COVER PAGE

ANALYSIS OF THE IMPACT OF COMMUNICATION PROTOCOLS ON SERVICE QUALITY IN ERTMS AUTOMATIC TRAIN CONTROL SYSTEMS

Paolo Lollini [1],
Leonardo Montecchi [1],
Melinda Magyar [2],
István Majzik [2],
Andrea Bondavalli [1].

[1]*Università degli Studi di Firenze, Dipartimento di Sistemi e Informatica*
Address: viale Morgagni 65, Firenze, Italy, I-50134
Phone: (+39) 055-4237460, Fax: (+39) 055-4237436,
E-mail:{lollini\bondavalli}@unifi.it, l.montecchi@alice.it

[2]*Budapest University of Technology and Economics, Department of Measurement and Information Systems*
Address: Magyar Tudósok krt. 2, Budapest, Hungary, H-1117
Phone: (+36-1) 463-3598, Fax: (+36-1) 463-2667,
E-mail: {mmelinda\majzik}@mit.bme.hu

Short abstract: The ERTMS-ATC system is a distributed system where the Driver Machine Interface (DMI) is a slave unit of the train onboard vital computer (EVC). Adopting a modular modeling methodology, in this paper we assess the impact of different types of EVC-DMI communication protocols (cyclic/acyclic) on significant dependability-related indicators.

Desired type of session: Regular session

Intended scope: model based approaches (using Stochastic Activity Networks) to analyze and evaluate RAMS parameters in the railway context.

ANALYSIS OF THE IMPACT OF COMMUNICATION PROTOCOLS ON SERVICE QUALITY IN ERTMS AUTOMATIC TRAIN CONTROL SYSTEMS

P. Lollini [1], L. Montecchi [1], M. Magyar [2], I. Majzik [2], A. Bondavalli [1]
[1]*Università degli Studi di Firenze, Dipartimento di Sistemi e Informatica*
Address: viale Morgagni 65, Firenze, Italy, I-50134
Phone: (+39) 055-4237460, Fax: (+39) 055-4237436, E-mail:{lollini\bondavalli}@unifi.it, l.montecchi@alice.it
[2]*Budapest University of Technology and Economics, Department of Measurement and Information Systems*
Address: Magyar Tudósok krt. 2, Budapest, Hungary, H-1117
Phone: (+36-1) 463-3598, Fax: (+36-1) 463-2667, E-mail: {mmelinda\majzik}@mit.bme.hu

Abstract: The ERTMS-ATC system is a distributed system where the Driver Machine Interface (DMI) is a slave unit of the train onboard vital computer (EVC). In this paper we analyze two types of communication protocols for the EVC-DMI interactions, based, respectively, on cyclic and acyclic messages' exchange. Adopting a modular modeling methodology, we assess their impact on significant dependability-related indicators. The analysis of the obtained results allows to quantify the improvements in adopting an acyclic protocol, possibly with one spare DMI component.
Keywords: communication protocols, design refinement, stochastic modeling, safety-critical systems, dependability analysis.

1. INTRODUCTION

The design of computer based systems often requires tradeoffs: different non-functional requirements like safety and availability, performance and reliability together with the cost constraints may call for design decisions that have conflicting consequences. For example, a safe operational mode may not provide the full service which decreases overall service availability, or a high performance mode may decrease reliability as components are stressed above the anticipated normal load. In safety-critical systems, it is widely accepted that safety is a design constraint that influences operational effectiveness, performance, and ease of use. However, common system safety techniques do not provide enough quantitative information for optimal decision making about risk management tradeoffs, and the design decisions that resolve the conflicting requirements are traditionally made on the basis of codes-of-practice and past experience. In this case the consequences of the decisions are evaluated late in the system validation phase, when tests and operational measurements could either justify or refute the decisions. In the latter case, the corrections and re-design phases will become costly and they will be performed only when clear evidences on the expected improvements are provided.

The goal of this paper is to assess the improvements that should be obtained re-designing a part of the ERTMS-ATC system. Shortly summarizing, in the first design of an Automatic Train Control (ATC) system a specific communication protocol has been introduced between two components: the train onboard vital computer (EVC), playing the role of the master, and the Driver Machine Interface (DMI), the slave. EVC uses messages to check the operational state of the slave. If the slave is not operational, then the master enters the safe mode (as a safety measure) and suspends the normal service, which can be resumed if the slave recovers. EVC and DMI interact through a communication protocol that is currently based on cyclic messages' exchange. In this paper we aim at quantifying the improvements in adopting a different communication protocol, based on acyclic messages' exchange, considering its impact on the quality of the service (QoS) provided by the whole system (the train). A sensitivity analysis will be performed considering some critical protocol's parameters (e.g., the periodicity of messages), and we will also assess the advantages in adopting different

recovery properties of the slave (e.g., the application of stand-by spare). This work has been conducted in the context of the SAFEDMI European project (SAFEDMI consortium, 2007), whose main objective is to design and develop a safe DMI being able to satisfy at least SIL 2 according to CENELEC specifications with all the related implications.

The structure of our paper follows the phases of a systematic model building and analysis process as follows. Section 2 presents the application context and the selection of measures to be evaluated on the basis of the system requirements. Section 3 describes the model construction process using Stochastic Activity Networks formalism, while some of the obtained results and the interpretation of the measures are presented in Section 4. Conclusions are finally drawn in Section 5.

2. SYSTEM CONTEXT

Railway automatic train control (ATC) systems are based both on track-side and on-board systems and their evolution has followed, step by step, the introduction of new technologies. The increasing level of train traffic and the spread of high-speed rail lines are now demanding an increasing safety level in the ATC systems. To ensure compatibility and inter-operability between the ATC systems produced in the various European countries, the European Rail Traffic Management System (ERTMS) programme has been set up to provide unique functional and non-functional standard requirements. The ERTMS architecture for the on-board ATC encompasses a Driver Machine Interface (DMI) component. Its main functions are the acquisition of driver's commands and the display of those information that support the train driving. Consequently, DMI is mainly made up of a display and a set of buttons for acquiring driver's commands; its functions and its ergonomic requirements are defined so to satisfy all the CENELEC related requirements (Cenelec, 2005). Railway Authorities require a DMI capable of visualizing data generated from different sources and producing safe visualization as drivers have to base their behavior on the displayed information. If DMI is not operational, then the vital on-board computer (EVC component) of the system enters the safe mode (as a safety measure) and suspends the normal service, which can be resumed if DMI recovers. In other words, when a DMI failure is detected by EVC, it triggers an emergency brake command and the train is stopped.

In the following section we provide a high-level description of the two types of communication protocols under analysis: the cyclic protocol and the acyclic one, recently proposed in the SAFEDMI context. It is important to note that the standard does not require explicitly the use of a cyclic protocol, and this is why we can envision the use of the acyclic one.

2.1 EVC-DMI interaction

The EVC-DMI interactions described in the following are mainly derived from UNISIG SUBSET-041 (2005) that contains a description of the functional interface specification of Man Machine Interface-ERTMS (i.e., DMI) expressed in terms of exchanged data between EVC and DMI. EVC and DMI communicate by exchanging the following types of information object.

From EVC to DMI:

- *CyclicInfo*. The *CyclicInfo* information object is periodically generated by EVC and contains information about the system. This information is shown to the driver.
- *AckRequest*. The *AckRequest* information object is generated in an aperiodic way and it requires specific actions from DMI and the driver.

From DMI to EVC:

- *DmiStatus*. The *DmiStatus* information object is generated by DMI after a *CyclicInfo* message is received, to inform EVC about its status.
- *AckReply*. The *AckReply* information object is generated by DMI after an *AckRequest* is received, processed and performed.

A typical **cyclic interaction** means a process in which EVC periodically sends *CyclicInfo* information object to DMI. The DMI system displays the information, produces sounds when required and replies to EVC with *DmiStatus* information object. The driver acts on the speed of the train and on the basis of the information shown on the display. In case of DMI failure, the *DmiStatus* message cannot be received by EVC that triggers an emergency brake command.

In a typical **acyclic** (or **acknowledge**) **interaction** the DMI receives from EVC an

AckRequest and shows the text message on the display according to the requested safe acknowledgement procedure. Once the driver is aware of the EVC request, he will press the proper keys. Finally, DMI acquires the driver's commands and answers EVC with an *AckReply* object. If the driver does not press the proper keys a DMI timeout occurs; as a consequence, DMI will not send the *AckReply* message and then the EVC timeout occurs as well (and the train stops).

2.2 System's behavior

In order to detect DMI failures the protocols define timeouts. When a timeout occurs, either from *AckRequest* (timeout in *Timeout_Request* seconds) or *CyclicInfo* (timeout in *Timeout_Cyclic_Upper* seconds), EVC considers DMI as failed and immediately performs an emergency brake, thus stopping the train, and will not return to normal operation until a DMI will be available. Spare DMIs can be used to increase system availability. Initially the system contains one working DMI and a number of *SpareDMI* spare DMI components (maybe zero, usually one). After the DMI failure is detected, a switching operation is performed to connect another available spare DMI that, once booted, becomes available. The switching operation may be achieved by an automatic switch or may be manually executed by the driver himself.

2.3 Measures of interest

The standard does not require explicitly that each *CyclicInfo* message has to be acknowledged by a *DmiStatus* message. However, the safety protocol designers have adopted this cyclic interaction, because in this way EVC is capable to detect the DMI failure earlier: note that without the *DmiStatus* reply, EVC could only detect the DMI failure when there is no *AckReply* answer to an aperiodic *AckRequest* (that needs driver's reaction).

The design decision of having *DmiStatus* messages or not is influenced by the following two aspects. On the one hand, the lack of *DmiStatus* messages (i.e., detection of DMI failure only in the case of the acyclic interaction), although it does not have direct safety-related consequences, has the above mentioned drawback: it does not satisfy the principle of detecting and handling the failure as early as possible. On the other hand, however, having only the acyclic interaction and no *DmiStatus* reply to *CyclicInfo* messages has availability related advantages: the chance of switch-over from a faulty DMI to a spare one between two aperiodic *AckRequest* messages (without disturbing the EVC) will be higher than in the case of the frequent cyclic interaction with regular *DmiStatus* messages (that have to be sent to the EVC). In this study we aim at *quantitatively evaluating* these consequences of the protocol design. In more detail, the system behavior has been analyzed through the evaluation of the following measures of interest.

1. The probability that the train will not successfully complete a mission (ride) of a given length (hours). The mission fails when an emergency break command is triggered by EVC. This measure will be presented in Section 4 as the number of interrupted rides over a total of 1000 rides.
2. The steady-state system availability, assuming that the train performs an infinite sequence of missions (rides) and accounting for the repair of the failed DMI components. This measure represents the steady-state probability that the train is correctly moving or, equivalently, that EVC is operational and no safe actions (emergency breaking) are triggered.
3. Probability (in percent) that a DMI failure causes a ride interruption. This measure is computed using the conditional probability $P(A|B) = P(AB)/P(B)$, where A is the event "the ride is interrupted" and B is the event "a DMI failure occurs".

3. MODEL DEVELOPMENT

The modeling process is based on the model composition approach for which the system is built from sub-models in a bottom-up fashion and the submodels have a limited view of the system. Several model composition techniques have been developed to support the systematic construction and validation of models (e.g., Sanders and Meyer, 1991; Rojas, 1996; Obal, 1998). All these techniques are supported by the composition operators offered by the corresponding modeling formalisms (Stochastic activity networks or Stochastic Well-formed Nets), and such operators are not available in

other formalisms that are commonly used for dependability modeling, e.g., GSPNs.

In this work we will use the Join composition operator (Sanders and Meyer, 1991) to compose system models based on Stochastic Activity Networks (Sanders and Meyer, 2001).

We have identified three basic submodels (called "atomic models" in SAN formalism) representing, respectively, the EVC behavior (**"EVC"** model), the DMI behavior (**"DMI"** model) and the EVC-DMI communications (**"EVC-comm-DMI"** model). Another model (**"Mission"** model) will be introduced only to enable the computation of the steady-state system availability. The overall model for the system under analysis will be obtained joining these basic submodels, sharing the places having the same name.

Note that the models representing the EVC and DMI behavior are independent from the selected communication scenario (cyclic or acyclic), while the communication model is based on the EVC-DMI interactions described in Section 2.1. EVC and DMI models incorporate knowledge of the failure modes and repair policy (*availability* design aspect), while the communication model includes the knowledge of the communication scenarios (*safety protocol* design aspect). This clear separation facilitates both the cooperation among different experts (as they can focus on the evaluation models that are related to their area of expertise) and the update of the models if the communication protocol or the repair policy are changed.

3.1 Modeling assumptions

The models have been built adopting the following assumptions:

- When DMI fails it is not possible to send a reply to the *CyclicInfo* message in time, because the timeout (a few seconds) is significantly smaller than the time needed to switch and to boot a spare DMI (which needs several minutes);
- Once a DMI becomes available, EVC immediately becomes operational.
- The time needed to switch between two DMI is uniformly distributed in the time interval [*SwitchTime_Lower*; *SwitchTime_Upper*].
- The time needed to boot a new DMI is normally distributed with mean *BootTime_Mean* and variance *BootTime_Variance*.
- EVC can issue a new *AckRequest* if the last *AckReply* message has been received or the corresponding timeout has occurred. The new *AckRequest* message will be issued after a time exponentially distributed with mean *AperiodicInterval*.
- Once an *AckRequest* message has been received, the *AckReply* message is sent after a delay uniformly distributed in the time interval [*DMIAckDelay_Lower*; *DMIAckDelay_Upper*].
- In the scenario in which the train performs an infinite sequence of missions (rides), all the failed DMI components can be repaired/replaced every *RidesBeforeRepair* rides (when the train is in a major railway station).

3.2 EVC atomic model

The EVC atomic model is depicted in *Figure 1*. The EVC component is initially in operational state (one token in place 'EVCOperational'), which holds until DMI replies are correctly received in time. When DMI has not been able to reply, a token arrives in place 'NoDMIReply' causing the ECV to perform an emergency brake and stop the train (place 'EmergencyStop'), until a DMI becomes again available ('DMIAlive'). The place 'BrakeCount' counts the number of emergency brakes in a single mission and it is used to measure the probability that a ride has been interrupted.

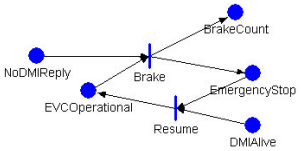

Fig. 1. EVC atomic model

3.3 DMI atomic model

The DMI atomic model is depicted in *Figure 2*. The model accounts for spare DMI components, if present, as well as for DMI failures and switching procedures among DMI. The place 'DMIWorking' initially contains one token (DMI in use) and any other available DMI is represented by a number of *SpareDMI* tokens in

place ‘Spare’. The exponential activity ‘Fail’ (with rate equal to the number of tokens in ‘DMIWorking’ divided by *DMI_MTTF*, where *DMI_MTTF* is the mean time to failure of DMI) fires when a DMI failure occurs and adds a token in each of the following places: ‘Broken’ to represent the status of the DMI involved, which turns from working into broken; ‘Failed’ to signal that a DMI has failed and a switch operation is needed; ‘NoCyclicReply’ to alert EVC that DMI has not been able to reply in time; ‘DMIFailCount’ which is used to count the number of DMI failures in a single mission and is needed to compute the probability that a DMI failure causes a ride interruption. After the failure the failed DMI is switched over to one of the others still available (‘Switch’ activity, uniformly distributed within the time interval [*SwitchTime_Lower*;*SwitchTime_Upper*]), the new DMI is booted (‘Boot’ activity, normally distributed with mean *BootTime_Mean* and variance *BootTime_Variance*) and then it becomes working. The place ‘DMIAlive’ is shared with the EVC model and it contains one token when a new DMI is available. A token in place ‘AllowRepair’ represents the state in which the train is in a major station and it can receive assistance, so all the failed DMI can be repaired. In this case the activity ‘Repair’ fires and a C code defined inside ‘OGRepair’ gate is executed (the code is not reported here for the sake of brevity, but it is available in Lollini *et al.*, 2008). As a result, a DMI becomes working and all the spare DMI are again available.

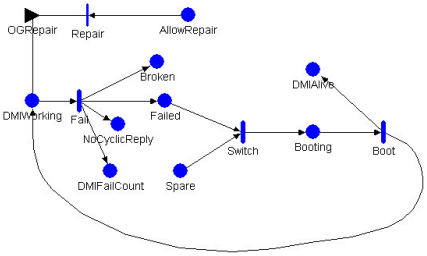

Fig. 2. DMI atomic model

3.4 EVC-comm-DMI atomic model

The atomic model representing the communications between DMI and EVC is presented in *Figure 3*. Initially, place ‘EVCOperational’ and ‘DMIWorking’ contain one token, while the others are empty. If EVC is operational (one token in place ‘EVCOperational’, shared with the EVC model) and no EVC requests are pending (place ‘AckRequest’ is empty), then the exponential activity ‘EVCSendRequest’ fires with rate 1.0/*AperiodicInterval*, thus representing the emission of the *AckRequest* message from EVC to DMI. If DMI is working (one token in place ‘DMIWorking’, shared with the DMI model), after a delay it sends an *AckReply* message to EVC (the activity ‘DMISendAck’ fires in a time uniformly distributed between *DMIAckDelay_Lower* and *DMIAckDelay_Upper* seconds). On the contrary, if DMI is not working the deterministic activity ‘AckRequestTimeout’ fires after *Timeout_Request* seconds and the DMI failure is signalled to the EVC model adding one token in place ‘NoDMIReply’, shared with the EVC model. The input gate ‘IGCyclicTimeout’ is used to enable or disable the cyclic interaction checking the value of a boolean variable (parameter *CyclicEnabled*); therefore we can use the same model to represent both the acyclic interaction and the cyclic one (this compact model has advantages in the model solution phase). When the cyclic interaction is enabled (*CyclicEnabled*==True) and one token is in place ‘NoCyclicReply’ (shared with the DMI model), then the activity ‘EVCCyclicTimeout’ fires in a time uniformly distributed between *Timeout_Cyclic_Lower* and *Timeout_Cyclic_Upper* seconds, thus revealing that DMI has not replied within the cyclic timeout. In this case the remaining time to timeout is uniformly distributed, since when DMI fails the time already elapsed from the receipt of the *CyclicInfo* message is uniformly distributed.

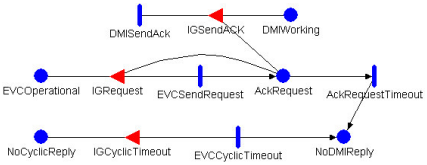

Fig. 3. EVC-comm-DMI atomic model

3.5 Mission atomic model

This additional model, depicted in *Figure 4*, has been introduced in order to compute the steady-state availability measure. Instead of considering a single mission only, that is a single train ride,

we suppose that once a mission is finished, a new one begins. Therefore we analyze a scenario in which the train performs an infinite sequence of missions, and the initial state of a new mission corresponds to the final state of the last one. While adding more details in the future, different mission types could be included.

Initially one token is in place 'Init', while the others are empty. The output function defined in the gate 'OGInitialize' is executed at the beginning of a new mission, and it puts a token in place 'AllowRepair' (shared with the DMI atomic model) every *RidesBeforeRepair* missions, to allow the repair/replacement of all the failed DMI. The mission ends when the deterministic activity 'MissionEnd' completes (in *RideLength* seconds). When a mission (ride) ends, the output gate 'OGMissionEnd' checks if the mission has succeeded (no tokens in place 'BrakeCount', shared with the EVC atomic model) or not, adds a token in place 'MissionCount' that counts the number of rides the train has completed, and then the mission is started over again putting a token in place 'GoodEnd' (if the mission has succeeded) or 'BadEnd'. The deterministic activities 'GoodMission' and 'BadMission' fires in 0.01 seconds and have been only introduced to allow the computation of the measure of interest (they have no influence on the system's behavior).

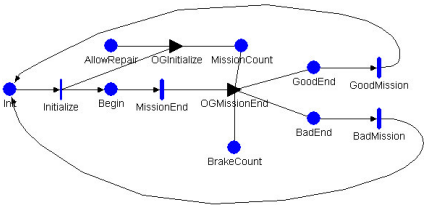

Fig. 4. Mission atomic model

4. MODEL EVALUATION

The system has been evaluated in order to investigate the influence of the cyclic and acyclic interaction between EVC and DMI on the dependability indicators identified in Section 2.3. *Table 1* shows the values we assigned to the main model parameters, following our experience and the constraints came from specification UNISIG SUBSET-041 (2005) defined by the European Railway Agency.

Symbol	Description	Default Value
AperiodicInterval	Mean time to issue a new *AckRequest* after the last *AckReply* has been received, or the corresponding timeout has occurred.	*variable* (sec.)
BootTime_Mean	Mean time to boot a new DMI	180 sec.
BootTime_Variance	Variance to boot a new DMI	1
CyclicEnabled	Boolean variable enabling (T) or disabling (F) the cyclic interaction	*variable* (T/F)
DMIAckDelay_Lower	Lower time bound to process the *AckRequest* and to send the *AckReply*	2 sec.
DMIAckDelay_Upper	Upper time bound to process the *AckRequest* and to send the *AckReply*	3 sec.
DMI_MTTF	Mean time to failure of DMI	1000 hours
RideLength	Duration of a single ride	8 hours
RidesBeforeRepair	Number of rides before all the failed DMI can be repaired/replaced	3
SpareDMI	Number of spare DMI components	1
SwitchTime_Lower	Lower time bound to switch between two DMI	30 sec.
SwitchTime_Upper	Upper time bound to switch between two DMI	60 sec.
Timeout_Cyclic_Lower	Lower time bound to have a cyclic timeout	0 sec.
Timeout_Cyclic_Upper	Upper time bound to have a cyclic timeout	3 sec.
Timeout_Request	*AckRequest* message timeout	5 sec.

Tab. 1. Relevant parameters and their values

Three evaluation scenarios have been set up in order to study the effects of the following three parameters: *AperiodicInterval* (mean time to issue a new *AckRequest* after the last *AckReply* has been received or the corresponding timeout has occurred), *RideLength* (duration of a single ride) and *DMI_MTTF* (mean time to failure of DMI). The scenarios have been analyzed considering the case in which both the cyclic and acyclic interactions are enabled ("cyclic and acyclic" case), and the case in which the interaction is only acyclic ("acyclic only" case, obtained by setting the *CyclicEnabled* parameter to False).

For the sake of brevity, in this paper we only present a subset of the experiments. The full set of analyzed scenarios can be found in Lollini *et al.*, 2008 (technical report).

The SAN models have been built and solved by simulation using Möbius (Daly *et al.*, 2000), a powerful multi-formalism/multi-solution tool. For each study we executed a minimum of 100000 simulation runs (batches), thus each result is the mean computed from a set of at least 100000 samples. Moreover, we set the relative confidence interval to 0.05 and the confidence level to 0.95. This means that the stopping criteria will not be satisfied until the confidence interval is within 5% of the mean estimate 95% of the time.

4.1 Analysis of the impact of *AperiodicInterval*

Figure 5 shows the number of interrupted rides for every 1000, considering different values for the *AperiodicInterval* parameter. This parameter represents the mean time between the receipt of the last *AckReply* message (or the corresponding *AckRequest* timeout) and the issue of a new *AckRequest* message. Considering the "acyclic only" case (without the cyclic interaction), the number of interrupted rides greatly decreases when larger request intervals are considered. For example, 2 rides are saved (they are not interrupted) passing from 1 minute to 3 minutes of *AperiodicInterval* duration. In case of a DMI failure, in fact, larger intervals allow the new DMI to be more likely switched over and booted before the *AckRequest* timeout expires. With the cyclic interaction enabled ("cyclic and acyclic" case), the number of interrupted rides is constant and the value is near the maximum obtained in the "acyclic only" case. Therefore, the cyclic message exchange causes the same number of interruption that would be caused by the acyclic interaction with a very small interval between *AckRequest* messages.

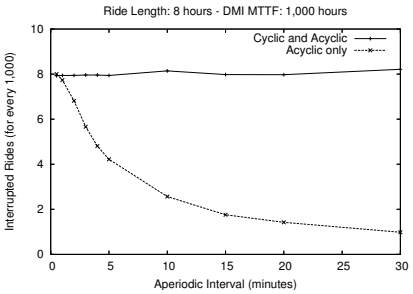

Fig 5. Number of interrupted rides for every 1000

Figure 6 shows the probability that a DMI failure causes a ride interruption with varying *AperiodicInterval* parameter and considering different values for the boot time parameter. Note that with the cyclic protocol enabled ("cyclic and acyclic" case), results are constant with value 100%, which means that every DMI failure causes an emergency brake (the cyclic period lasts only a few seconds and consequently no spare DMI can be switched and booted in the meanwhile). In the "acyclic only" case, larger intervals between requests lower this value in accordance with the results depicted in *Figure 5*. For example, 30% of the DMI failures do not cause anymore an emergency break passing from 1 minute to 3 minutes of *AperiodicInterval* duration. Another observation is that the faster is the boot process (characterized by *BootTime_Mean*, the mean time needed to boot a spare DMI), the lower is the probability that the DMI failure causes a ride interruption. For example, 20% of the DMI failures do not cause anymore an emergency break considering an *AperiodicInterval* duration of 3 minutes and halving the boot time duration (from 4 minutes to 2 minutes). This result justifies the design of a DMI with fast boot procedure.

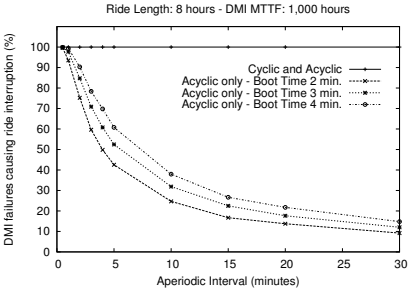

Fig 6. Probability that a DMI failure causes a ride interruption

Figure 7 shows the steady-state system availability with varying *AperiodicInterval* parameter, with and without the use of one spare DMI (in the "acyclic only" case). The system is considered unavailable if the train is stopped because EVC has not received the *AckReply* message from DMI within the fixed timeout, and available otherwise. Considering the spare DMI the system availability significantly improves (about 1% more).

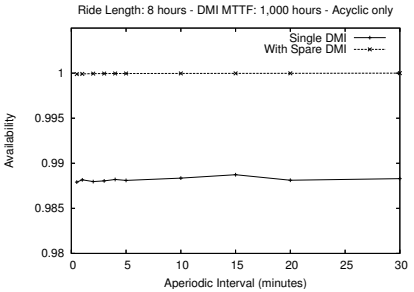

Fig 7. Steady-state system availability in the acyclic scenario

Finally, *Figure 8* shows the steady-state system availability varying the *DMI MTTF* parameter (mean time to failure of the DMI). Using more reliable DMI components, the system availability increases independently of the used protocols. The difference between the two plots comes from the early detection of the DMI failure event in case of using the cyclic protocol ("cyclic and acyclic" case), since in this case every DMI failure causes an emergency brake.

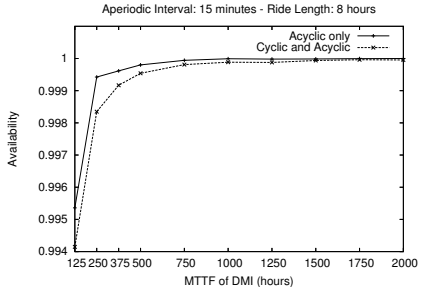

Fig 8. Steady-state system availability

5. CONCLUDING REMARKS

This work has focused on the interactions between two components constituting the Automatic Train Control system: the onboard vital computer (EVC) and the Driver Machine Interface (DMI). According to their functional interface specification, the two components mainly communicate by exchanging of different types of information objects that can be generated by EVC in a cyclic or acyclic way.

In this paper we have adopted a modular modeling methodology to assess the impact of cyclic/acyclic EVC-DMI interactions on a set of significant dependability-related indicators. The hierarchical and modular modeling methodology to capture the overall system behavior has been first described. Then, a sensitivity analysis based on simulation of the defined models in a transient period of time and in a steady-state condition has been performed to tune a part of the main critical system's parameters. The analysis of the obtained results allows to better understand the dynamics and the involved phenomena, and it points out the advantages in adopting the acyclic interaction, possibly with one spare DMI component. As such, it significantly contributes to the early design refinement of the architecture for the DMI system that is going to be specified in the SAFEDMI project.

ACKNOWLEDGEMENT

This work has been partially supported by the EC IST Project SAFEDMI (Contract n. 031413).

REFERENCES

CENELEC (2005). ERTMS - Driver Machine Interface.

D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster and W. H. Sanders (2000). Möbius: an extensible tool for performance and dependability modeling. In B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, editors, *11th International Conference, TOOLS 2000,* volume 1786 of LNCS, Springer Verlag, pages 332-336.

P. Lollini, L. Montecchi, M. Magyar, I. Majzik and A. Bondavalli (2008). Assessing the impact of cyclic/acyclic EVC-DMI interactions in Automatic Train Control systems. Technical Report rcl080401, University of Florence, Dip. Sistemi e Informatica, RCL group, http://dcl.isti.cnr.it/Documentation/ Papers/Techreports.html.

W. D. Obal (1998). Measure-Adaptive State-Space Construction Methods. PhD thesis, Univ. of Arizona.

I. Rojas (1996). Compositional construction of SWN models. *The Computer Journal*, 38(7):612-621.

SAFEDMI consortium (2007). IST-FP6-STREP-031413 Safe Driver Machine Interface (DMI) for ERTMS Automatic Train Control. http://www.safedmi.org/.

W. H. Sanders and J. F. Meyer (1991). Reduced base model construction methods for Stochastic Activity Networks. *IEEE Journal on Selected Areas in Communications*, 9(1):25-36, January 1991.

W. H. Sanders and J. F. Meyer (2001). Stochastic Activity Networks: Formal definitions and concepts. *Lectures on Formal Methods and Performance Analysis*, volume 2090 of LNCS, Springer Verlag, pages 315-343.

UNISIG SUBSET-041 (2005). Performance requirements for interoperability - 2.1.0.