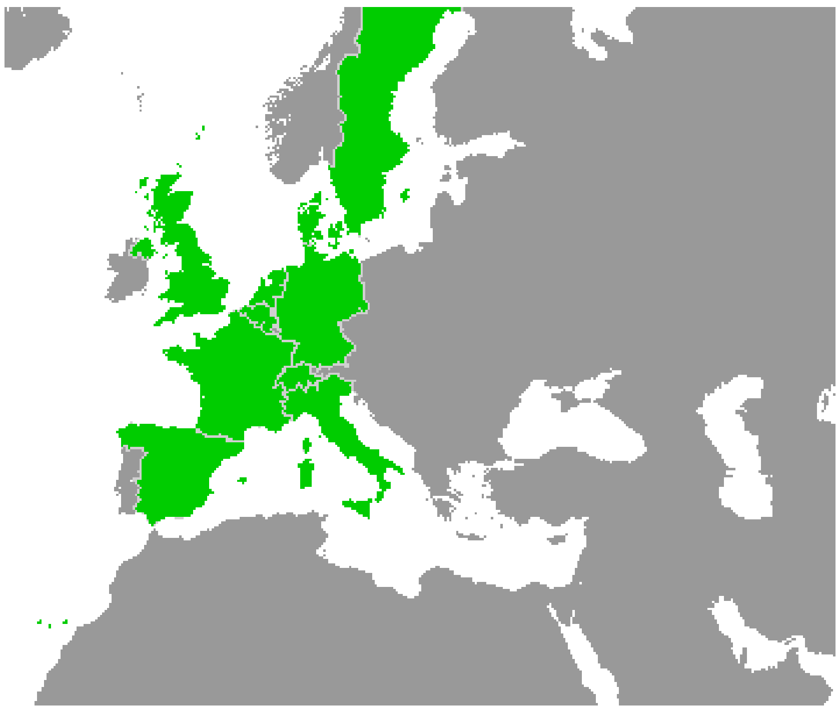
ITEA2 Project
2012 – 2015

Work-Package 2: “Requirements”

Safety properties for OpenETCS through two examples

Sylvain Baro
SNCF

February 2013



Safety properties for OpenETCS through two examples

Sylvain Baro
SNCF

SNCF INFRA/IG-SYS

Prepared for ITEA2 openETCS consortium
Europa

1 Introduction

Some safety documentation is available in the ERTMS documentation set, despite it has not been updated for the Baseline 3.

The SUBSET-88 provides system fault trees, and lists several FMEA documents for various parts of the system. The SUBSET-91 summarize the leaves-events of the fault trees, and provides some safety requirements.

The goal of the current document is to go through some prototyping examples chosen in the document *SRS subset for modelling tool benchmarking* and to define on these examples how the Safety Requirement could be extracted from the ERTMS documentation downto the model level, and what formalism could be used in order to express them.

Please, note that the purpose of the models provided herebelow is not to be completely "formal". It is only to be sufficiently formal in order to convince oneself that it will be possible to express executable models and safety properties, and to outline the process followed in order to provide these.

The formalism used for the model is state machines with actions and variables. As for the safety properties it is a simple propositional calculus which specific predicates to handle transitions and states of the model [1].

Please also note that the process followed is described in an "intuitive" way, in order to make it easily understandable and to show the different options. This document is not meant to describe the Safety Analysis process in a formal way, this should be the purpose of the Safety Plan (WP4). The process describe here should only be considered as "one possible solution".

2 Reference documents

- WP2/D01 — *SRS subset for modelling tool benchmarking*
- SUBSET-026 3.3.0 — *System Requirement Specification*
- SUBSET-081 2.3.0 — *Transmission System: Failure modes and effects analysis*
- SUBSET-088 2.3.0 — *ETCS Application Levels 1 & 2 - Safety Analysis*
- SUBSET-091 2.5.0 — *Safety Requirements for the Technical Interoperability of ETCS in Levels 1 & 2*

3 Example 1: Establishing a communication session

In this example, we try to model the requirements in the 3.5.3 section of the SRS SUBSET 26, for Level 2/3.

The whole purpose of the function is to establish a communication either from the OBU, or from the RBC. For the sake of simplicity, we will only cover the case where the communication is initiated by the OBU.

[1] This is sufficient for the examples in this document, but it will probably not be the case for the whole OpenETCS project.

For IPR reason, the figure was removed.
The reader shoud refer to SUBSET-88-2 Part 1, Appendix B, p. 2

Figure 1. Core Hazard Fault Tree (SUBSET-88)

For IPR reason, the figure was removed.
The reader shoud refer to SUBSET-88-2 Part 1, Appendix B, p. 8

Figure 2. Incorrect Mode Fault Tree (SUBSET-88)

3.1 Safety Properties from UNISIG documents

The processus of identification of the safety properties is a twin process of the processus of defining the system and subsystems.

Once the overall system is defined, the first step for the safety analysis should be to identify which are the Safety Properties laying on the objects that will be studied and modelled. Usually, in order to defined these Properties, it is necessary to start at a very high level by the Preliminary Hazard Analysis (PHA) which states the system Dread Events for the overall system. These events are very high level: for example we could find electrocution, or head-on collision.

The overall system should then be split into different subsystem with different area of responsibilities (for example onboard subsystem, interlocking, driver…)

The system-level Dread Events must now be allocated to the different subsystems according to the system architecture, and refined to match the subsystem boundaries and area of responsibilities. This can be done using Fault Tree[2] Analysis.

In the case of ERTMS, SUBSET-88 provides such Fault Trees. Fig. 1 describes for example the events leading to the Dread Event "Brake Control Function Failure[3]". The main event is then refined to leaves or other sub-Fault Trees, in particular the Tree identified by GATE80, corresponding to an incorrect determination of the current mode.

This mid-level event is then also refined (Fig. 2) to events and especially leaves that are expected to be of the SRS level. This leads in particular to the lower-level events KERNEL-3 to KERNEL-6 (see Fig. 2 or 3) which are related to the radio transmission. These lower level events are summarized into the SUBSET-91 document, which is a mandatory part of the ERTMS documentation.

Open Issue. Should the Fault Trees Analysis provided in the SUBSET-88 be considered as complete? My feeling is that it is not the case, because there is no ERTMS Safety Plan (AFAIK) to track the way the analysis was conducted, so there is no proof that the analysis is sufficiently rigorous.

Moreover, we'll see in Sect. 3.4 that the trees should be refined again before being at the proper level.

It is now necessary to analyze if each scenario is covered and how (and if the coverage is sufficient). We will call "scenario" the succession of events needed to provoke a studied Dread Event (which could be high- or low-level). Covering the scenario could be done by proving

[2]We will call them Fault Trees even if all events in the trees will not necessarily be Faults and Failures but could be random or sparse events.

[3]Event if this does not go up to the main system Dread Events, one can easily imagine how this could cause a collision

For IPR reason, the figure was removed.
The reader shoud refer to SUBSET-88-2 Part 2, p. 23-24

Figure 3. Kernel Dread Events (as in SUBSET-88 or SUBSET-91)

For IPR reason, the figure was removed.
The reader shoud refer to SUBSET-81-2, p. 15

Figure 4. Part of the FMEA for Transmission System (SUBSET-81)

that a requested leaf is always impossible to meet, or by proving that the conjunction of all the requested sub-events would be rarer than the THR (Tolerable Hazard Rate) requested for ERTMS application or its allocation to the studied sub-system.

SUBSET-88 provides the following informations on the events that interest us (cf. Fig. 3). This table lists the base events (leaves) of the tree and describes the impact of such events, and the mitigation used to avoid safety problems. The problem is that whether there is a safety issue is not made that explicit: if we take KERNEL-6, for example, the description states that a loss of communication could lead either to a failure to receive more restrictive route information, or to the driver being obliged to use a lower level. We can imagine that these are not catastrophic consequences (we wouldn't risk the safety on the probability that a message does not reach its destination), but in order to make this statement, it needs a system-level analysis.

Nevertheless, in this case the third part of SUBSET-88 (9.3.3.1) provides an answer [4]:

> As indicated in 9.2.2.1, the data exchange between track and train is defined in the ETCS specifications such that normally the deletion of a message does not result in a hazard. Moreover, deletion of critical messages is mitigated by means of acknowledgement procedures. The only case where deletion may lead to a hazardous situation is in the case of Emergency messages and this situation is considered in the following section. On the basis of these further considerations the possibility of undetected deletion of messages is not carried forward as a provable / testable target.

> *Open Issue. Is it in our scope to provide the analysis stating that message deletion never leads to an unsafe situation?*

In the case of KERNEL-4, on the other hand, the situation is clearer. The table states that "Radio link supervision" is a protective function against this event. Hence it is here easy to determine a safety property for the corresponding function!

In the particular case of the transmission, there is yet another document that provides information on safety issue: SUBSET-81 "FMEA for Transmission System". We see for the line 5.2.4.2 (Fig. 4) that some of the mitigations of the consequences of the loss of message lie in the transmission protocol, but also in the way of using/dimensionning ERTMS, which should consider (at a system-level) the impact of the loss of messages.

3.2 Model of the function

The (simplified) proposed model for the function is provided Fig. 5 and 6. The transitions and actions are the following. It is not useful here to detail more the actions and formalism. In the following, the state "Checked" corresponds to the full establishment of the communication (communcation is established and version is checked).

[4]The same explanation may be found also in SUBSET-91.

From	**To**	**Condition**	**Action**
NoCom	Trying	Order(RBC_ID) received	RBC_Com := RBC_ID
Trying	Trying	No connexion setup	Req. setup of connexion to RBC_ID
Trying	Setup	Connexion setup	Send Initiation Msg
Setup	Established	Received System_Version(RBC_ID,V)	V_Com := V
Established	NoCom	not compatible(V_Com)	Send Incompatible_Version Msg; Inform Driver; Terminate (3.5.5)
Established	EstabChecked	compatible(V_Com)	Send Estab. report
. . .	. . .	. . .	. . .

Figure 5. Establish a communication session (transitions)

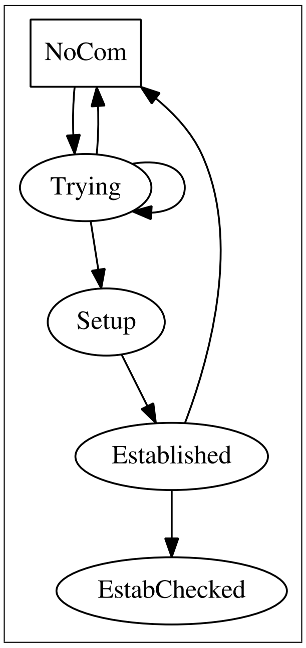


Figure 6. Establish a communication session

3.3 Safety properties

If we try to infer some safety properties from this state machine, in a *bottom-up* way (rather than the *top-down* way presented in Sect. 3.1) we can identify a few possible failures. A process to be used in order to find out these events could be to build a system FMECA starting from the formal model.

Note: In the following, we use the verb "to establish" in the usual meaning, and in the meaning of the chapter title of the SRS. But if we refer to the requirement of the SRS, it should be "established and version checked" because the version check comes after the establishment. Nevertheless, for the sake of clarity I will use "establish" in its general use, which corresponds to the state EstabChecked.

- Not being able to establish a communication while required;
- Establishing a communication while not required;
- Establishing a communication with the wrong RBC;
- Establishing a communication with an incompatible version RBC;
- ...?

Considering the events studied in Sect. 3.1, we can state that the first item does not lead to an unsafe state. It will not yield a safety property. As for the second, the safety status is not as clear. It is necessary to conduct a system safety analysis in order to decide if failure leads to an unsafe state, and in this cas to state a property to cover this event (for the sake of brevity, we will not do it in this document). As for the third event, we will consider it as unsafe, because if we cannot guarantee that the same RBC is used during the whole process, it is unlikely that we will be able to perform the version control requested to cover the lattest issue. The last failure (establishing a communication with an incompatible version RBC) could clearly lead to an unsafe state, if possible (although a system-level analysis should confirm this).

Assuming these are all the safety event we have to cover, we could therefore provide the following property in order to cover them. This property is somehow redundant with the model, but it is restricted to the bare safety need.

$$State = Checked \Rightarrow compat(V\_Com) \wedge RBC\_ID = RBC\_Com$$

3.4 Filling the gaps...

If we compare the events provided by the Fault Trees (KERNEL-*xx*) it is clear that they are still too high level to be able to be used to build directly the safety properties on the studied function. These properties are still high level compared to the degree of description in the SRS. This is due to the fact that these events are *functional* although the function studied (establishing a communication) is a rather low level service.

The question is "would it be possible to go *formally* from the high level Dread Events to the properties which goes on the same level than the model?" In my opinion this is not practically feasible. It would need a very high level model of the system (much higher than the SRS) without any allocation to the different subsystem. It would also be necessary to introduce a sufficiently wide notion of environment to be able to formalize that communicating with an improper version of software could lead to an unsafe behavior.

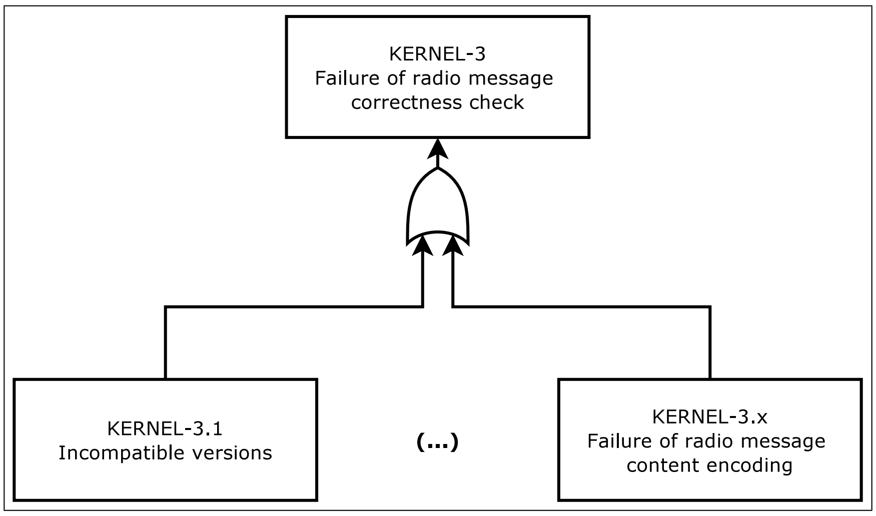

KERNEL-3
Failure of radio message correctness check
KERNEL-3.1
Incompatible versions
(...)
KERNEL-3.x
Failure of radio message content encoding

Figure 7. Extending the Fault Tree

From	To	Condition	Action
SB	IS	Isolated	
SB	FS	FS conditions and not Isolated	
SB	SH	SH conditions and not Isolated	

Figure 8. Mode transitions from SB to SH, FS and IS (transitions)

It is therefore necessary to conduct a system-level “brain, paper and pen” safety analysis to be able to build the safety properties on the proper level with regard to a model of the SRS. This analysis could be carried out either with a *top-down* analysis (cf. Fig. 7) or by a *bottom-up* analysis (*e.g.* FMECA), or by combining both approaches. It is important to point out that whichever is the followed method, it is the level of expertise of the safety analyst above everything else that will guarantee that no unsafe case have been forgotten.

4 Example 2: Mode Transitions

The purpose is to modelize some of the mode transitions of the SUBSET-26 4.6.2 table (mode transitions), namely the transitions from SB to SH, FS and IS.

4.1 Model of the function

The model here is very simple (Fig. 8 and 9) because we do not want to dive into the condition details.

4.2 Safety properties

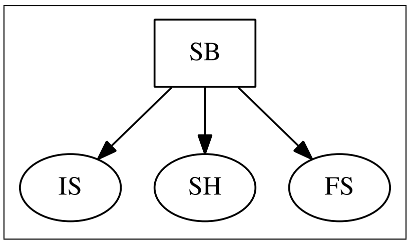


Figure 9. Mode transitions from SB to SH, FS and IS

From	To	Condition	Action
SB	IS	Isolated	
SH	IS	Isolated	
FS	IS	Isolated	
SB	FS	FS conditions and not Isolated	
SB	SH	SH conditions and not Isolated	
SH	SB	Exit SH conditions and not Isolated	
FS	SB	Exit FS conditions and not Isolated	
FS	SH	SH conditions and not Isolated	

Figure 10. Mode transitions with states SB to SH, FS and IS (transitions)

As for the safety properties, the SUBSET-88 provides the following event.

GATE80: Incorrect determination of the current mode

Without a clear hierarchy of modes stating that being in one given mode instead of other ones, we shall consider that as far as safety is concerned, a transition must occur if and only if the corresponding conditions are verified at the time of the transition.

We do not want to define here a complicated formalism. We use simple predicates $State(X)$ to write that the machine is in the state X, and $A \rightarrow B$ that the machine triggers a transition from state A to state B. The important thing here is that we are introducing some dynamic behavior (although we keep on using proposition calculus).

$$SB \rightarrow IS \Rightarrow Isolated\_by\_Driver$$

$$Isolated\_by\_Driver \wedge State(SB) \Rightarrow SB \rightarrow IS$$

$$SB \rightarrow FS \Rightarrow FS\_Conditions \wedge \neg Isolated$$

$$FS\_Conditions \wedge \neg Isolated \wedge State(SB) \Rightarrow SB \rightarrow FS$$

$$SB \rightarrow SH \Rightarrow SH\_Conditions \wedge \neg Isolated$$

$$SH\_Conditions \wedge \neg Isolated \wedge State(SB) \Rightarrow SB \rightarrow SH$$

These properties are a bit disappointing, because they are a mere paraphrasing of the state machine provided in the model. We could try to go a little bit further with a system-level analysis of the SUBSET-26 mode transition table. We would like to provide in the safety properties the conditions which are really relevant to safety (but these might as well be *all*).

For this task we need to consider the transition out of the states. For the sake of brevity, we consider only the four modes IS, FS, SB and SH (but we use the full sub-state machine of the real ETCS mode with these modes) and do not discriminate here the different Shunting conditions depending of if the initial state is FS or SB (Fig. 11 and 11).

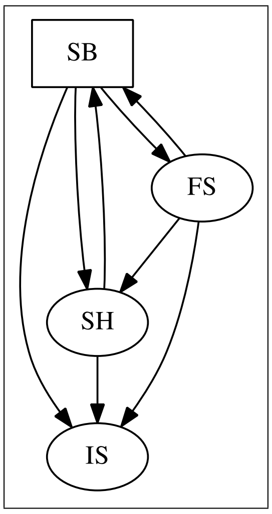


Figure 11. Mode transitions with states SB to SH, FS and IS

From the initial set of properties, it seems difficult to be able to remove some. We could add new ones, but it would need to be interesting from a system standpoint. For example we might want to prove that the OBU is *never* in isolated mode if not requested by the driver. We could write this as:

$$(SB \rightarrow IS) \vee (SH \rightarrow IS) \vee (FS \rightarrow IS) \Rightarrow Isolated\_by\_Driver$$

We could also state that the IS state is a "dead" state (state with no exits). This provides the following, for the subset of modes provided here.

$$\neg((IS \rightarrow FS) \vee (IS \rightarrow SB) \vee (IS \rightarrow SH))$$

This is of course redundant with the previous properties the state machine, but it provides a simple and understandable safety property that cover exactly the safety need.

We could try to do the same for the other modes, but it will be the same problem. Most of the safety behavior is yielded by the model itself and it is difficult to produce a set of declarative properties, and to avoid paraphrase. Nevertheless, it is useful to point out some specific and important system properties, even if they are not sufficient to capture the whole behavior of the state machine. In this case, these properties must come on top of the mere rephrase of the model.

Of course, it would also be possible to switch to a "'bigger scale" view, and state properties directly from the transition conditions to the behaviors allowed in the mode, but these properties would probably be difficult to prove without the intermediate layer provided by the modes themselves. And as far as modularity is considered, the properties and proofs would be difficult to maintain in case of evolution of the system.

5 Conclusion

The two examples studied (establishment of the communication and mode transitions) are not exhaustive in order to determine what would be the best formal language for safety properties (for this purpose it would be useful to study also the braking curves and the MA), but they are sufficient to outline the upper part (or "system" part) of the process that would be necessary in order to prove safety in the context of OpenETCS: it starts by determining what is Vital and

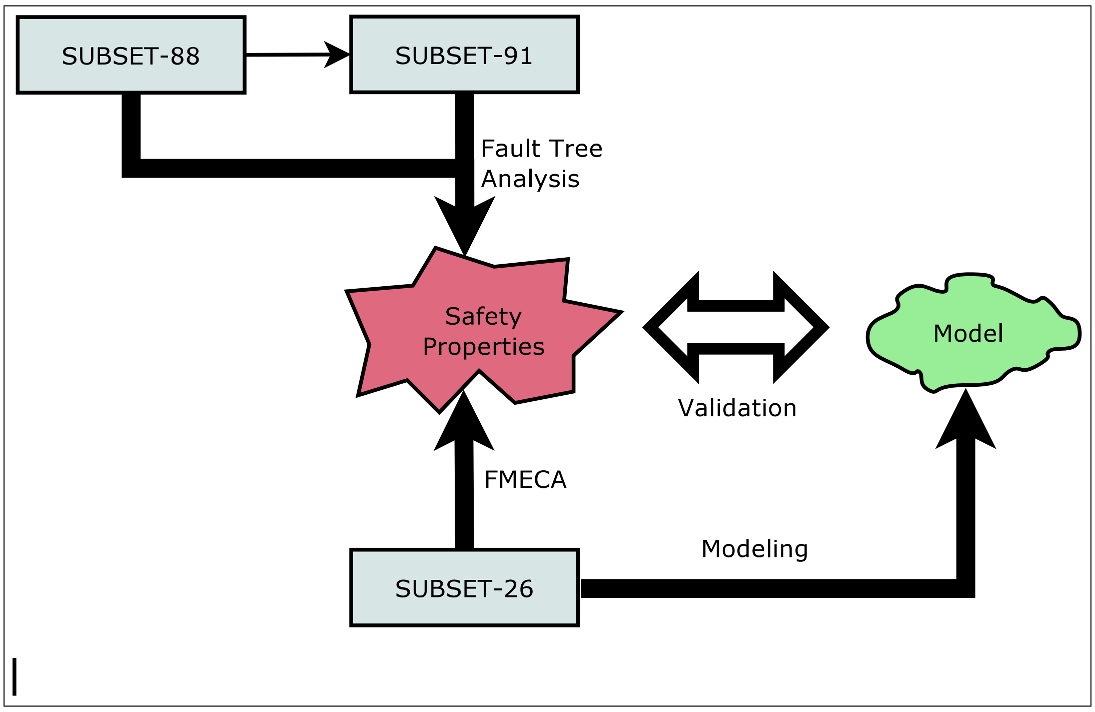


Figure 12. Outline of the safety analysis process

Non Vital, and by going down from the higher level safety properties to the model level safety properties. A synthetic view of the process is presented on Fig. 12

On these two examples, we have seen that in some cases, it only needs a few safety properties to capture the full safety behavior of the function, while in other cases, the full state machine (or a rephrase of it) is necessary. We have also seen than in the latter, it can also be useful to provide safety properties to express a particularly interesting (or sensitive) point.

In both case, the safety properties provide the backbone of the safety analysis. Writing them down allow to enhance knowledge and comprehension of the system, and they may also be used as Proof Obligations, as assertion to check when the model is running, or even in order to provide help in the generation (manual or automatic) of test cases.