Dependability Modeling and Analysis in Dynamic Systems

Salvatore Distefano and Antonio Puliafito
University of Messina, Engineering Faculty,
Contrada di Dio, S. Agata, 98166 Messina, Italy.
Tel: +39 090 3977318, fax: +39 090 3977471,
Email: salvatdi[apulia]@ingegneria.unime.it

Abstract

Dependability evaluation is an important, often indispensable, step in (critical) systems design and analysis processes. The introduction of control and/or computing systems to automate processes increases the overall system complexity and therefore has an impact in terms of dependability. When a system grows, dynamic effects, not present or manifested before, could arise or become significant in terms of reliability/availability: the system could be affected by common cause failures, the system components could interfere, effects due to load sharing arise and therefore should be considered. Moreover it is of interest to evaluate redundancy and maintenance policies. In those cases it is not possible to recur to notations as reliability block diagrams (RBD), fault trees (FT) or reliability graphs (RG) to represent the system, since the statistical independence assumption is not satisfied. Also more enhanced formalisms as dynamic FT (DFT) could result not adequate to the goal. To overcome those problems we developed a new formalism derived from RBD: the dynamic RBD (DRBD). In this paper we explain how to use the DRBD notation in system modeling and analysis, coming inside a methodology that, starting from the system structure, drives to the overall system availability evaluation following modeling and analysis phases. To do this we use an example drawn from literature consisting of a multiprocessor distributed computing system, also comparing our approach with the DFT one.

1 Introduction

A system is a collection of components, subsystems and/or assemblies arranged according to a specific design in order to achieve acceptable performance and reliability levels. The types of components, their quantities, their qualities and the manner in which they are arranged within the system have a direct effect on the system's reliability. The main goal of system's reliability study is the construction of a model (life distribution) that represents the times-to-failure of the entire system based on the life distributions of the components, subassemblies and/or assemblies (*"black boxes"*) from which it is composed [6].

Several are the approaches to represent and analyze system reliability. The choice is among *analytic* (Markov models and derived, Petri nets, stochastic reward nets, Bayesian networks, ...) or *simulation* techniques (discrete event, Monte Carlo, ...). All of them are powerful reliability analysis methods, but, on the other hand, they are not much "user friendly": often it is really hard to obtain a model directly from the specifications, especially in case of complex systems. This fact motivated the definition of specific reliability/availability modeling formalisms as *reliability block diagrams* (RBD) [20], *fault trees* (FT) [26] and *reliability graphs* (RG) [21]. Although RBD, RG and FT provide a view of the system close to the modeler, more readable and understandable than any other formalism, they are defined on an (heavy) assumption: the components must be *statistically independent*. They do not provide any elements or capabilities to model reliability interactions among components or subsystems, or to represent system reliability configuration changing, aspects conventionally identified as *dynamic*. Common examples of such relationships are: *load-sharing*, *standby redundancy*, *interferences*, *dependencies*, *probabilistic and common mode failures*. Moreover a failed component/subsystem could be repaired (maintenance, *reliability growth model*), the system could be *phased mission*, ... In particular these remarks concern large and complex parallel, distributed and network-centric computing systems, where different subsystems, devices or components could interacts each other. For example load sharing phenomena could affect the network availability; dependability resources-optimization requirements could be translated into maintenance and/or redundancy schemes and policies; some interference or inter-dependence between devices could arise (wireless devices, sensors, ...); common

1-4244-0910-1/07/$20.00

cause failures could group electric devices, in case of power jumps or sudden changes of temperature (melting point of transistors, ...).

These arguments awakened the scientific community to the need of new formalisms as the *dynamic fault trees* (DFT) [4, 12]. DFT extend static FT to enable modeling of time dependent failures, introducing new dynamic gates and elements. DFT are a good attempt in dynamic reliability modeling, but they cannot adequately represent several of the dynamic behaviours previously listed. More specifically, using DFT it is hard (and in some cases not possible) to compose dependencies reflecting characteristics of complex and/or hierarchical systems, to define customizable redundancy schema or policy, to represent load sharing and probabilistic-common failure mode phenomena, and to adequately model reparability features. To overcome these problems or lacks in reliability/availability modeling, in [10, 8, 9, 7] we have defined a new notation named *dynamic reliability block diagrams* (DRBD), by extending the RBD formalism.

In this paper we explain how to use the DRBD notation in system modeling and analysis. To do this we propose a methodology that, starting from the system structure, drives to the overall system availability analysis by evaluating the corresponding DRBD model. We try to specify the underlined methodology step by step, by parallely describing an example/case study of a multiprocessor distributed computing system as complement for the explanation, in order to clarify this latter. A comparison with the DFT approach is also provided showing how it is possible to map from the existing model types to the new model type. By this way, the capabilities, the flexibility, and the other features of the DRBD formalism in modeling and analysis of system reliability/availability could be adequately pointed out. The reminder of the paper is organized as follows: section 2 presents the multiprocessor computing system taken as example to explain our methodology. In section 3 some background concepts of the utilized notations are provided. Section 4 describes how to apply the DRBD modeling approach to the motivating example, then, the analysis of the DRBD model thus obtained is reported in section 5, comparing our approach to the DFT one through the results. Lastly, section 6 provides some final considerations on the DRBD methodology.

2 Motivating Example: a Multiprocessor Distributed Computing System

In order to better describe our proposed methodology, we decided to adopt a step by step approach focused around the reliability analysis of a multiprocessor systems, taken as a case study.

It refers to a multiprocessor computing system drawn from literature ([16, 18]) from which we summarize the following description, accompanied by the scheme reported in Fig. 1. It is composed by two computing module: CM_1 and CM_2. Each of them contains one processor (P_1 and P_2 respectively), one memory (M_1 and M_2) and two hard disks: a primary (D_{11} and D_{21}) and a backup disk (D_{12} and D_{22}). Initially, the primary disk is accessed by the computing modules, while the backup disk contains the copy of the information inside the primary disk, and is accessed only periodically for update operations. If the primary disk fails, it is replaced in its function by the backup disk. In terms of reliability the disks are identical, they are characterized by the same failure rate or reliability cumulative distribution function (cdf). But, when the primary disk is operational, the failure rate of the backup disk decreases as consequence of the management policy that decreases its workload condition.

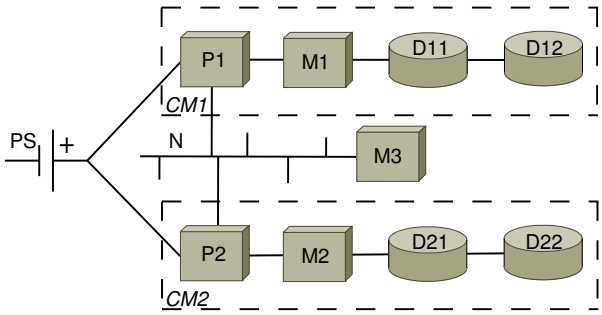


Figure 1. Schematic representation of the Multiprocessor Distributed Computing System

The computing modules are connected by means of the bus N; moreover, P_1 and P_2 are energized by the power supply PS: the failure of PS forces P_1 and P_2 to fail.

M_3 is a spare memory replacing M_1 or M_2 in the case of failure. If M_1 and M_2 are operational, M_3 is just kept alive, but it is not accessed to load/store any data by the processors. When M_1 or M_2 fail, M_3 substitutes the failed unit.

In order to work properly the multiprocessors computing system of Fig. 1 requires that at least one computing module (CM_1 or CM_2), the power supply PS and the bus N are operating correctly. Moreover a computing module (CM_1 and CM_2) is operational if the processor (P_1 and P_2 respectively), one between the local memory (M_1 and M_2) and the shared memory M_3 and one disk (D_{11} or D_{21} for CM_1 and D_{12} or D_{22} for CM_2) are not failed.

The example presented, which comes from [16] and [18], where it is also analyzed, does not represent/implement the optimal system's management configuration available in terms of reliability. To improve the system reliability, a better solution could be to consider the two computing modules CM_1 and CM_2 as redundant subsystems. Another lack of the model is that no load sharing effects are

considered between the computing modules: the fact that both CM_1 and CM_2 are operating could increase the reliability of each computing module, sharing the workload with the other one. If only a computing module works, it must process all the incoming requests, which were previously elaborated in cooperation with the other CM. For this reason, the single CM could be more stressed and therefore its probability to fail could increase. These behaviours cannot be modeled by a DFT because they involve dependencies composition and/or modeling of uncovered aspects, such as the load sharing. Aspects that are instead adequately contemplated in the DRBD approach.

In this paper we describe how to model the multiprocessors computing system of Fig.1, using both DFT and DRBD in order to better explain the DRBD methodology and, at the same time, to compare the two approaches. In other words, the comparison between DRBD and DFT is subordinated and finalized to the DRBD methodology explanation, without specifying any mapping rules between DRBD and DFT, nor penetrating into the advanced DRBD modeling where behaviours not representable by DFT are investigated.

3 System Reliability Modeling Notations

To model and analyze the example described in the previous section it is necessary to briefly introduce the notations used. In subsection 3.1 an overview of FT and DFT is provided. Then, in subsection 3.2, we outline the RBD notation to better introduce the DRBD, which are discussed in subsection 3.3. More details on the DRBD formalism can be found in [10, 8, 9, 7].

3.1 FT and DFT

Fault trees provide a compact, graphical method to analyze system reliability. Static trees [26] use Boolean gates to represent how component failures combine to produce system failure. The static gates are therefore purely combinatorial logic gates applied to the input failure events. Although static fault trees are more commonly used, they are limited in modeling systems that have no sequential relationships among their component failures. Component failure sequences are best captured using dynamic fault trees ([4, 12, 11]) that extend static trees to enable modeling of time dependent failures. Dynamic trees add a temporal notion, since system failures can depend on the order of component failures. They can model dynamic replacement of failed components from pools of spares (CSP, WSP and HSP gates); failures that occur only if others occur in certain orders (PAND gates); dependencies that propagate the failure of one component to others (FDEP gates); and specification of constraints on failure orders that simplify analysis computations (SEQ gates).

3.2 RBD

The main virtue of a RBD is that it is easy to read. In a RBD, the logic diagram is arranged to indicate which combinations of component failures result in the failure of the system, or which combinations of properly working components keep the system operational. A block in RBD represents the working physical component, and the failure of this component is indicated by the removal of the corresponding block. If enough blocks are removed in an RBD to interrupt the connection between the input and output points, the system fails.

Generally three main types of connection (series, parallel and non series-parallel) can be established between two or more components [20]. The blocks in series, parallel and non series-parallel structure can be merged into a new block which reliability can be computed by using the series and parallel structure equations.

3.3 DRBD

RBD ensure interesting features in reliability modeling such as simplicity, versatility and expressive power. Such interesting features are inherited in a DRBD model, which moreover allows taking into account the system dynamics. A system is considered time-variant, if its components' states evolve taking into account the sequence of events occurred. It is possible to define reliability relationships (dependencies) among components by associating such relationships to events. DRBD implement this issue through two key points: characterizing each component by an own system dynamics and specifying the concept of dependency as the building block of dynamic reliability modeling.

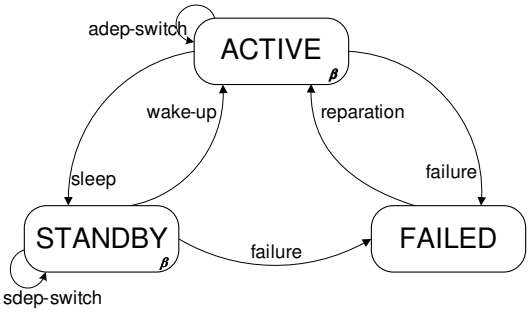
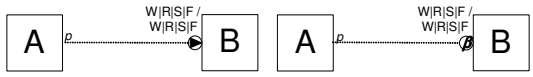


Figure 2. DRBD States-Events Machine

In a DRBD model each component is characterized by a variable *state* identifying its operational condition at a given time. The evolution of a component's state (component's dynamics) is characterized by the *events* occurring to it. The states a generic DRBD component can assume are: *active* if the component works without any problem, *failed* if component is not operational, following up its failure, and *standby* if it is reliable but not available. Active components participate actively to perform the system's task, while standby

components do not contribute to this, they do not interact with the other components. But, at the same time, a component in standby is not failed, it just performs its internal activities.

An event represents the transition from a components state to another one: the *failure* event models a states changes from active or standby to the failed state, the *wake-up* switches from standby to active states, the *sleep* from active to standby states, the *reparation* from failed to active state, the *adep-switch* represents the transitions between two active states and the *sdep-switch* between two standby states. These two latter events are related to the concurrency property of dependencies. Fig. 2 summarizes DRBD states and events. For details see [10, 8, 7].



(a) (b)

Figure 3. DRBD order (a) and strong (b) dependencies representation

The main enhancement introduced by DBRD is the capability to model dependencies among subsystems or components concerning their reliability interactions. A dependency establishes a reliability relationship between two components or subsystems, a *driver* and a *target*. When a specified event, named *action* or *trigger*, occurs to the driver, the dependency condition is applied to the target. This condition is associated to a specific target event, named *reaction*. When a satisfied dependency condition becomes unsatisfied, the target component comes back to the fully active state. The dependency is the tool to model several dynamic reliability aspects of a generic system in the DRBD domain, as those cited in section 1. A dependency could model two kinds of different relationships between a couple of driver-target components: the *order* (Fig. 3 (a)) establishes the sequence order between trigger and reaction events, the *strong* (Fig. 3 (b)) forces the target component to react when a trigger event occurs.

A dependency is also characterized by the action (trigger) and the reaction events. Four types of trigger and reaction events can be identified: wake-up (W), reparation (R), sleep (S) and failure (F). Combining action and reaction, 16 types of dependencies are identified, 32 in considering the relationships. Moreover, since the two relationships model two different behaviors, they can also be composed into more complex dependencies, identifying stronger conditions of dependence [7]. In the examples shown in Fig. 3, A is the driver component and B the target. The dependency action or trigger event is indicated by a letter (W, R, S or F as above), and the reaction by another letter (from the same set), separated from the action by a slash. The total string is placed near the circle. In case of order dependencies an arrow is placed inside the circle (Fig. 3 (a)), while a number characterizes strong dependencies (the of Fig. 3 (b)): it indicates the *dependency rate* β. This latter characterizes strong dependencies with wake-up (W) and/or sleep (S) reaction, weighting, in terms of reliability, the dependence of target component from driver.

The concept of dependency is exploited in DRBD as the basis to represent all the dynamic reliability behaviors. For example, redundancy can be easily represented by exploiting and combining dependencies among units. Other possible applications of dependencies in dynamic system reliability modeling could be load sharing, common cause failure, reparation, and so on. Further details of these interesting capabilities of DRBD are out of the scope of the present work and can be found in [10, 8, 9, 7].

4 The Modeling

In this section we describe the process of modeling the multiprocessor computing system introduced in section 2, by mean of DFT (in subsection 4.1) and DRBD (subsection 4.2) in order to compare the two modeling approaches.

4.1 The DFT Model

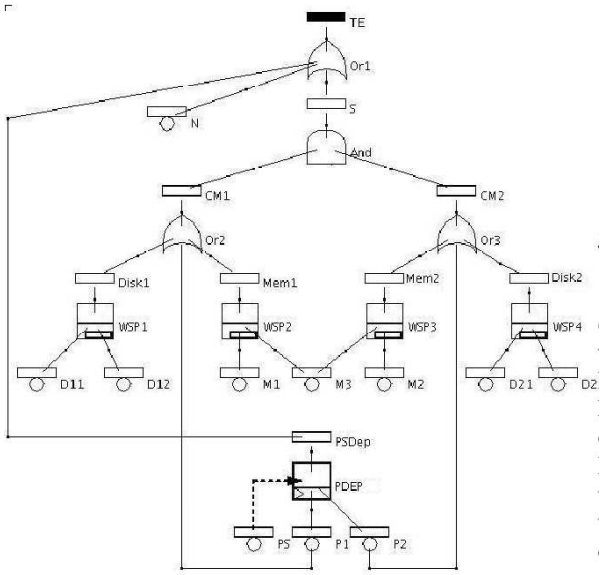


Figure 4. DFT model of the Multiprocessor Distributed Computing System

The DFT modeling the multiprocessor computing system is depicted in Fig. 4, in correspondence to the scheme of Fig. 1 through the names. The DFT model is composed by a FDEP and four WSPs. The FDEP gate models the functional dependency among the power supply (PS) and the two processors P_1 and P_2. WSP1 and WSP4 represent the hard disks management policy: the primary disks D_{11} and D_{21} drive WSP1 and WSP4 gates respectively in the control of the corresponding backup disks D_{12} and D_{22}. In

other words, the backup disks D_{12} and D_{22} are considered as spare units of the primary disks D_{11} and D_{21} respectively. From the probabilistic/analytical point of view, this choice well describes the reliability behaviour of the components, but, from a semantic viewpoint, it does not adequately represent the real condition: the backup disks do not assume a standby configuration because they communicate with the primary disks, making active operations. A more realistic modeling should therefore take into account this fact.

The *partly-loaded standby redundancy policy* applied to the memory units M_1, M_2 and M_3, is represented by the WSP2 and WSP3 gates: if M_1 or M_2 fail, M_3 is activated. The other DFT gates are static: the internal events $DISK_1$ and $DISK_2$ represent the failure of the storage blocks related to the computing modules CM_1 and CM_2 respectively, analogously MEM_1 and MEM_2 represent the computing modules' memory block failure. The failure of the processor (P_1 and P_2) or of the memory block (MEM_1 and MEM_2) or of the disk block ($DISK_1$ and $DISK_2$) drives to the failure of the corresponding computing module (CM_1 and CM_2 internal events). Finally, if both the computing modules fail, or the power supply PS goes down, or the bus N fails, the overall system fault occurs, represented in the DFT as the top event TE.

4.2 The DRBD Model

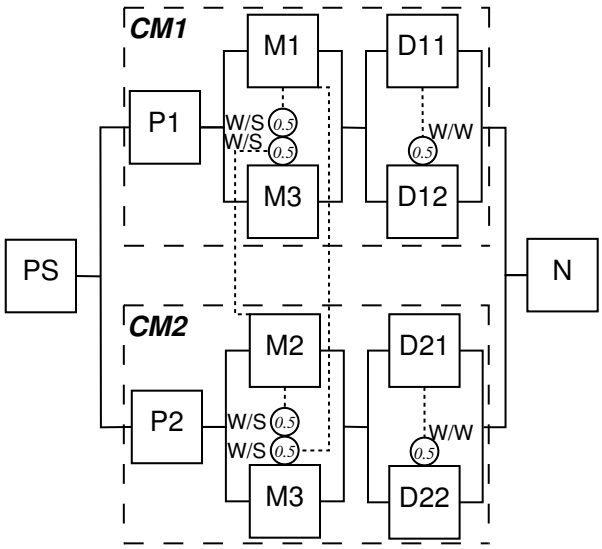


Figure 5. DRBD model of the Multiprocessor Distributed Computing System

The DRBD model reported in Fig. 5 represents the multiprocessors computing system object of this study. As in the DFT case, there is a name correspondence between the DRBD components and the system devices. Since the power supply PS energizes the two processors P_1 and P_2, we interpret this behaviour, and consequently represent it in the DRBD model, as a simple series between each processor and the PS. As in the case of the warm spare disks discussed in the previous subsection, this is just a semantic imperfection of the DFT model, in the fact that the failure of PS generally does not mean the processors failure; this behaviour does not implement a common failure mode condition. Anyway, exploiting the same principle applied for DFT modeling, this could be a good solution to map a FDEP gate into the DRBD domain, in the case that no reparability features must be modeled.

The disks management policy is represented in DRBD by a wake-up/wake-up strong dependency: when the primary disks D_{11} and/or D_{21} are operational, the backup disks D_{12} and/or D_{22} respectively are partial active ($0 < \beta < 1$), maintaining the backup. When a primary disk fails, the corresponding backup disk that substitutes it becomes fully active and fully energized since the dependency condition becomes unsatisfied.

Wake-up/standby strong dependencies are instead exploited to model the redundancy policy managing the memories. These represent the disabling conditions applied by M_1 and M_2, when they are operational, on M_3, keeping this latter in standby. The overall dependency must be applied to M_3 if and only if both M_1 and M_2 are at the same time operational; when one of these fails, M_3 must switch to the fully active state. To realize this condition the two wake-up/standby strong dependencies, from M_1 to M_3 and from M_2 to M_3 are *series composed*: when both are simultaneously satisfied the component M_3 is placed in standby, otherwise M_3 is active. This composed dependency is duplicated in the DRBD model for the sake of clarity, identifying and separating the two computing modules CM_1 and CM_2.

5 The Analysis

The purposes of this section are to describe how a DRBD can be analyzed, also confronting the DRBD approach to the DFT one in order to demonstrate the effectiveness of the former. Firstly, in subsection 5.1, an overview of the possible DRBD solution techniques is introduced, specifying the solution algorithm then applied, as reported in subsection 5.2, in the analysis.

5.1 DRBD Analysis

The DRBD formalism is a powerful notation to model system reliability, deriving from RBD. If the independence assumption stands, i.e. each component of the DRBD model is stochastically independent from the others, the DRBD model can be considered as a RBD and therefore analyzed by applying the combinatorial structures equations [20], obtaining the total reliability function analyt-

ically. Unfortunately the combinatorial/analytic method cannot be applied or extended to DRBD models with dependence among components. In these cases it is necessary to map the DRBD model onto an analyzable domain, implementing a two step methodology where the DRBD model is an intermediate model interposed between the modeler and the analysis domain, as also done in DFT. Therefore, referring to the DFT case for analogy and inspiring on literature ([17, 15, 24, 5]), a wide range of possibilities are available for analyzing a *"dynamic"* DRBD model by translating it into an analysis domain. As introduced in section 1, the choice is between *analytic* or *simulation* methods. For what regards the *analytic* methods there are many possible alternatives, according to the nature of the reliability cdfs or the overall complexity. The most widely used is the continuous time Markov chain (CTCM), but it is limited in the reliability cdfs tractable. A more general approach should be based on Petri nets (PN), queueing networks, Bayesian networks (BN) or similar analytic formalisms. For example the approaches described in [27, 16] are applications of the PN formalism in DFT analysis. Other interesting techniques to analyze a DFT exploit stochastic reward nets [19, 16], Markov reward models [25], Bayesian networks [29], stochastic well-formed nets [1] and queueing networks [3]. Some of these methodologies remove the restrictions related to the CTMC methods.

For what concerns DRBD, at now, we have developed a solution based on CTMC, with the restrictions mentioned above. To overcome this restrictions we are investigating about a solution based on non-Markovian stochastic PN (NMSPN) [2], applied in [9] and here to analyze the motivating example, but in the actual version also restricted to constant failure rate assumption. For more details refer to [7].

An attractive alternative to analytic approaches is the simulation [28, 13], because it allows the modeling of any reliability distribution without any particular restrictions. One of the drawback of simulation is the long elaboration time needed to achieve accuracy in the solution. However, using variance reductions techniques the elaboration time could be dramatically reduced. The great part of the simulation methodologies applied to the DFT analysis [17, 14], exploit the *Monte Carlo* technique. Since the Monte Carlo simulation approach is very flexible, we retain it could be successfully applicable to the DRBD analysis and we are actually investigating in this way.

However the optimal solution is to combine different techniques [25, 17] by adequately decomposing the DRBD model into independent parts or subsystems. The parts involving dependencies are identified as *dynamic subsystems*. The others are identified as *static subsystems*. The static subsystems are analyzed using the combinatorial/analytic equations, while the dynamic subsystems can be analyzed referring to one of the methods introduced above, according to the nature of the reliability distributions and the complexity of the considered part. After a separated elaboration of each subsystem, the results must be merged by applying the structure equations to the subsystems. This algorithm can be summarized in three steps:

1. *Split the DRBD* in as much static and dynamic subsystems as possible. Each subsystem must be independent from the others.
2. *Analyze the subsystems*: the static ones are analyzed by applying the RBD structure equations, while the dynamic parts are mapped onto an analytic/simulation method, evaluating case by case, according to the nature of the reliability functions contained in the subsystem, the complexity of the subsystem and the expected accuracy.
3. *Rejoin the results* coming from the different elaborations by applying the RBD structure equations.

5.2 Results

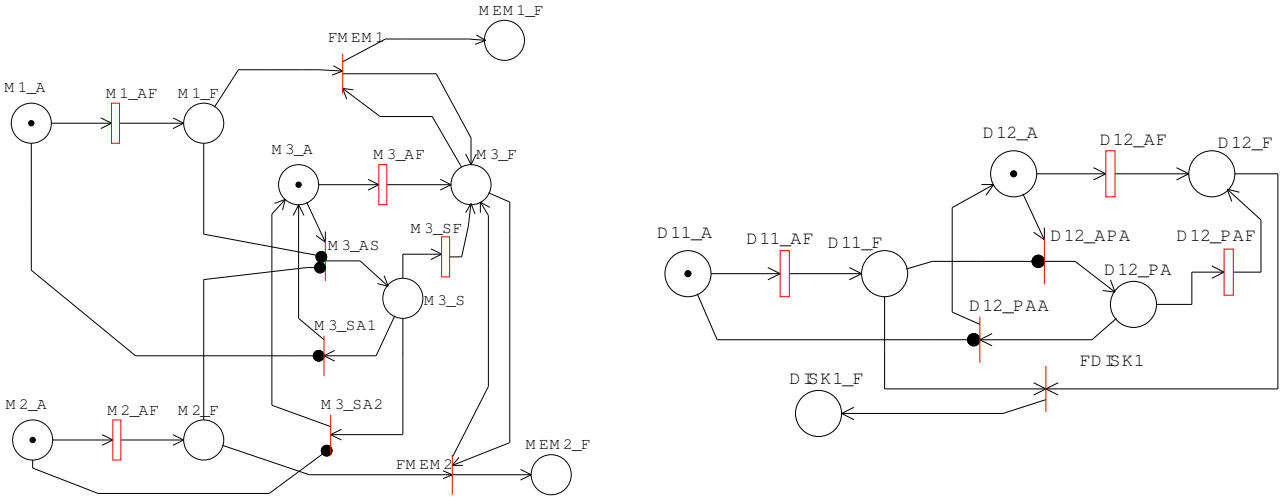
The example described in section 2 and modeled in section 4, has been studied in depth by analyzing the overall system reliability cdf trend in time, knowing the components' reliability cdfs or the corresponding failure rates. All the components have been modeled by a constant failure rate λ characterizing exponential reliability cdfs or *memoryless* systems.

Component	λ	α	β
N	2		
P_1, P_2	500		
PS	6000		
D_{11}, D_{21}	80000	0.5	0.5
D_{12}, D_{22}	80000	0.5	0.5
M_1, M_2	30		
M_3	30	0.5	0.5

Table 1. Parameters related to the multiprocessors computing system example

The values contained in Table 1, drawn from literature ([16, 18]) as the motivating example, report the parameters used to analyze this latter. λ, as introduced above, is the failure rate, α indicates the dormancy factor and β the dependency rate of the dependency between the corresponding components, where $\beta = 1 - \alpha$. The λ values are expressed in *failures in time* (FITs), i.e. number of faults per billion device hours (1 FIT = 1 fault/10^9 hours).

By applying the underlined algorithm to the multiprocessors computing systems DRBD reported in Fig. 5, it can be subdivided in three subsystems: the first, static, is composed by the series among the power supply PS and the



(a) (b)

Figure 6. GSPN modeling the memory (a) and the disk (b) blocks

bus N, series connected with the parallel between the two computing modules CM_1 and CM_2 that are the other two subsystems. Since these latter are identical, it is possible to study only one computing module subsystem and then apply the parallel structure equation to obtain the reliability of the parallel between the two computing modules. A computing module is further subdivided onto the series of three blocks: the processor, the memory and the disk. The memory and the disk blocks are dynamic parts. To study these dynamic parts the generalized SPNs (GSPNs) reported in Fig. 6 are exploited. Analyzing the two GSPNs through the WebSPN tool [22] and putting all together by applying the RBD structures equations, the results shown in Fig. 7 are obtained. Fig. 7 reports the overall multiprocessors computing system reliability cdf.

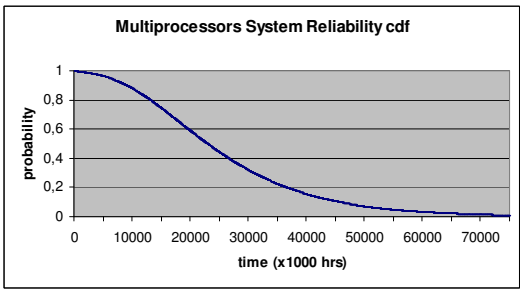


Figure 7. Trend of the system reliability cdf

By the same way, the DFT model depicted in Fig. 4 corresponding to the motivating example discussed, has been analyzed in [18] by exploiting three different tool: DBNet [18], DRPFTproc [1] and Galileo [23]. The first analyzes the DFT by translating it into a dynamic Bayesian network (DBN) and therefore by solving the DBN. DRPFTproc is based on modularization and conversion to stochastic well-formed nets (SWN) of the dynamic gates, tracing back the problem to a SWN solution. Galileo approaches the problem by firstly modularizing it, then solving the obtained modules by exploiting binary decision diagrams and CTMCs.

Time	*DBNet*	*DRPFTproc*	*Galileo*	*DRBD*
1000	0.006009	0.006009	0.006009	0.006009
2000	0.012245	0.012245	0.012245	0.012245
3000	0.019182	0.019183	0.019183	0.019183
4000	0.027352	0.027355	0.027355	0.027354
5000	0.037238	0.037241	0.037241	0.037240

Table 2. Unreliability results obtained analyzing the multiprocessors computing system example

The results obtained by such analysis are summarized in Table 2, where they are compared each other also referring to the DRBD approach. Tab 2 summarizes some system unreliability probabilities calculated in specific time instants. The time is expressed in hours. These results demonstrate and validate the effectiveness of the DRBD approach, providing consistent values for all the tests.

6 Conclusions

In this paper the effectiveness of the DRBD methodology in representing dynamic reliability/availabillity aspects, is demonstrated by exploiting a multiprocessors computing system example drawn from literature. Basing on this latter the DRBD methodology is explained step by step, also establishing a deep comparison with the DFT approach.

The problem to analyze a DRBD model is faced in the paper, describing an interesting algorithm that combines different solutions. This algorithm is applied to the multiprocessors system analysis, in order to obtain the overall system reliability. These results compared to the one obtained by the DFT analysis, allow to identify the DRBD as

a valid alternative in dynamic reliability/availability analysis scenario, also considering its potentiality in flexibility, modeling power an the other capabilities.

References

[1] A. Bobbio, G. Franceschinis, R. Gaeta, and L. Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level petri net semantics. *IEEE Trans. Softw. Eng.*, 29(3):270–287, 2003.

[2] A. Bobbio, A. Puliafito, and M. Telek. A modeling framework to implement preemption policies in non-markovian spns. *IEEE Transaction on Software Engineering*, 26(1):36–54, 2000.

[3] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience, 2^{nd} edition, May 2006.

[4] M. A. Boyd. *Dynamic Fault Tree Models: Techniques for Analysis of Advanced Fault Tolerant Computer Systems*. PhD thesis, Duke University, Department of Computer Science, Apr. 1991.

[5] D. Coppit, K. J. Sullivan, and J. B. Dugan. Formal semantics for computational engineering: A case study on dynamic fault trees. In *ISSRE*, pages 270–282, 2000.

[6] R. Corporation. *System Analysis Reference: Reliability Availability and Optimization*. Reliasoft Publishing, 2003.

[7] S. Distefano. *System Dependability and Performances: Techniques, Methodologies and Tools*. PhD thesis, University of Messina, 2005.

[8] S. Distefano and A. Puliafito. System modeling with dynamic reliability block diagrams. In *Proceedings of the Safety and Reliability Conference (ESREL06)*. ESRA, September 2006.

[9] S. Distefano, M. Scarpa, and A. Puliafito. Modeling distributed computing system reliability with drbd. In *SRDS '06: Proceedings of the 25^{th} IEEE Symposium on Reliable Distributed Systems (SRDS'06)*, pages 106–118, Washington, DC, USA, 2006. IEEE Computer Society.

[10] S. Distefano and L. Xing. A new modeling approach: Dynamic reliability block diagrams. In *Proceedings of the 52^{nd} Annual Reliability and Mantainability Symposium (RAMS06)*. IEEE, January 2006.

[11] J. B. Dugan, S. Bavuso, and M. Boyd. Dynamic fault tree models for fault tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, September 1992.

[12] J. B. Dugan and S. A. Doyle. New results in fault-tree analysis. In *Reliability and Maintainability Symposium*, pages 568 – 573, January 1996. Tutorial Notes.

[13] K. D. Figiel and D. R. Sule. A generalized reliability block diagram (rbd) simulation. In *Simulation Conference*, pages 551 – 556, 1990.

[14] S. G. Gedam and S. Beaudet. Monte carlo simulation using excel(r) spreadsheet for predicting reliability of a complex system. In *Reliability and Maintainability Symposium*, pages 188 – 193, 2000.

[15] M. Malhotra and K. S. Trivedi. Reliability and performability techniques and tools: A survey. In *MMB*, pages 27–48, 1993.

[16] M. Malhotra and K. S. Trivedi. Dependability modeling using petri-nets. *IEEE Transaction on Reliability*, 44(3):428–440, September 1995.

[17] R. Manian, J. B. Dugan, D. Coppit, and K. J. Sullivan. Combining various solution techniques for dynamic fault tree analysis of computer systems. In *IEEE International High-Assurance Systems Engineering Symposium*, 1998.

[18] S. Montani, L. Portinale, A. Bobbio, and D. C. Raiteri. Automatically translating dynamic fault trees into dynamic bayesian networks by means of a software tool. In *Proceedings of the The First International Conference on Availability, Reliability and Security, ARES 2006,*, pages 804–809. IEEE Computer Society, 2006.

[19] J. Muppala, G. Ciardo, and K. Trivedi. Stochastic reward nets for reliability prediction. *Communications in Reliability, Maintainability and Serviceability*, 1(2):9–20, July 1994.

[20] M. Rausand and A. Høyland. *System Reliability Theory: Models, Statistical Methods, and Applications*. Wiley-IEEE, 3^{rd} edition, November 2003.

[21] R. Sahner, K. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-based Approach Using the SHARPE Software Package*. Kluwer Academic Publisher, 1996.

[22] M. Scarpa, A. Puliafito, and S. Distefano. A parallel approach for the solution of non Markovian Petri Nets. In J. Dongarra and D. Laforenza and S. Orlando, editor, *10th European PVM/MPI Users' Group Conference (EuroPVM/MPI03)*, pages 196–203, Venice, Italy, September/October 2003. Springer Verlag - LNCS 2840.

[23] K. J. Sullivan, J. B. Dugan, and D. Coppit. The galileo fault tree analysis tool. In *Proceedings of the 29th Annual International Symposium on Fault-Tolerant Computing*, pages 232–5, Madison, Wisconsin, 15–18 1999. IEEE.

[24] K. S. Trivedi, S. Hunter, S. Garg, and R. Fricks. Reliability analysis techniques explored through a communication network example. In *International Workshop on Computer-Aided Design, Test, and Evaluation for Dependability*, July 1996.

[25] M. Veeraraghavan and K. S. Trivedi. A combinatorial algorithm for performance and reliability analysis using multistate models. *IEEE Trans. Comput.*, 43(2):229–234, 1994.

[26] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. *Fault Tree Handbook*. U. S. Nuclear Regulatory Commission, NUREG-0492, Washington DC, 1981.

[27] V. V. Volovoi. Modeling of system reliability using petri nets with aging tokens. *Reliability Engineering and System Safety*, 84(2):149–161, 2004.

[28] W. Wang, J. M. Loman, R. G. Arno, P. Vassiliou, E. R. Furlong, and D. Ogden. Reliability block diagram simulation techniques applied to the ieee std. 493 standard network. *IEEE Transaction on Industry Applications*, 40(3):887–895, May-June 2004.

[29] Z. Zhou, G. Jin, D. Dong, and J. Zhou. Reliability analysis of multistate systems based on bayesian networks. In *ECBS '06: Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06)*, pages 344–352, Washington, DC, USA, 2006. IEEE Computer Society.