A Modular Approach for Model-based Dependability Evaluation of a Class of Systems

Stefano Porcarelli[1], Felicita Di Giandomenico[1],
Paolo Lollini[2], and Andrea Bondavalli[2]

[1] Italian National Research Council, ISTI Dept., via Moruzzi 1, I-56124, Italy
{porcarelli, digiandomenico}@isti.cnr.it
[2] University of Firenze, Dip. Sistemi e Informatica, via Lombroso 67/A, I-50134, Italy
{lollini, a.bondavalli}@dsi.unifi.it

Abstract. Modeling for dependability and performance evaluation has proven to be a useful and versatile approach in all the phases of the system life cycle. Indeed, a widely used approach in performability modeling is to describe the system by state space models (like Markov models). However, for large systems, the state space of the system model may result extremely large, hardening its solution. Taking advantage of the characteristics of a particular class of systems, this paper develops a methodology to construct an efficient, scalable and easily maintainable architectural model for such class, especially tailored to dependability analysis. Although limited in its applicability, the proposed methodology shows very attractive because of its ability to master complexity, both in the model design phase and, then, in its solution. A case study is also included, selected from the addressed class of systems.

1 Introduction

Analytical and simulative modeling for dependability and performance evaluation has proven to be a useful and versatile approach in all the phases of system life cycle. During design phase, models give an early validation of the concepts and architectural choices, allow comparing different solutions to highlight problems within the design and to select the most suitable one. During the operational life of the systems, models allow to detect dependability and performance bottlenecks and to suggest solutions to be adopted for future releases.

The model-based evaluation method has gained wide applicability and usage since a few decades. However, to keep pace with dominant characteristics of modern applications, namely complexity, modeling methodologies need to evolve towards more and more efficient solutions. 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 their relevant aspects is not trivial at all: the information explosion problem remains the major difficulty for practical applications. To successfully deal with large and complex models, the "divide and conquer" approach in which the solution of the entire model is constructed on the basis of the solutions of its individual sub-models [2] is followed. However,

to properly cope with state explosion, it is not sufficient to resort to a modular and compositional approach at model design level, but this same approach has to be pursued at model solution level as well. Since this is clearly a pressing requirement, a number of research studies have started to tackle this problem in the last years, and some interesting approaches have appeared in the literature. [1] describes an approach for solving reliability models for systems with repairable components and introduces several approximations (effective in practice). [3] describes a general-purpose technique useful for an efficient analytic solution of a particular class of nonproduct-form models. Some other authors ([6], [10]) describe techniques based on generating the model of a modular system by composition of the submodels of its components. [7] proposes an efficient modeling approach and solution for Phase Mission Systems. As expected, given the high difficulty of the problem when approached in its most general terms, the existing proposals do not attempt general solution for the entire spectrum of systems; rather, they address specific system categories. Pursuing similar objectives, the goal of this paper is to present a novel methodology for dependability and performability evaluation tailored to a particular class of systems.

Specifically, we consider systems running applications composed of a set of functional tasks organized in such a way that the flow of computation moves along a hierarchy, without any dependency among components at the same hierarchical level, as better described in the next sections. Typical applications resembling such structure are control systems or resource management systems, where activities are carried on cyclically and consist of the following sequential tasks: monitoring of the system/subsystem behavior, elaboration of some computation to determine next actions and actuation of the selected actions. Taking advantage of the peculiar characteristics of such systems category, our modeling approach provides an efficient, scalable and easily maintainable architectural model that allows to better master complexity both in the design of the model and in its solution. To illustrate the methodology, its application to a case study in the field of resource management systems is shown.

The rest of this paper is organized as follows. Section 2 describes the particular class of systems considered. Section 3 is devoted to the full description of the modeling approach. In Section 4 a case study is lightly presented to illustrate a practical application of the methodology. Conclusions are drawn in Section 5.

2 Characteristics of the considered class of systems

The class of systems our methodology focuses on is mainly characterized by a hierarchical structure of the system components and of the computation flow, as graphically depicted in Figure 1.

In more details, the main properties characterizing this class of systems are:

1. The system is composed of sets of hardware or software components (*COMP*) which can be logically grouped in "stages" (Stage 1, ..., k, $k+1$, $k+2$, ...);
2. A component at stage k (COMP^k) may interact only with those at stages $(k-1)$ and $(k+1)$ by means of message exchange and these interactions are

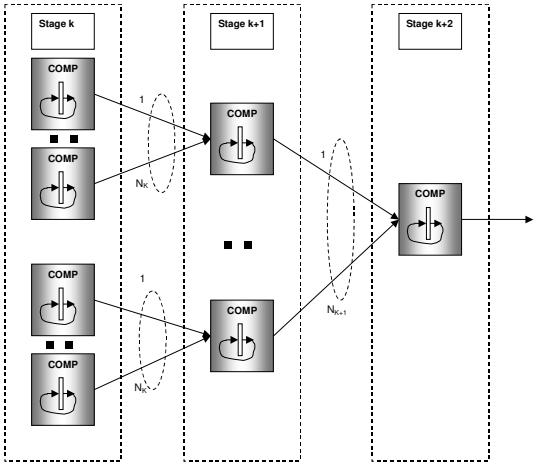

Fig. 1. Targeted class of systems

unidirectional (e.g. from stage k to stage $(k+1)$). Therefore, there could be a functional dependency between COMP ^{k} and COMP $^{k+1}$;

3. The interaction among components and the failure assumptions on each component are highlighted in Figure 2. This scheme is very general and must be specialized for the particular component under analysis. To explain the generic component's behavior, let's suppose it receives an input following a Poisson distribution with a rate λ^{IN}. These inputs are assumed to be correct or incorrect with a probability α and $1-\alpha$, respectively.
   In correspondence of inputs, which arrive with a rate λ^{IN}, the component produces an output with a rate $p * \lambda^{IN}$, where p is the probability a received input leads the component to produce an output. Moreover, the component is assumed to possibly behave incorrectly by self-generating spurious outputs with a rate λ^{S}. Thus, the "potential"[3] output rate of the component is expressed as $\lambda^{IN \rightarrow OUT} = \lambda^{IN} + \lambda^{S}$. From the point of view of propagation, an output issued by COMP is propagated to another component with a rate $\lambda^{OUT} = (p_{Correct} + p_{Corrupted}) * \lambda^{IN \rightarrow OUT}$, where $p_{Correct}$ and $p_{Corrupted}$ represent the probabilities of generating a correct output (correct emission) and an incorrect output (incorrect emission), respectively. In general, a *correct emission* happens whenever a correct output is produced. Obviously, a correct emission is possible only in response to a correct input and the system is free from errors. An *incorrect emission* happens either in reply to an incorrect input, or as consequence of a spurious output or of a wrong processing of a correct input. A *correct omission* may happen as consequence of an incorrect input or of an erroneous status of the system. An *incorrect omission* may happen as consequence of wrong processing of a correct input. These

[3] Here, a "potential" output encompasses both emitted and omitted output ($p = 1$), while for "output" we refer only to those emitted.

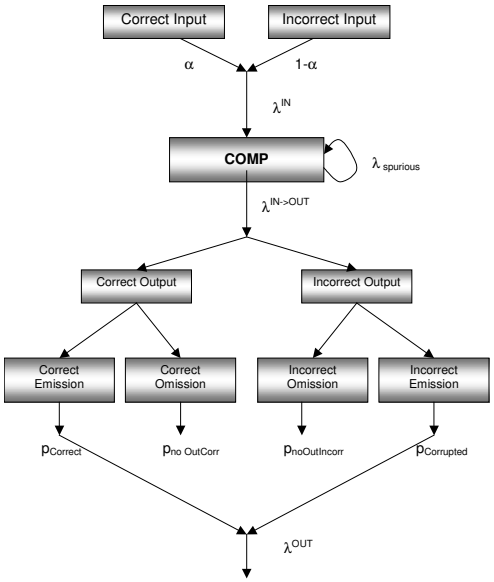

Fig. 2. How a Generic Component Interacts with Others

input-output combinations are summarized in Table 1. The input/output parameters characterizing each component are instead summarized in Table 2, where $p_{noOutCorr}$ and $p_{noOutIncorr}$ are the probabilities that the output is correctly omitted and incorrectly omitted, respectively.

Input	**Corresponding feasible output**
Spurious output (internally generated)	Incorrect Emission
Correct input	Correct Emission, Incorrect Emission, Incorrect Omission
Incorrect input	Correct Omission, Incorrect Emission

Table 1. Input-output combinations

As already discussed in the Introduction, although the presented system characterization does not cover each conceivable system, yet it is well suited for a restricted but popular class of systems, like control systems and resource management systems. Given the behavior structure and failure semantic depicted in Figure 2, typical measures of interest from the dependability point of view in this context are:

1. The probability of correct and incorrect emission;
2. The probability of correct and incorrect omission;
3. The overall probability that the system does not undertake wrong actions.

Input parameters	Output parameters
α, λ^{IN}	λ^{OUT}, $p_{Correct}$, $p_{noOutCorr}$, $p_{noOutIncorr}$, $p_{Corrupted}$

Table 2. Input-output parameters for a component model

3 The Proposed Modeling Approach

In this section we describe our modeling approach, which fully exploits the above discussed characteristics of the reference class of systems. First, the general overview is presented, related concepts are introduced and some motivations for the work are provided. Then, the details of the methodology in terms of model assumptions and model solution are given.

3.1 General overview

In order to construct an efficient, scalable and easily maintainable architectural model, we introduce a stepwise modeling refinement approach, both for the model design process and for the model solution. Another advantage of this approach is to allow models refinement as soon as system implementation details are known or/and need to be added or investigated. The philosophy of our modeling approach is shown in Figure 3. The model design process adopts a top-down approach, moving from the entire system description to the definition of the detailed sub-models, while the model solution process follows a bottom-up approach. As inspired by [5], the system is firstly analyzed from a functional point of view (functional analysis), in order to identify its critical system functions with respect to the validation objectives. Each of these functions corresponds to a critical service provided by a component.

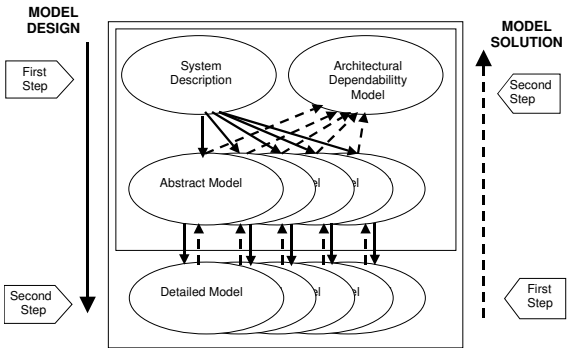

Fig. 3. Modeling approach

The overall system is then decomposed in subcomponents, each one performing a critical subfunction, and each subfunction is implemented using a

model that describes its behavior. Therefore, starting from the high-level abstract model, we perform a decomposition in more elementary (but more detailed) sub–models, until the required level of detail is obtained.

The definition of the functional (*abstract*) model represents the first step of our modeling approach. The rules and the interfaces for merging them in the architectural dependability model are also identified in this phase. The second step consists in detailing each service in terms of its software and hardware components in a detailed (*structural*) model accounting for their behavior (with respect to the occurrence of faults). The fundamental property of a functional model is to take into account all the relationships among services: a service can depend directly from the state of another service or, indirectly, on the output generated from another service. The detailed model defines the structural dependencies (when existing) among the internal sub–components: the state of a sub–component can depend from the state (failed or healthy) of another sub–component.

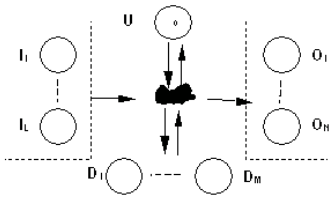

Fig. 4. Functional-level model related to a single service

Figure 4 shows the functional-level model related to a single service. The internal state S is here composed of the place U, representing the nominal state, and of the places $D_1 \ldots D_M$, representing different possible erroneous (degraded) states. The places $I_1 \ldots I_L$ and $O_1 \ldots O_N$ represent, respectively, the input (correct or exceptional, due to propagation of failures from interacting modules) and the output of the model (correct behavior or failure - distinguishing several failure modes). The state changes (from the nominal, correct state to the erroneous states and viceversa) and the flow between the input and output places are regulated by a structural model of the service implementation, indicated in Figure 4 as a black cloud.

An example of a structural model using SAN [11] is shown in Figure 5, where the service F is obtained through components C_1, C_2 and C_3. In turn, each component has the same structure of Figure 4, where the input/output relationships with other components are not considered. Notice that how these components concur to determine the state of the service (F, $/F_1$ or $/F_2$) is described in a simple way by means of the input gates *repairGate* and *failureGate* and the output gate *outGate* (in this example they are not fully specified because they are system dependent). C_2 and C_3 are in parallel and in series with C_1: F is in its nominal state if C_1 is up and at least one of C_2 and C_3 is up. The

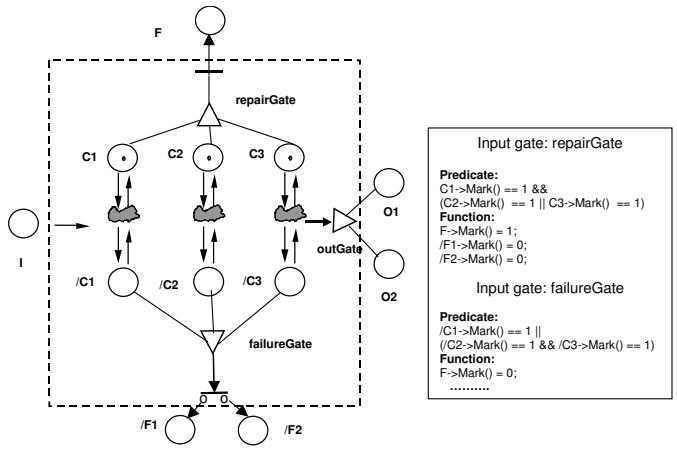

Fig. 5. An example of Detailed Model

output gate *gateOut* defines the relationship between input and output, given the internal status of the system.

So far, we have described how to model a complex system starting from its functional specification and applying a stepwise refinement to decompose it in small sub–models. Now, the second part of the methodology is presented, which concerns the modular model solution, carried out in a bottom-up fashion. In fact, a methodological approach becomes attractive when it is not only directed to build models in a compositional way, but it also includes some capabilities to reduce their solution complexity. According to Figure 3, the overall architectural model is achieved combining together only the abstract models. The parameters to be used for this model are obtained solving the corresponding lower level ones.

3.2 Environment characteristics

Suppose that the k-th stage of the system (cf. Figure 1) consists of a number of groups, all identical, of N_k components (COMP_1^k, ..., $\mathrm{COMP}_{N_k}^k$). We denote with $\lambda_i^{OUT,\ COMP^k}$ the intensity of the output process of the i-th component of a group belonging to stage k (COMP_i^k). We make the following assumptions:

1. The distribution of the input process of each component is Poisson with rate λ^{IN}. This is accepted in the literature when the number of arrivals in a given time interval of time are independent of past arrivals.
2. The distribution of the output process of each component is Poisson distributed with a rate λ^{OUT}. This assumption corresponds, for example, to the case in which the inputs are processed sequentially without queuing and losses, and the processing time of the input is deterministic. Equivalently, we could obtain the same output distribution considering that the service time is Poisson distributed and that the component operates as a steady-state M/M/1 queuing network [12].

Using the assumption that the output process of COMP_i^k is Poisson distributed with rate $\lambda_i^{OUT,\ COMP^k}$, the superposition of N_k Poisson processes with intensities $\lambda_1^{OUT,\ COMP^k}, \ldots, \lambda_{N_k}^{OUT,\ COMP^k}$ is equivalent to a Poisson process with intensity equal to $\lambda_1^{OUT,\ COMP^k} + \ldots + \lambda_{N_k}^{OUT,\ COMP^k}$.

Solving the detailed model of components COMP_1^k, ..., $\text{COMP}_{N_k}^k$ leads to the evaluation of the probabilities of correct/incorrect output emission/omission and the intensity of the output process of a group of N_k components. Let's defining as $P_{Correct}^{k_i}$, and $P_{Corrupted}^{k_i}$ the probability of correct emission, and the probability of incorrect emission of COMP_i^k, respectively. Notice that these probabilities depend upon the intensity of the input process ($\lambda_i^{IN,\ COMP^k}$) and of spurious alarms ($\lambda_i^{S,\ COMP^k}$) (both supposed being Poisson). The following relations holds:

$$\Lambda^{OUT,\ COMP^k} = \sum_{i=1}^{N_k} \lambda_i^{OUT,\ COMP^k} \quad , \tag{1}$$

$$\alpha_{COMP^{k+1}} = \frac{1}{\Lambda^{OUT,\ COMP^k}} \sum_{i=1}^{N_k} \lambda_i^{OUT,\ COMP^k} \frac{P_{Correct}^{k_i}}{(P_{Correct}^{k_i} + P_{Corrupted}^{k_i})} \quad , \tag{2}$$

where $\Lambda^{OUT,\ COMP^k}$ is the intensity of the process achieved by aggregating the output processes of the components COMP_1^k, ..., $\text{COMP}_{N_k}^k$, while $\alpha_{COMP^{k+1}}$ is the probability that the next component at stage $k+1$ receives a correct input. Analogous considerations hold for COMP^{k+1}, and so on. This general approach can be specified for the following cases:

- If all groups of N_k components at stage k are identical, the total number of detailed models to be solved in order to evaluate the system of Figure 1 is equal to $\sum_{k=0}^{K} N_k$, where K is the number of "stages" in the system. It corresponds to evaluate the system of Figure 1 with its equivalent model of Figure 6 where $\Lambda^{OUT,\ COMP^k}$ and $\alpha_{COMP^{k+1}}$ are evaluated by means of equations (1) and (2), respectively. This way, the "tree" structure of the system of Figure 1 collapses in a unique "branch" from the point of view of system evaluation.
- If all groups of N_k components can not be considered identical at each stage, the number of models to be solved depends on the number of different "branches" in which the overall model can be simplified (see Figure 1).
- If for each stage k of the system (cf. Figure 1), all the N_k components COMP_1^k, ..., $\text{COMP}_{N_k}^k$ are identical, it is possible to solve only K detailed models, one for each stage. Therefore, if all the components at level k are identical, than $\lambda_i^{OUT,\ COMP^k} = \lambda^{OUT,\ COMP^k}$, $P_{Correct}^{k_i} = P_{Correct}^k$, $P_{Corrupted}^{k_i} = P_{Corrupted}^k$, and the previous equations reduce to

$$\Lambda^{OUT,\ COMP^k} = N_k * \lambda^{OUT,\ COMP^k} \quad , \tag{3}$$

$$\alpha_{COMP^{k+1}} = \frac{P_{Correct}^k}{(P_{Correct}^k + P_{Corrupted}^k)} \quad . \tag{4}$$

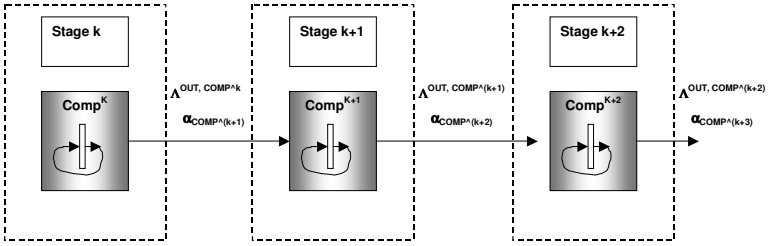

Fig. 6. Simplified system model equivalents to Figure 1

Equivalently, the general model of Figure 1 is reduced to the equivalent simplified system model of Figure 6 that can be solved more easily.

If it can not be assumed that the output process of COMP^{k_i} follows a Poisson distribution, the general approach is still valid provided that the detailed model is slightly modified allowing to estimate the real distribution of such a process. The same distribution will be used as input at the $k+1$ stage. However, in general, it will be no longer possible to solve the models analytically.

If the measures of interest are probabilities, the moments of the distribution of the events which yield such probabilities are not considered at all. In this case it is not necessary to use, at the abstract level, models having the same distribution estimated at the detailed ones. If, on the contrary, we are interested in evaluating the moments of the distribution of correct/incorrect output emission/omission, the output processes distributions achieved by the detailed models have to be used for the solution of the abstract models.

3.3 Model Solution Scheme

According to Figure 3 (showing the philosophy of our modeling approach) the model solution follows a bottom-up approach: solution of a detailed model is exploited to set up the parameters of the corresponding abstract model and of the detailed model of the next (contiguous) components (the output of the detailed COMP^k model acts as input for the detailed COMP^{k+1} model). To keep the presentation simple, the model solution scheme is described in the case where all the N_k components at each stage k are identical; therefore only K detailed models (one for each stage) have to be solved. Figure 7 shows the relationships among a detailed model of COMP^k and the model COMP^{k+1}.

With reference to the measures of interest listed in Section 2, the outcomes of the detailed model COMP^k are:

1. $p^k_{noOutCor}$: is the probability that no output is produced by component COMP^k, as a consequence of an incorrect input;
2. $p^k_{noOutIncorr}$: is the probability that an expected output is incorrectly not propagated by component COMP^k, as consequence of an internal fault;

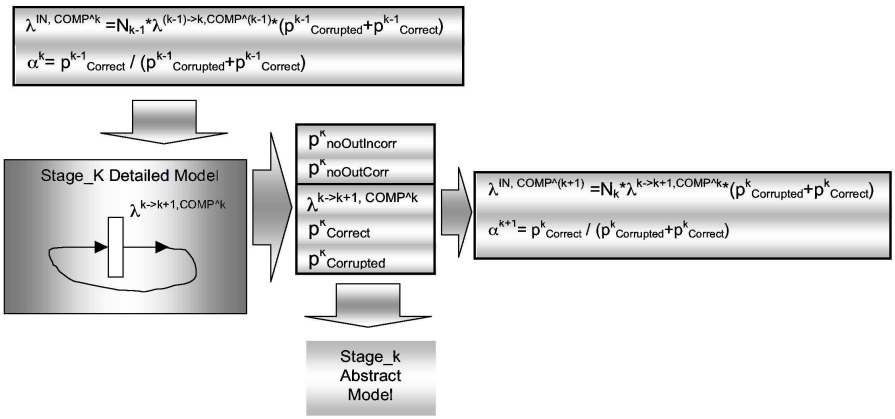

Fig. 7. Relationships between models solutions

3. $\lambda^{IN \rightarrow OUT,\ COMP^k} * (p^k_{Corrupted} + p^k_{Correct})$: is the rate of messages propagated by component COMP^k to component COMP^{k+1};
4. $p^k_{Correct}$: is the correct emission probability;
5. $p^k_{Corrupted}$: is the emission failure probability. This value encompasses both an expected wrong emission (as consequence of wrong internal processing) and the unexpected emission (as consequence of an internal self-generated false alarm).

All these parameters are used in the abstract model of component COMP^k (see Figure 7) while $\lambda^{IN \rightarrow OUT,\ COMP^k}$, $p^k_{Correct}$ and $p^k_{Corrupted}$ are used to derive the parameter $\lambda^{IN,\ COMP^{k+1}}$ to be used in the detailed model of COMP^{k+1}. In the system framework COMP^k and COMP^{k+1} represent two components directly connected that exchange messages in one direction (from COMP^k to COMP^{k+1}).

Summarizing, the overall solution scheme is shown in Figure 8. The detailed models are solved separately: firstly, it is solved the model of COMP^k, then the values provided by equations (3) and (4) are passed as input to the detailed model of COMP^{k+1} and so on. Finally, the probabilities of correct/incorrect output emission/omission are passed to the corresponding abstract models, they are joined together and then the overall abstract model is solved.

The advantages of the proposed approach are in two directions: first, to cope with the problem of state space explosion when modeling a complex system and, second, to allow efficient model solution for those systems having most of their components identical and interacting each others only by means of message exchange. Actually, in case the components are not all equal, a larger number of detailed models have to be solved but still separately. Thus, the overall model, encompassing all the useful information with respect to the measures of interest, is achieved by joining the abstract models.

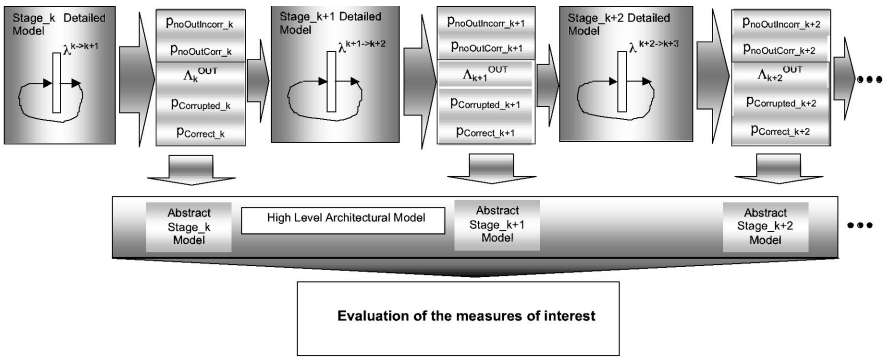

Fig. 8. Overall Solution Scheme

4 An application: the CAUTION++ system architecture

The main objective of CAUTION++ [9] is the smooth transition from existing wireless systems to new generation ones. This is pursued by designing and developing a novel, low cost, flexible, highly efficient and scalable system able to be utilized by mobile operators to increase the performance of all network segments (namely, GSM, GPRS, UMTS and WLAN). Different segments can use different technologies and radio access. CAUTION++ exploits the available system resources by enabling real-time monitoring, alarming, immediate adaptive application of RRM (Radio Resource Management) techniques, vertical handover to other systems (possibly of other operators) having as a major goal to optimize the operators' revenue and the users' satisfaction.

The architectural solution is based on the concept of "monitor and manage". All resources at the air-interface are monitored in real-time and proper system components are developed to handle generated alarms through a set of RRM (Radio Resource Management) techniques, to be applied where needed. The decisions-making process is performed at two levels: a local resource management in charge of managing the capacity of a each single network and a global resource management, which is in charge of inter-network coordination for the sake of the overall optimization of network capacity.

Figure 9 shows the main components of the CAUTION++ architecture. Each network segment has its own ITMU (Interface Traffic Monitoring Unit) and RMU (Resource Management unit) which allow to monitor and manage the attached network, respectively. Within each operator network, a GMU (Global Management unit) can perform a global optimization. Different GMUs cooperate to optimize among different operators. A Location Server (LS) can be used to track users' mobility and location: such information can be exploited by GMU for a global optimization.

At the level of resource management provisions as offered by CAUTION++, the starting point in addressing dependability issues is that the system under development builds upon a number of existing network segments, each one charac-

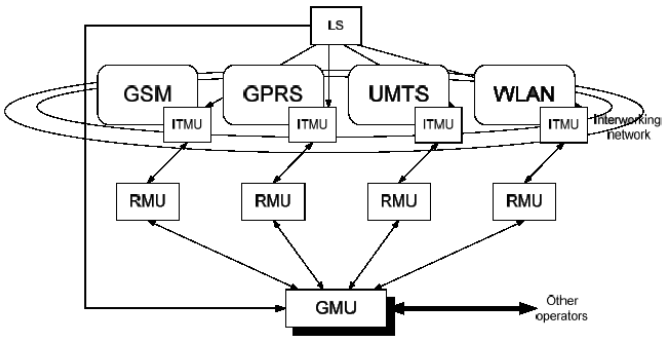

Fig. 9. Network Architecture for provision of capacity management mechanisms

terized by specific dependability and performance properties, in accordance with the specific configuration adopted by the involved operators. The basic network segments are therefore to be regarded as "non-touchable" system components. Their dependability and performance aspects can obviously be analyzed ([4]). Therefore, the dependability requirements in such context pertain the components of the CAUTION++ architecture and the infrastructure connecting them.

The most important and challenging dependability requirement on the CAUTION++ architecture is to prevent RMU and/or GMU subsystems from carrying out a reconfiguration action wrongly or when is not necessary (as consequence of some fault). Therefore, we are interested in the probability of correct/incorrect emission and omission at RMU and GMU level.

4.1 Overall Solution Scheme for the CAUTION++ architecture

The solution scheme for CAUTION++ is presented in Figure 10. It consists of a three "stage" system in which each "stage" is composed of one component (ITMU, RMU and GMU detailed model). This schema could be also valid in a more complex scenario where more than one ITMU, RMU and GMU are present, provided that all components at the same level are identical. For a more detailed description of this case study, refer to [8].

5 Conclusions

This paper has focused on the development of a general evaluation methodology to master system complexity and favor model reusability and refinement as much as possible. The proposed method applies to a limited, but significant class of systems characterized by a hierarchical computation flow (typical representatives are control systems). In fact, the methodology exploits such system property to set up a modular and compositional approach, both for the model design process and for the solution process. The resulting efficiency and easiness of the overall

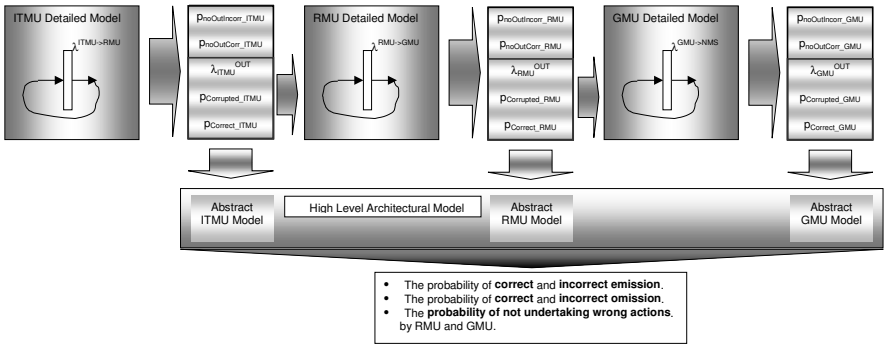

Fig. 10. CAUTION++ Overall Solution Scheme

evaluation activity makes this method, when applicable, very attractive. The CAUTION++ system architecture has been also briefly introduced, to show a practical case study for the described methodology.

The solution scheme has been presented in the most simple and efficient case where all components at the same level of the hierarchy are equal. Considering a more general case would imply a higher number of models to be individually set up and solved, but still retaining the same ability of modularity and compositionality. As future work, we intend to refine our approach, by providing a completely general description to fully address the target systems class.

Acknowledgments This work has been partially supported by the European Community through the IST-2001-38229 CAUTION++ project and by the Italian Ministry for University, Science and Technology Research (MURST), project "Strumenti, Ambienti e Applicazioni Innovative per la Societa dell'Informazione, Sottoprogetto 4".

References

1. M. Balakrishnan and K.S. Trivedi. Componentwise decomposition for an efficient reliability computation of systems with repairable components. In *Int. IEEE Symp. Fault-Tolerant Computing (FTCS-25)*, pages 259–268, 1995.
2. G. Balbo. Introduction to stochastic Petri nets. In J.-P. Katoen, H. Brinksma, and H. Hermanns, editors, *Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science Berg en Dal, The Netherlands, July 3-7, 2000, Revised Lectures*, volume 2090 of *Lecture Notes in Computer Science*, pages 84–155. Springer-Verlag, 2001.
3. G. Balbo, S.C. Bruell, and S. Ghanta. Combining queuing networks and generalized stochastic Petri nets for the solution of complex models of system behavior. *IEEE Trans. Computers*, 37(10):1251–1268, 1988.
4. S. Porcarelli, F. Di Giandomenico, A. Bondavalli, M. Barbera, and I. Mura. Accurate Availability Estimation of GPRS. *IEEE Transactions on Mobile Computing*, 2(3):233–247, 2003.

5. C. Betous-Almeida, and K. Kanoun. Stepwise construction and refinement of dependability models. In *Proc. IEEE International Conference on Dependable Systems and Networks DSN 2082*, Washington D.C., 2002.
6. J.F. Meyer and W.H. Sanders. Specification and construction of performability models. In *Workshop on Performability Modeling of Computer and Comm. Systems*, pages 1–32, 1993.
7. I. Mura and A. Bondavalli. Markov Regenerative Stochastic Petri Nets to Model and Evaluate Phased Mission Systems Dependability. *IEEE Computer Society Press*, 50, 2001.
8. S. Porcarelli, F. Di Giandomenico, A. Bondavalli, and P Lollini. Model-based evaluation of a radio resource management system for wireless networks. In *Computing Frontiers (to appear)*, Ischia, Italy., April 2004.
9. CAUTION++ IST Project. Capacity Utilization in Cellular Networks of Present and Future Generation++. http://www.telecom.ece.ntua.gr/CautionPlus/.
10. I. Rojas. Compositional construction of SWN models. *The Computer Journal*, 38(7):612–621, 1995.
11. W. H. Sanders, and J. F. Meyer. A Unified Approach for Specifying Measures of Performance, Dependability and Performability. In *Dependable Computing for Critical Applications*, volume 4 of *Dependable Computing and Fault-Tolerant Systems*, pages 215–237. Springer Verlag, 1991.
12. K. S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley and Sons, New York, 2001.