Hierarchical modelling of complex control systems: dependability analysis of a railway interlocking

Andrea Bondavalli*, Manuela Nelli†‡, Luca Simoncini‡ and Giorgio Mongardi§

*DIS, University of Florence, Via Lombroso 6/17, I-50134 Firenze, Italy. Email: a.bondavalli@dsi.unifi.it
†CNUCE Istituto del CNR, Via Vittorio Alfieri 1, 56010 Ghezzano (Pisa), Italy
‡ANSALDO Segnalamento Ferroviario, Piossasco, Torino, Italy

This paper reports an experience made in building a model and analysing the dependability of an actual railway station interlocking control system. Despite our analysis has been restricted to the Safety Nucleus subsystem, mastering complexity and size required a considerable effort. We identified a modelling strategy, based on a modular, hierarchical decomposition allowing to use different methods and tools for modelling at the various level of the hierarchy. This multi-layered modelling methodology led to an accurate representation of the system behaviour and allowed us (i) to keep under control the size of the models within the different levels to be easily managed by the automatic tools, (ii) to make changes in the model in a very easy and cheap way. The paper contains also examples of the extensive analyses performed regarding the sensitivity of the dependability measures to variations of critical parameters and towards the validation of the assumptions made.

Keywords: computer based interlocking systems, analytical modelling and evaluation, hierarchical modelling methodology, unsafety, reliability, availability, sensitivity analysis

1. INTRODUCTION

Today critical systems like railway station interlocking systems employed in all technologically advanced countries are controlled by computers, mainly to cope with the increasing complexity of the control operations which is a source of failures. Several systems have been built [e.g.1–6], and have been used since a few years by those Railway Authorities wishing to have a good cost/benefit ratio. To ensure an adequate dependability of the systems coming to operation, many standards exists (like the new ERTMS [7–9]) which establish targets on the dependability attributes, like safety, reliability and availability, and prescribe methodologies for system specification, design, verification and validation.

As a consequence, evaluations are necessary for the system assessment and the fault forecasting. These evaluations may be performed both during the design phase, in order to test if the behaviour of the architecture under study is as expected, or to compare different solutions, and during the verification phase, to assess whether the developed system satisfies the assigned targets. System assessment can be performed using several approaches like testing, fault injection often combined with analytical models. The modelling approach is generally cheap for manufacturers and has proven to be useful in all the phases of the system life cycle. During the design phase, models allow us to compare different solutions and to select the most suitable one (among those obeying other design constraints), and to highlight problems within the design. In assessing an already built system, models allow to characterise specific aspects, to detect dependability bottlenecks and to suggest solutions to be adopted in future releases. In the literature several papers exist in the field of dependability analysis [e.g. 10–15]. In addition, many of the existing modelling techniques like
Markov Chains, Stochastic Petri Nets and Stochastic Activity Networks [16] are supported by tools like UltraSAN [17], SURF-2 [18] and others [19] to help in building and solving models. However, these computer controlled systems, compared to old electromechanical ones, pose non-trivial problems in their design and analysis. Most difficult are those parts of the systems where the interactions between the redundant hardware and the application software have a critical impact on system safety. These interactions influence the modelling complexity since they induce stochastic dependencies that must be taken into account in modelling the behaviour of components and their interactions.

Such complexity exacerbates also some problems with modelling that must be taken under special care. The first problem is complexity; in fact, although building models of simple mechanisms may be easy, the overall description of critical complex systems accounting at the same time for all the relevant aspects is not trivial at all. To master complexity a modelling methodology is needed so that only the relevant aspects can be detailed still allowing numerical results to be effectively computable: for instance, if a model specifies too many details, then the number of its states may explode giving raise to processing problems.

Models may need many parameters (the meaning thereof is not always intuitive for designers) and require us to determine the values to assign to them (usually by way of experimental tests) which may be very difficult. In addition, simplifying hypotheses are very often necessary to keep the model manageable; the choice of such hypotheses is critical. Making assumptions, on one hand, allows us to obtain simpler models, but, on the other, leads to approximations of the system behaviour. The resulting error should always be estimated, either through sensitivity analyses or by comparing the results returned by the model containing the assumption and a model where it has been released. A feasible modelling approach starts with simple models which are made more and more complex and detailed by releasing those assumptions having an unacceptable impact on the obtained results.

We have experience in building a model on an actual critical system. This system, developed by Ansaldo Trasporti, is a railway station interlocking control system: the ACC CIS (Computer Interlocking System) [4], is made of redundant replicated hardware and redundant diverse software. In this paper we describe our modelling experience taking into account all aspects of their interactions (including correlation between the diverse software variants) and of the criticality of the several components. Our approach has been to realise the system model in a structured way. This allows us to cope with complexity and to focus, where interesting, on specific behaviour for a more detailed analysis. Structuring in different levels separated by well identified interfaces allows to realise each level with different methodologies and to perform its evaluation with different tools without the need of modifying the general structure of the model. Each level has been subdivided into several sub-levels for a finer analysis of some characteristics. Despite our effort for reducing the complexity of the individual levels, some of them remained complex: these have been realised using different methodologies to compare and validate the used model. The paper contains also examples of the extensive analyses which have been performed performed regarding the sensitivity of the dependability measures to variations of critical parameters and towards the validation of the assumptions made.

The paper is organised as follows. Section 2 contains an overview of the Ansaldo ACC CIS system; Section 3 contains a description of our assumptions, the meaning of the basic parameters used and describes and of our modelling approach. Sections 4 and 5 contain a description of the various models for one execution and for the mission, respectively. Section 6 contains a few examples of the evaluations we performed: measurements of the sensitivity of dependability attributes at varying critical parameters and of the effects of realising assumptions. Finally Section 7 concludes this paper.

2. ANSALDO ACC CIS SYSTEM

The ACC CIS system [4] is constructed by Ansaldo for railway station signalling control system. The CIS is structured in two subsystems as shown in Figure 1: one devoted to vital functions, the other to supervision ones. The former, called Vital Section, is the subsystem which performs vital functions: it comprises the (Safety Nucleus, SN) and a number of Trackside Units (TU), depending on the station size, that communicate state of the station to the SN through a proprietary serial bus. The latter performs Operations and Indications, and Alarm, Recording and Telecontrol functions (OI-ART), it is made of a number of processing nodes, connected through a LAN, and is located close to the SN in the Central Post. This subsystem allows continuous monitoring of the system state and events recording useful to make estimations and find out less reliable sections.

The Safety Nucleus [20] is the core part of the system and its structure is partially reported in Figure 2. It comprises six units with a separated power supply unit. The three Nsi sections represent three computers which are connected in TMR configuration, i.e. working with a ‘2-out-of-3’ majority: three diverse software programs performing iteratively the same tasks, run inside three identical hardware sections. The Exclusion Logic is a fail-safe circuit whose job is to electrically isolate the section that TMR indicated to be excluded. The activation/de-activation unit is a device that switches on and controls power supply units. The video switching unit controls video images to be transmitted to the monitors of the operator terminal.

The system is designed to keep on running even after the failure of one section. In such case the section is excluded.
and the system is reconfigured to run with only two sections with a ‘2-out-of-2’ majority, until the failed section is restored (usually after a few minutes). If a disagreement is found while all three sections are active, the disagreeing section recovers the correct value and participate to the next loop, if one section disagrees twice in a row it is excluded. No disagreement is tolerated when only two sections are active. The TMR sections carry out the same functions; the hardware, the basic software architecture and the operating environment are exactly the same, while ‘design diversity’ [21] was adopted in the development of software application modules. Each section is composed by two physically separated units which carry out different functions in parallel:

- **GIOUL** (operator interface manager and logical unit): executes the actual processing and manages the interactions with the Operator Terminal and the OI-ART subsystem;
- **GP** (trackside manager): manages the communications with the Trackside Units and modifies, whenever necessary, the commands given by GIOUL.

The processing loops last 1 second for GIOUL and 250 msec for GP: this causes the communications between GIOUL and GP belonging to the same TMR section to be performed at every second (GIOUL loop), i.e. once every four loops of GP. Instead the communications between GIOUL and separately between the three GP units) are carried out at every processing loop. The databases, whose control is performed by the GIOUL, the acceptance test on the input from the Operator Terminal. Hardware aspects cover internal boards and physical characteristics of the communications while software aspects cover the operating system and software modules that are sequentially activated during the processing loops. The databases, whose control represents one of the ways for detecting errors in various modules, cover both hardware and software aspects: database malfunction can be due to either corruption of memory cells or an error of the managing software. One of the tasks that GIOUL has to perform is checking the correctness of the inputs issued by the operator terminal keyboard before transmitting them to the other modules for their processing; this check is very important since it can avoid the system to send wrong commands to the Trackside Units. For this reason the software module performing this check is not considered together with the other software modules of GIOUL. We also made the choice of not modelling in detail the system while an excluded section is restored. More precisely we account for the time required for restoring a section but we neglect the particular configurations that GIOUL and GP can assume during that period.

Despite our decision to restrict to the Safety Nucleus a considerable effort has been necessary in order to account for (i) the complexity of the system, (ii) a proper trade-off between detailing the relevant mechanisms and an explosion of the model, (iii) the need for simplifying assumption and (iv) evaluation of the errors introduced with the consequent requirement to release some of the assumptions. We identified a modelling strategy based on a modular, hierarchical decomposition such that (i) different methods and tools may be used for modelling at the various level of the hierarchy selecting the method which appears to be the most appropri-
ate, (ii) each model is small enough and does not result in computational explosion, (iii) specific aspects are confined in a few sub-models and modifications do not require to redefine the model completely. This has proven useful for analysing the impact of the several assumptions and to decide which could be acceptable and those that had to be relaxed.

To start with, we have built a simple model adopting the following simplifying assumptions:

1. ‘Compensation’ among errors never happens;
2. The Video Switching, the Activation/de-Activation and the (external fail-safe) Exclusion Logic units are considered reliable;
3. The module that exploits majority voting within GIOUL and GP is considered reliable;
4. The Exclusion Management module within GIOUL and GP is considered reliable;
5. Identical erroneous outputs are produced only by correlated errors, while independent errors in the different units are always distinguishable by the voting.
6. Symmetry: the error probabilities of GIOUL and GP are the same for the three sections.
7. The hardware communication resources of the Nucleus are considered together with the other hardware aspects; the software dealing with communications is assumed reliable.
8. Errors affecting different components of the same unit (GIOUL or GP) are statistically independent.
9. During one execution, both GIOUL and GP may suffer from many errors, at most one for each component (software, hardware, databases and acceptance test for GIOUL).
10. The execution of each iteration is statistically independent from the others.
11. GIOUL units receive identical inputs from the keyboard.

Then, many of these assumptions have been released (one at a time) to check their impact. In the rest of this paper we will show our analyses concerning the release of the assumptions 5 and 3. While assumption 5 has been completely released by simply considering the event ‘two or three wrong results caused by independent errors may constitute a wrong majority’, we have substituted assumption 3 with more pessimistic assumptions about the voter behaviour. First we have considered the assumption 3a and then the assumption 3b which is more conservative:

3a. When the voter fails, (i) if a majority exist the voter does not not recognise it (ii) if a majority does not exists, the voter selects a wrong result.
3b. If the voter fails, independently from the existence of a majority, it selects a wrong result.

The definition of the basic events we have considered and the symbols used to denote their probabilities are reported in Table 1.

### 3.2 Modelling approach

The modelling approach is strongly related to the type of system considered where we have synchronous periodic executions of GIOULs and GPs. Each iteration can be modelled independently from the modelling of a complete mission. The model was conceived in a modular and hierarchical fashion, structured in layers. Each layer has been structured for producing some results while hiding implementation details and internal characteristics: output values from one layer may thus be used as parameters of the next higher layer. In this way the entire modelling can be simply handled. Further, different layers can be modelled using different tools and methodologies: this leads to flexible and changeable sub-models so that one can vary the accuracy and detail with which specific aspects can be studied. The specific structure of each sub-model depends both on the system architecture and on the measurements and evaluations to be obtained. The model of the Safety Nucleus of the Ansaldo CIS system we have built, shown in Figure 3, can be split into two main parts: the first part deals with one execution

<table>
<thead>
<tr>
<th>Error type (Events)</th>
<th>Symbol (GIOUL)</th>
<th>Symbol (GP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>independent error in a unit caused by hardware fault</td>
<td>qhl</td>
<td>qhp</td>
</tr>
<tr>
<td>an error caused by hardware fault is not detected by the diagnostics</td>
<td>qbd</td>
<td>qbd</td>
</tr>
<tr>
<td>spurious error: the diagnostic errs detecting a (non-pre-sent) error due to independent hardware fault</td>
<td>qhnl</td>
<td>qhnp</td>
</tr>
<tr>
<td>an independent error in a database is detected</td>
<td>qbr</td>
<td>qbr</td>
</tr>
<tr>
<td>an independent error in a database is not detected</td>
<td>qbnr</td>
<td>qbr</td>
</tr>
<tr>
<td>correlated error between three databases</td>
<td>q3bd</td>
<td>q3bd</td>
</tr>
<tr>
<td>correlated error between two databases</td>
<td>q2bd</td>
<td>q2bd</td>
</tr>
<tr>
<td>independent software error in a unit</td>
<td>q3l</td>
<td>q3p</td>
</tr>
<tr>
<td>correlated software error between three units</td>
<td>q3al</td>
<td>q3p</td>
</tr>
<tr>
<td>correlated software error between two units</td>
<td>q2al</td>
<td>q2p</td>
</tr>
<tr>
<td>independent error of the acceptance test in a unit (it may either accept a wrong input or reject a correct one)</td>
<td>qail</td>
<td>.....</td>
</tr>
<tr>
<td>correlated error between the acceptance tests of three units accepting the same wrong input</td>
<td>q3al</td>
<td>.....</td>
</tr>
<tr>
<td>correlated error between the acceptance tests of two units accepting the same wrong input</td>
<td>q2al</td>
<td>.....</td>
</tr>
<tr>
<td>two independent software errors indistinguishable by the voter (only in the model where assumption 5 is released)</td>
<td>qdl</td>
<td>qdp</td>
</tr>
<tr>
<td>at least two of three independent software errors indistinguishable by the voter (only in the model where assumption 5 is released)</td>
<td>qdtl</td>
<td>qdtl</td>
</tr>
<tr>
<td>independent error of the voter (only in the model with the assumptions 3a and 3b)</td>
<td>qvl</td>
<td>qvp</td>
</tr>
</tbody>
</table>
and computes the probabilities of success or failure; the second one, building on this, allows the evaluations of the dependability attributes for an entire mission.

In the previous section we explained that if a disagreement is found while all three sections are active, GIOUL and GP can identify and recover the correct value and participate to the next loop. This holds for one single disagreement: if one GIOUL or GP disagrees twice in a row the entire section (GIOUL AND GP) is excluded at the end of the current loop. Therefore, in order to represent as close as possible the actual system behaviour, we had to make several models to keep memory of previous disagreement of one section at the beginning of the execution. To describe the GIOUL and GP TMR units at level 1 (Figure 3) we defined:

- five sub-models of the behaviour of GP in configurations 3h, 3h.1, 3h.2, 3h.3, 2h (3h.x means that section x (1, 2 or 3) disagreed during the previous loop);
- five sub-models of the behaviour of GIOUL (3h, 3h.1, 3h.2, 3h.3, 2h).

One system execution (level 2) lasts 1 second, it is composed by one GIOUL (1 second) and four GP (250 msec) iterations and could also be considered as brief one-second mission. Due to the need to keep memory of previous disagreements, the system can be in 17 different states and 17 different models for one-execution have been defined, one model the system when only two sections are active, while the remaining describe the system with three active sections (level 2 in Figure 3):

- 3h/3h: GIOUL and GP are correctly working at the beginning of the execution.
- 3h,x/3h: the GIOUL of section x (1, 2 or 3) disagreed during the previous loop while GP is correctly working.
- 3h/3h,y: GIOUL units are correctly working while the GP of section y (1, 2 or 3) dis-agreed during the previous (GP) execution.
- 3h,x/3h,y: both the GIOUL of section x and the GP of section y disagreed during the previous loop (x and y can represent the same section or different ones)
- 2h/2h: execution begins with only two active sections.

Each of the 17 models uses, in different combinations and sequences, the same basic objects of level 1 and describes the essential characteristics of the Safety Nucleus. The models of level 2 are conceived to compute (and to provide to level three) the following probabilities:

- probability of safe failure of ‘one-execution’: it is the probability that the system fails during one execution and stops avoiding catastrophic damages (this is ensured by the ACC system that is designed so that it stops when malfunctions occur, forcing devices and subsystems to lock in a safe state).
- probability of catastrophic failure of ‘one-execution’: it is the probability that the Nucleus, failing, keeps on sending erroneous commands causing serious damages.
- probability of success of ‘one-execution’: it is the probability that the system performs an entire one second mission correctly. This implies that the system is ready to start the next execution. It is actually composed by the probabilities to reach one of the 17 possible configurations.

The mission model (level 3), accounting for all the models of the system related to one execution, describes the system behaviour during time. Once that both the single execution and the mission models have been constructed we focused on which kind of measurements are required. For our highly critical system reliability, safety and availability have been evaluated. While reliability and safety can be both obtained by computing the probabilities of catastrophic and safe failure at time \( t \) defined as the duration of the mission, availability required the definition of a specific availability model.

4. MODELS FOR ONE EXECUTION

Two methodologies have been adopted to build the models for ‘one-execution’: Discrete time Markov chains that have been manually drawn and the probability evaluation has been accomplished using ‘Mathematica’, and Stochastic Activity Networks that have been directly solved using the software tool ‘Ultrasan’ [17]. Since Markov chains are often impractical, even if they provide symbolic results, Ultrasan has been adopted in order to avoid building 17 repetitive models using only Markov chains. Only two models (3h/3h and 2h/2h) have been completely built using Markov chains in order to test and validate the results obtained by Ultrasan. This redundancy in building models has been very useful: some errors occurred during the model developing phase have been detected. Ultrasan has been a good choice, since we could develop one single model that allowed to compute the results for the 17 different scenarios. In fact, by assigning different values to the variables of the model, thus representing different initial markings, we could represent the different states of the system and account for previous failures of the various sub-components. The model is also able to distinguish the various configurations without having to replicate the unchanged aspects. Only two of the seventeen models were tested using Markov chains but those two models are the most relevant ones and cover all the scenarios that need to be represented. The results obtained by Ultrasan and Markov, using the same values for the parameters, were in

vol 16 no 4 july 2001 253
agreement. Now we show, as an example, some objects belonging to the two lower levels. First the Markov chains are described and later the SAN model.

As an example of a model of the level 1 (base object level), we report in Figure 4 the detailed description representing the behaviour of the GIOUL TMR in configuration 3h. The description of the states of such model is given Table 2, while the transition probabilities are derived as combinations of probabilities of the basic events reported in Table 1. The models for other configurations can be obtained by analogy with this. The Markov chain in Figure 4 is organised into five levels plus the final states: 3H, 3H.1, 3H.2, 3H.3, 2H (success states), SF and CF. Level one involves the hardware aspects, the level two the diagnostic tests carried out on boards and communications channels (hardware). Level three checks databases; level four investigates the behaviour of the acceptance test on the keyboard input whereas level five involves software. It should be noted that the model doesn't represent the timing relations among the various events.

The next higher level (level 2 of Figure 3) is not concerned with this detailed view, from its perspective it is just necessary to observe that the GIOUL TMR, performing one execution, can jump from the initial 3h state into all the other
GIOL TMR successful configurations or into the failure (safe or catastrophic) states according to the transition probabilities resulting by the resolution of model in Figure 4. Thus its view can be represented by the Markov chain in Figure 5 where all the paths, that go from the initial state to each of the final states, are substituted with one arch and the relative transition probability.

Once all the models of level 1 have been obtained and the related transition probabilities computed, these objects are composed into the several models for one system execution (level 2 Figure 3). Also for this level, only the description of one of the 17 configurations is provided; the remaining 16 can easily be obtained by analogy. This time we proceed in a top-down approach to show our models by showing first what is the viewpoint of the next level, the mission level (level 3 of Figure 3). Figure 6 depicts the black-box model of configuration 3h/3h showing just the transitions to other configurations after one execution and their probabilities. Of course, in order to obtain the transition probabilities mentioned above it is necessary to explode the arcs which go from the ‘3h/3h’ state to the final states and describe the system behaviour when one iteration starts with the GIOL and GP TMR working perfectly. This explosion, however, is not shown for the sake of space.

The system model obtained with Ultrasan allows to represent all the 17 configurations of one execution. Also in modelling using Ultrasan we started following a modular approach, building first the basic objects (level 1 of Figure 3) and then putting them together in the system model for one execution. Unfortunately, however, the ‘Composed Model’ mechanism offered by Ultrasan to define separated models and then join them slows down the execution speed of the compound model. Thus we decided not to take advantage of this opportunity but preferred to speed up the evaluations as much as possible.

Figure 7 shows the model that we actually used. Despite it can just provide an idea of the size and complexity, the existence of two sub-models is visible: the upper part represents one iteration of GP and it is executed four times, the lower part represents one iteration of GIOL executed once. An additional general problem to the understanding of the behaviour of models built using Ultrasan derives from the extensive use of C code that is hidden into the gates. (This model was evaluated using the Ultrasan transient solver).

### Table 2 Description of the states of the Markov chain depicted in Figure 4

<table>
<thead>
<tr>
<th>State notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>3h</td>
<td>start of GIOL 3h</td>
</tr>
<tr>
<td>h0</td>
<td>not hardware fault</td>
</tr>
<tr>
<td>h1</td>
<td>hardware fault into an unit</td>
</tr>
<tr>
<td>h2</td>
<td>hardware fault into two units</td>
</tr>
<tr>
<td>db hw0</td>
<td>not hardware diagnosed fault</td>
</tr>
<tr>
<td>db hw1</td>
<td>hardware diagnosed fault into a unit</td>
</tr>
<tr>
<td>db sf</td>
<td>diagnostic tests detected failure due to hardware faults</td>
</tr>
<tr>
<td>3db 1nr</td>
<td>hardware not diagnosed fault that appears as a processing error</td>
</tr>
<tr>
<td>ac hd0</td>
<td>hardware and databases are correct</td>
</tr>
<tr>
<td>ac hd1</td>
<td>a unit is already failed because of a hardware or database fault</td>
</tr>
<tr>
<td>ac sf</td>
<td>two or three units are already failed because of hardware or database faults</td>
</tr>
<tr>
<td>ac cf</td>
<td>catastrophic GIOL TMR failure due to database faults</td>
</tr>
<tr>
<td>3ac 1nr</td>
<td>hardware or database not diagnosed faults that appear as a processing error</td>
</tr>
<tr>
<td>swa0 hd0</td>
<td>hardware, databases and acceptance test are correct</td>
</tr>
<tr>
<td>swa1 hd0</td>
<td>a unit is already failed because of a hardware or database fault</td>
</tr>
<tr>
<td>sw sf</td>
<td>two or three units are already failed because of hardware, database or acceptance test faults</td>
</tr>
<tr>
<td>sw cf</td>
<td>catastrophic GIOL TMR failure due to database or acceptance test faults</td>
</tr>
<tr>
<td>3sw 1nr</td>
<td>hardware or database not diagnosed fault that appears as a processing error</td>
</tr>
</tbody>
</table>

5. MODELS FOR THE MISSION AND MEASURES

Also the model for a mission has been developed using both Markov chains and SANs. Modelling a mission using discrete time Markov chains does not allow to account for time parameters, thus continuous time Markov Chains have been used. The obtained model is approximate. In fact, not only it uses exponential distributions in place of the deterministic ones (this is not a problem since we use a very long mission time with respect to the time required from one execution), but the main approximation comes from the fact that the n states ‘2h/2h’ in the hypothetical discrete time Markov model are compressed in only one state ‘2h/2h’. The output rates of this compressed ‘2h/2h’ state approximate the behaviour...
of the system (averaging both time and probabilities) [20]. The continuous time Markov model defined is partially depicted in Figure 8 only to give an idea on the state transitions. This model allows to evaluate the probabilities of safe or catastrophic failure of missions of a given duration. The model has been solved obtaining the system of differential equations [22]. The solution was obtained using Mathematica: it provided both symbolic and numerical results. The solutions we have found are complete since they allow to evaluate the probability of safe or catastrophic failure over any given time interval t.

To validate this approximated model we compared the results obtained with those returned by a mission model built using Ultrasan (where time distribution is still exponential). The Ultrasan model is structured into two layers: the lower layer is composed by a number of sub-models each one representing a given system configuration, and an higher level (‘Composed Model’) that joins the sub-models as required. For the model of a mission we did use the Composed Model without slowing down the execution speed because each sub-model already contained timed transitions at the beginning.

Figure 9a shows the sub-model representing the configuration ‘3h/3h’: the other sub-models are very similar. In Figure 9b the ‘Composed Model’ is shown; each box indicates one sub-model of one-execution and the ‘join’ box links them together. The two mission models (with the Markov chains and with SANs) have been tested against each other on the same input data for variable mission duration up to one year.

The results provided are of the same magnitude as soon as t reaches 1000 seconds, and for t from one day up they can be considered identical; as t grows, the number of identical significant digits increases and even exceeds the desired accuracy. While we preferred to use the Ultrasan model to obtain the results for one execution (and used a Markov one to validate it) because of the possibility to represent all the different configurations within a single model, here we have done the opposite: we used the model based on Markov to compute the results and the one based on SANs to validate it. We preferred to use Mathematica and the Markov model to achieve results at a higher speed: only few minutes are required to provide the results for one year missions while our Ultrasan model requires several days running on a SUN 10 workstation.

Once the probabilities ‘f’, of safe failure and ‘fc’, of catastrophic failure in one year have been obtained, it is immediate to find out the reliability and safety measures while to obtain availability measures it is necessary build an availability model which is represented by the continuous time Markov model illustrated in Figure 10. When the system is in state F, it is operating and provides a correct service. R and C indicate the repair states: the system is not available following a safe failure (R) or a catastrophic failure (C). The system resuming rates are ‘m’, following a safe failure, and ‘mm’, following a catastrophic failure. ‘l’ indicates the system activation rate.
6. EVALUATION

6.1 Test plan and values of the input

Once the several models have been built, many different analyses can be performed. Examples are (i) checks on whether the system meets its requirements, (ii) analysis of the sensitivity to the various parameters, (iii) evaluation of the impact of various assumptions on the dependability figures.

Focusing on the probability of failure (reliability) and of catastrophic failure (safety), we show first the sensitivity to the main parameters (hardware, software) and the sensitivity to the parameter representing the software correlation; then we show the system behaviour when assumption 5 is released and finally the effects of substituting assumption 3 with 3a and 3b. Due to space limitations we do not report on all the evaluations carried out [23] but show just a few examples. The input variables for the set of models can be split in those necessary to the models representing one execution and those for models of the mission. The models of one execution require the probability values of the basic events (described in Table 1), while those related to the mission require (i) the results of the evaluation of one execution, (ii) the repair time for restoring an excluded section (set to 15 minutes), (iii) the mission time (1 year), and (iv) the recovery time (only for the availability model). Since there are many input variables, reasonable values have been a priori assigned to all the parameters. The probability values of the basic events (on a period of one hour of observation) are reported in Table 3.

In order to reduce the number of parameters in our plots we assume the probability of software independent faults to be the same for GIOUL and GP, qil = qip = qsw, where qsw varies in the range 1E-3÷1E-5; besides qvl = qvp = qv and qdl = qdp = qd. Furthermore, as it is shown in Table 3, some of the parameters for correlated software errors have been assigned values dependent by the probabilities of indepen-
dent errors and by a parameter representing the correlation factor. The default value assigned to this parameter corresponds to the assumption of a positive correlation among software errors. In our experiments the parameters for the mission models were given the following values: the repair time for restoring an excluded section has been set to 15 minutes, the mission duration set to 1 year, the system activation rate \( l \) to 1 per hour, the repair rate following a safe failure, \( m \), to 2 per hour and the repair rate following a catastrophic failure, \( mm \), to 1/2 per hour.

### 6.2 Measurements with variable software error probability

Figures 11, 12 and 13 show dependability measurements as a function of the probability of software independent error. The range goes from 1E-3 to 1E-6 per hour; curves for different values of the hardware error probability per unit are reported.

Observing the shapes of the curves in Figure 11, it is clear that reliability is sensitive to variations of the software error probability if hardware quality is good enough (ranging from 1E-5 to 1E-6), instead it becomes more and more insensitive as the hardware error probability increases. The curves also point out that decreasing the probability of software error over 1E-5 (that is improving the quality of the software) is practically useless without decreasing all the remaining system parameters at the same time as well. In fact, while in the left side of the figure, from 1E-3 to 1E-5, it can be observed that the reliability improves, it remains approximately constant for the values in the right side.

The shape of the curves representing the probability of catastrophic failure, shown in Figure 12, appears quite different. Safety is equally sensitive to the probability of independent software error: the more the software error probability decreases and the more the safety improves. Of course, this is valid under certain conditions, that is as long as the remaining parameters are kept unchanged and the software error probability is kept over a certain value. The same figure also suggests that the safety seems completely insensitive to variations of the hardware error probability: in fact, it is difficult to distinguish among the four curves shown. It should also be noted that the system satisfies its target (that for such systems is usually 1E-7 per year) if independent error probability values are less than or equal to 1E-5.

Figure 13, the only figure on availability, clearly points out that the availability is very high and almost constant when the software error probability ranges form 1E-4 to 1E-6, while it is a bit worse for higher values. The figure shows also that there are almost no variations of the availability for values of the hardware error probability equal or better than 1E-4. In any case the system is never affected by availability lacks, since the target (5 minutes unavailability over 8600 hours, i.e. 0.9999995) is satisfied in all the considered range of the software error probability. In short, within our parameters setting, the software must be regarded as a critical factor for safety, while it appears of almost no concern for availability.

### 6.3 Sensitivity to the correlation

We show how dependability measures are affected when changing the relationship linking the correlated error probability to the independent error probability (per execution) in the model representing the system with all the assumptions previously made. The coefficient (corr) has been varied in the range 1E+0 to 1E+4, for three values of the software error probability: 1E-3, 1E-4 and 1E-5. Figure 14 contains three curves (marked F) representing the probability of failure (reliability) and three curves (marked FC) representing the probability of catastrophic failure (safety). It clearly shows the different shapes assumed by the reliability curves with respect to the safety ones. The reliability curves depend on qsw the software error probability and on ‘corr’. For qsw
It appears that the safety values do not change much over all the range. The only significant variations are obtained for high values of qd and the variations are more significant for lower values of corr. In fact, white-marked curves (corr = 1) are more sensitive than black-marked ones (corr = 10). This appears only for very high values of qd, for low values (which are more likely to be expected) the variations are minimal. For high values of corr no sensible effect can be appreciated even for qd = 1. This study suggests that, in the interesting range of values of the input parameters, assumption 5 can be acceptable since it has a minimal impact on safety, the most critical dependability attribute.

6.5 Comparison of two alternatives for substituting assumption 3

Assumption 3 – the module that exploits majority voting within GIOUL and GP is considered reliable – has been substituted with more pessimistic assumptions about the voter behaviour, introducing a probability qv that the voter fails. First we have considered assumption 3a and then the assumption 3b which is more conservative:

3a. When the voter fails (with probability qv), (i) if a majority exist the voter does not not recognise it (ii) if a majority does not exists, the voter selects a wrong result. 
3b. If the voter fails (with probability qv), independently from the existence of a majority, it selects a wrong result.

We have varied qv in the range 1E-12÷1E-6, to provide sensitivity analysis to the error probability of the voter. Figure 16 shows both reliability (curves F) and safety (curves FC), for assumption 3a reporting also the limit value qv = 0 which corresponds to the correct voter behaviour (assumption 3).

It can be observed that for increasing values of qv (qv ≥ 1E-10) reliability decreases, and this behaviour is more apparent for lower values of qsw. This means that the voter is critical for the system, it may be a bottleneck thus it needs to be simple requiring a very careful validation. If the validation of the voter allows to reach figures of qv = 1E-10, then using assumption 3 (in place of assumption 3a) does not represent an obstacle for a correct modelling of the system. The variations of safety (curves FC) are instead very minimal for all values of qv (they cannot be perceived in our plots), thus we can conclude that safety is insensitive to variations of qv.
and hardware element interaction giving the proper importance also to the secondary aspects (databases and keyboard acceptance test). These aspects are more specific than software and hardware but are critical for the system behaviour as well. Moreover, for our purposes, a more accurate detail, to be accomplished using an additional level, has been considered unnecessary, but in principle the methodology could be applied recursively in order to detail specific aspects. The modular decomposition of the methodology allowed us to keep under control the size of the models within the different levels which can thus be easily managed by the automatic tools. One of the most significant merits of such a modelling approach is that making changes is very easy and cheap. Many times only few objects or levels need to be modified in order to model alternative system architectures (i.e. to investigate which is the most suitable fault-tolerant structure within the design phase) or to analyse the validity of assumptions.

For some of the modules, alternative models have been built using different techniques and tools allowing to exploit some of the benefits usually provided by the ‘diversity’ concept. Despite the fact that it was the same team that built the models, we found that describing the behaviour of the same object by means of many methodologies and tools and comparing their results helped in better understanding the system, has been useful to validate the models themselves and allowed us to discover a few errors made during the modelling activity. Obviously, as the ‘design diversity’ cannot avoid consequences due to correlated errors, making use of different techniques is not a guarantee for the absence of modelling errors, but it can be considered a good way to point out to the ambiguity of the specifications. Therefore we consider this possibility as a valuable one for validation of the models to be investigated more deeply and which must be included, as an important feature, in the methodology. Besides sensitivity analyses to variations of parameters on a model including some assumptions, other models in which assumptions were released or reflecting different assumptions have been built (rather easily, as already pointed out). Comparisons between the results obtained allowed us to evaluate the impact of the assumptions and to understand which were acceptable and those that had to be relaxed. The last evaluation described exemplifies also that the choice of assumption is very critical and must be performed very carefully showing how assumptions that appears to be very similar have completely different impact on the dependability attributes.

With respect to the evaluated characteristics of the Ansaldo ACC CIS, our analyses allowed us to establish the dependability bottlenecks of the current system and to state targets for the several subcomponents such that the system targets could be reached. We could thus provide hints for next releases or modifications of the systems and information to assign targets, and consequently the budget, to the various components of the system.

REFERENCES

7 CENELEC Railway Applications: Software for Railway Control and Protection Systems – prEN50128 (1997)
18 LAAS-CNRS SURF-2 User guide (1994)