ORIGINAL RESEARCH

Dependability assessment of critical systems

Salvatore Distefano[1,2]

Received: 21 December 2014 / Accepted: 16 March 2015 / Published online: 1 April 2015

Abstract Dependability evaluation is an important, mandatory step in designing and analyzing critical systems. Indeed, in critical systems, it is necessary to take into account not only operational or functional (static) relationships among components, but also non-functional, dynamic ones such as interferences or dependencies. They could be either internal, if arising from interactions among components, or external, if due to the external environment. To properly evaluate critical system dependability, accurate models are therefore required, able to deal with dynamic, dependent behaviors, especially if the system is complex. The main goal of this paper is to identify and specify the dynamic-dependent aspects that can affect the dependability of a critical system. Starting from the concept of dependence at the basis of system decomposition, an analytic framework and some of the most important dynamic-dependent aspects and behaviors are characterized in terms of dynamic reliability.

Keywords Critical systems · Dependability · Dependence · System decomposition · Dynamic reliability

1 Introduction

A system is a set of entities or components interacting to achieve a common goal. It often implements value added services, where the functionalities of-the-whole cannot be provided by a single component. This requires to interconnect through functional or structural connections the system components implementing the interactions required to provide the system functionalities. However, some unintentional, unexpected, often undesired *"emergent"* behaviours could arise from such interactions among subsystems or from those with the external environment. Side effects that can be usually referred to interference, dependence, concurrency, sharing and/or (standby) redundancy policies. This way they can be identified with the term *dependencies*.

Critical systems are a special class of systems providing functionalities whose malfunction, fault or failure may have severe consequence on the equipment or, even worse, on people and/or environment. For these systems, dependability properties such as reliability, availability, maintainability and safety are therefore of topic importance, strict requirements at design time to be satisfied at runtime. To meet these requirements and demand it is therefore mandatory providing and implementing suitable approaches and methods able to take into account dependent-dynamic aspects in critical system dependability assessment. Techniques and tools commonly adopted in dependability evaluation have to be reconsidered, avoiding over-simplistic models that could provide unsatisfactory or even erroneous results since, likely, they do not take into account dependencies. From practice, examples of such behaviors are *load-sharing*, *standby redundancy*, *interferences*, *dependent, on-demand, cascade,* and *common cause events*, *variable operating conditions*, just to name a few.

Thus, it is necessary to adequately represent such aspects through detailed models, exploiting specific techniques. In the case of statistical-independence among components, *combinatorial models/notations* such as *reliability block diagrams* (RBD) (Rausand and Høyland 2003), *fault/event trees* (FT/ET) (Vesely et al. 1981;

✉ Salvatore Distefano
salvatore.distefano@polimi.it; s_distefano@it.kfu.ru

[1] Politecnico di Milano, Milano, Italy

[2] Kazan Federal University, Kazan, Russia

Sahner et al. 1996) and *reliability graphs* (RG) (Sahner et al. 1996) can be used for evaluating critical system dependability attributes. In large-complex systems evaluation, approximated techniques using sequence/path exploration algorithms (minpath sets, mincut sets, binary decision diagrams, noise reduction index algorithms Rausand and Høyland 2003; Vesely et al. 1981; Bouissou and Bonc 2003) should be exploited.

But, unfortunately, the statistic-independence assumption is not satisfied if dependencies, as the ones listed above, affect the system. Therefore, more powerful, usually lower level, techniques and formalisms are needed for modeling and evaluating critical system dependability, such as *state space methods* (Markov models, Petri nets Bolch et al. 2006; Bruneo et al. 2010; Distefano and Trivedi 2013, Boolean logic driven Markov process (BDMP) Bouissou and Bonc 2003, etc.), *hybrid* (combinatorial/state space) *techniques* [(dynamic fault trees (DFT) (Dugan et al. 1992; Chiacchio et al. 2013), dynamic reliability block diagrams (DRBD) (Distefano and Puliafito 2009), OpenSESAME (Walter et al. 2008)] or *simulation* (Marseguerra and Zio 2002).

The main goal of this paper is to identify and characterize dynamic-dependent aspects in dependability evaluation of critical systems. Particular emphasis is given to dependency, due to their importance in critical system dependability modeling. The focus is therefore on studying in depth the dependency concepts at the basis of the system decomposition into subsystems, investigating both the *propagation* and the *application* mechanisms (Distefano 2009, 2011). This work is thus centered on the method used for representing complex dynamic/dependent behaviors, specifying how to compose modular dependencies through the *composition* mechanisms acting as "glue" for the dependency elements. This way, some common dependent-dynamic behaviors are identified and discussed.

Thus, an analytic formal framework for representing critical system dependability is proposed in Sect. 2. Then, in Sect. 3, some specific common dynamic aspects of dynamic-dependent systems are characterized and discussed, while in Sect. 4 (dynamic) dependability modeling techniques are described. Section 5 closes the paper with some final remarks and ideas for future work.

2 Critical system specification

From an abstract, high level perspective, a *system* may be described as an organized, purposeful structure pursuing a specific objective and/or implementing a given function. With specific regard to dependability, "a complex safety critical system is a system whose safety cannot be shown solely by test, whose logic is difficult to comprehend without the aid of analytical tools, and that might directly or indirectly contribute to put human lives at risk, damage the environment, or cause big economical losses" (SAE International 1996).

Abstractly, any system can be characterized by its *boundary* identifying what is inside or belongs to the system and what outside, part of the external *environment*. Since social and psychological phenomena tend to resist to a quantitative modeling by posing basic difficulties already on the plane of boundary identification, in such cases the boundaries between the system and the environment are usually not clearly established.

This characterization allows to consider and investigate a system from two different perspectives, the internal and the external ones. From an *internal* perspective, a system is an interacting group of items (components, entities, elements, objects, factors, members, parts etc.) forming a unified whole pursuing a common objective (Mer 2012). On the other hand, from the *external* perspective, a system is considered as a *black-box*, aiming at investigating its interactions with the environment.

Emergent properties represent one of the most significant challenges for the engineering of complex systems, in particular for critical ones (Bozzano and Villafiorita 2010). They are usually considered as unexpected behaviours arising from both internal interactions among the system components (interdependencies, interferences, etc.) and/or external (components–environment) ones. To some extents, emergent properties can be beneficial for the systems, but they can also be harmful if they undermine important safety requirements. Therefore, they can be particularly dangerous for critical systems, where the consequence of errors, faults or failures can involve people and/or environment, thus resulting catastrophic.

The system approach attempts to view the world in terms of irreducibly integrated systems. It focuses attention on the whole, as well as on the complex interrelationships among its constituent parts. In order to implement and apply such view, it is necessary to clearly identify the subsystems, the constituent parts of the system relevant for it, avoiding, on one hand, in-depth useless specialization and, on the other hand, over-simplifications or rough approximations. In other words, the understanding of a large, complex system almost inevitably requires its *decomposition*. However, system decomposition may have many different meanings and may be applied and implemented in different ways (Distefano 2009; Chiacchio et al. 2013; Distefano and Trivedi 2013), but some common principles and rules can be identified and applied in decomposing a generic system.

The approach suggested in Naylor (1981) is based on the belief that any form of decomposition of large-complex systems is inevitably based, either explicitly or implicitly, on the concept of *dependence*. In particular, decompositions involve identifying *parts* or *subsystems* and saying how they *depend on* one another. In other words, a system can be characterized in terms of *structural* properties, which specify its static-structural organization, identifying the explicit-functional interconnections among the subsystems required for achieving and implementing the system goals, and *dependencies*, which instead are not explicitly related to the functional behaviour of the system, characterizing side effects affecting components and/or subsystems such as interferences among the subsystems as well as among them and the external environment.

This way, we can formally approach the problem following Naylor (1981), where the system is identified as a set $\mathcal{S}$ of parts or subsystems, or sometimes, of variables. A collection of subsets of $\mathcal{S}$, namely $\mathfrak{M}$, is used to model the level of detail and point of view from which the system is observed, where $\mathbf{O} \in \mathfrak{M} | \mathbf{O} \subseteq \mathcal{S}$ is an *observable set*. Observable sets are usually subsystems. Change in point of view or level of detail yields a different collection, say $\mathfrak{M}'$, of observable sets.

The system can be therefore specified, according to the level of detail considered, as the ordered pair

$$(\mathcal{S}, \mathfrak{M}).$$

Thus we can define a *dependency* among, generally, collections of observable sets or subsystems as an ordered triple $(\mathcal{S}, \mathfrak{M}, \mathbb{D})$, where $\mathcal{S}$ is a non empty set of subsystems, $\mathfrak{M}$ is a collection of subsets of $\mathcal{S}$, and $\mathbb{D}$ is a relation from $(2M - \emptyset)$ to $2M$, where $2M$ is the power set of $\mathfrak{M}$ identifying all the possible subsets of elements (subsystems) in $\mathcal{M}$ and therefore the dependencies of the system $(\mathcal{S}, \mathfrak{M})$, while $\emptyset$ denotes the empty collection of observable sets. If $(\mathfrak{A}, \mathfrak{B}) \in \mathbb{D}$, we say that $\mathfrak{A} \subseteq \mathfrak{M}$ depends on $\mathfrak{B} \subseteq \mathfrak{M}$. Moreover, if $\mathfrak{A}$ depends on $\mathfrak{B}$, then it depends on any collection that contains $\mathfrak{B}$, i.e. $(\mathfrak{A}, \mathfrak{B}) \in \mathcal{D}, \forall \mathfrak{B}' \in \mathfrak{M} \supset \mathfrak{B} \rightarrow (\mathfrak{A}, \mathfrak{B}') \in \mathbb{D}$.

By applying this approach, the system is decomposed into a set of independent subsystems $\mathfrak{I}$. Each subsystem or observable set $\mathbf{S_i} \in \mathfrak{I}$, with $i = 1, \ldots, n$ where n is the total number of system parts identified, is characterized by one or more variables that quantify the specific performance and reliability parameters taken into account.

By knowing the system dependencies and its decomposition, it is possible to adequately investigate the system dynamics. A way for expressing the problem is through the dynamical system theory. Dynamics is the study of change, and a *dynamical system* is just a recipe for saying how a system of variables interacts and changes with time. Therefore, starting from the subsystem variables related to its decomposition $\mathbf{S_i} \in \mathfrak{I}$, described through specific functions $\mathbf{F_i}(\cdot) = \{\mathbf{F_i^1}(\cdot), \ldots, \mathbf{F_i^m}(\cdot)\}$, where m is the number of parameters, we want to formally describe the system dynamics.

A dynamical system can be formally represented by a system of equations quantifying the variable changes in time t. The variables are given by

$$\begin{aligned}
\frac{d(\mathbf{F_1(t)})}{dt} &= \mathbf{G_1}(\mathbf{F_1(t)}, \mathbf{F_2(t)}, \ldots, \mathbf{F_n(t)}, \mathbf{t}) \\
\frac{d(\mathbf{F_2(t)})}{dt} &= \mathbf{G_2}(\mathbf{F_1(t)}, \mathbf{F_2(t)}, \ldots, \mathbf{F_n(t)}, \mathbf{t}) \\
\ldots & \ldots \qquad \ldots \\
\frac{d(\mathbf{F_n(t)})}{dt} &= \mathbf{G_n}(\mathbf{F_1(t)}, \mathbf{F_2(t)}, \ldots, \mathbf{F_n(t)}, \mathbf{t})
\end{aligned}$$

and the right hand side of each equation is a function, $\mathbf{G}$, which specifies how fast variables change in time. In general, the rates depend on the values of the other variables. If they depend on each other in a non-linear way the problem can be very hard to approach. Nevertheless, the important point is that as long as we can evaluate the different functions for a given set of variables and time, we can always say something about how the system will evolve. We will use this trick extensively, to show that you can often understand the behavior of the entire systems (sometimes) without even solving the differential equations.

3 Critical aspects: dependencies

Through the above discussion, some specific aspects and behaviors to take into account in critical system dependability assessment have been identified. In this section we specifically focus on such aspects, starting from the basic dependency then characterized into different classes.

As discussed above, the modeling approach adopted is based on the concept of *dependency*. This concept is further characterized here by considering a generic dependency as the simplest dynamic-dependent relationship between two events somehow related to the system, involving either two subsystems or a subsystem and the external environment, as introduced in Distefano (2009). Indeed, dependencies are relationships involving two parties: a subsystem or the external environment driving the dependency (*driver*) and the subsystem *target* of the dependency. Dependencies involving the external environment are always *unidirectional*, *one-way* while, in dependencies between subsystems, reciprocal influences may arise and therefore the dependencies can be either *two-way/mutual* or one-way otherwise.

It is possible to consider a *dependency* as the elementary unit, the building block by which a complex dynamic-dependent behaviour can be represented. This way, a modular approach to represent complex dynamic behaviours as a compositions of simple dependencies, mixing their effects, can be implemented. From this perspective, a dependency is considered as a *cause/effect* relationship. The *cause* of a dependency is referable to the occurrence of a specific *trigger* event related to the driver, i.e. an external environment event or a event occurring to the driver subsystem. As a consequence, the *effect* of a dependency is instead referable to a specific (*reaction*) event stimulated by the former on the target subsystem. The *propagation* of a dependency consists in the application of the effect, i.e. the reaction event, to the target when the cause, i.e. the trigger, occurs. It is characterized by the *propagation probability* p, i.e. the probability a triggered dependency is applied, or also the probability the dependency effect follows up the corresponding cause. The effect of a propagated dependency is instead quantifiable in terms of (reliability, maintenance, availability) distributions or probability functions $F(x)$. When a dependency is applied and propagated to the target dependent subsystem, the trend of the corresponding CDF changes and therefore has to be modeled accordingly, as discussed in the following section.

The *composition* of dependencies is an operation involving two or more subsystems. The result of such operation is still a (*composed*) dependency. The dependencies' composition is implemented by aggregating the trigger events of the simple dependencies into a *"composed"* trigger event, expressed as a condition of the simple triggers. Several basic relationships among events can be identified, broadly classified into two groups:

– *order* ($<, \leq, =, >, \geq, \neq$) specifying the temporal order among occurrences of events;
– *logical* (NOT/$-$, OR/$+$, AND/*, XOR/$\oplus$) specifying logical-Boolean conditions among occurrences of events.

It is also possible to combine such operators managing nesting conditions through brackets. In such cases, it is necessary to specify a priority order among the operators. Table 1 reports and classifies the event operators according to their priority: brackets have the highest priority, while the XOR operator has the lowest one. The *associativity* column specifies the order by which operators with the same priority are evaluated.

Table 1 Priority of event operators

Priority	Operator	Associativity
1	()	Left to right
2	NOT/$-$	Right to left
3	Relational ($<\ \leq\ =\ >\ \geq\ \neq$)	None
4	AND/*	Left to right
5	OR/$+$	Left to right
6	XOR/$\oplus$	Left to right

Through such operators it is possible to compose events that can be triggers of complex dependencies. Any order and any logic condition among events can be represented by their combination.

In terms of effects, the composition among dependencies is classified as: *concurrent/mutually exclusive* or *cooperating/overlapping*. In the former case, the effects of the dependencies to be composed conflict, are mutually exclusive, and therefore only one of them can be applied to the target. This always occurs if the dependencies to be composed have different reactions. A technique to manage and implement the mutual exclusion among concurrent dependencies is to associate priorities with them. If conflicts among concurrent dependencies arise, they are solved by the *priority evaluation algorithm* that univocally establishes the *winner* dependency (Distefano 2009).

On the other hand, a special case of composition among dependencies is characterized by *symbiotic* dependencies. The assumption regulating the composition of symbiotic dependencies is that all the dependencies to be composed must have *compatible* or *equal* reactions. If such assumption is satisfied, the effects of the symbiotic-composed dependencies are merged according to a specific *merging function* quantifying the corresponding impact (Distefano 2009). A behavior that can be represented by cooperating dependencies composition is the load sharing: the load of a single subsystem, and consequently the observed quantity (reliability, availability or performance), depends on how many subsystems sharing the overall load with the former. By modeling the impact of each load sharing subsystem to the others involved through simple dependencies, the overall impact on such subsystem can be represented by a symbiotic composition among the incoming dependencies corresponding to all the involved subsystems.

The approach of composing simple dependencies in order to represent complex dynamic-dependent behaviors is the key point of our modular/*pattern-based* reliability evaluation technique. This powerful and flexible mechanism allows to model several dynamic aspects due to load sharing, changing environments, multi-source interferences, limited resources, dependent, on-demand, cascade events as better specified in the following.

3.1 Dependent events

Dependent events arise from interactions among the subsystems composing a system. A classification of dependent events, starting from the specification of IEC 61508-4 ed2.0 (2010-04), is summarized in Table 2.

Dependent events in reliability are usually related to failures (causal, cascading, common cause/mode failures), although any other event, related to reliability and availability such as repair, standby, disable, activation/re-

Table 2 Dependent, cascade, common cause and common mode events hierarchy

Dependent event	Event whose probability cannot be expressed as the simple product of the unconditional time-to-event probabilities of the individual events which caused it
Common cause event	Event, triggered by other ones, causing coincident events on two or more subsystems (*shared cause*)
Common mode event	Common cause event that triggers similar events (same mode) in multiple subsystems (*shared cause and effect/mode*)
Causal or cascade event	All the dependent event that are not common cause

activation, enable, resume, etc., can be considered (Ficco et al. 2011; Distefano 2011; Distefano et al. 2011). Common cause/mode failures can arise from power supply break-outs, sudden increases of temperature, overheating, electromagnetic interference, radiation, electrostatic discharge, from catastrophic events, etc. Cascade failures mainly regard grid and/or network systems/subsystems. They can also affect, for example, software and real time computing systems.

Dependent events can also affect the system performance. Example of such events are synchronization events (synchronization on data and/or resources, locks, monitors, barriers, etc.), or any generic interaction among subsystems.

For example, dependent events can be used in modeling *load sharing*. The load sharing condition characterizes subsystems that, in performing their tasks, share their workload. Since a less stressed system is usually higher reliable and also better performs, we can argue that its reliability and performance depend on the workload. Therefore, in case of load sharing among two or more subsystems, since the amount of workload managed by each subsystem depends on the number of subsystems sharing the whole workload and on the corresponding power of service, such aspect can be represented as a mutual dependent behavior among the involved subsystems.

3.2 Shared resource

Shared resources are among the primary causes of dependent-dynamic behaviors in complex systems. Dependencies arise from the management of concurrent accesses to such resources, especially when these (accesses or even the resources) are limited, bounded or restricted. Such concept is intuitively valid for performance, since the concurrency reflects into an overhead for the system.

It can be also characterized in terms of reliability and availability. A good example in reliability context is maintenance. Indeed, maintenance strategies have to take into account the *repair resources/facilities* available. In literature this is known as the *repairman problem* (Cox and Smith 1961; Feller 1968; Barlow and Proschan 1965). In case of infinite resources the maintenance/repair policy can be represented by a Cdf describing the time-to-repair random variable. Otherwise, it is necessary to evaluate the interactions due to simultaneous failures in the repairs of the corresponding subsystems. A technique usually exploited is to group subsystems identifying *repair group* sets associated with the corresponding *repairmen* set. In this way, a repair depends on the availability of repairmen and also on the number of failures of subsystems belonging to the same repair group occurring at the same time. Several ways to manage shared repair resources are possible, among the other: processor sharing, FCFS, random next, priority-based, etc.

3.3 Changing environment

Another important dynamic dependent aspect is related to the effects of the external environment into the system. Obviously this strongly depends on the system characterization, on what is comprised in the system specification and what is instead considered external environment. For example, considering a software as a system, composed of different subsystems or modules and functions, the external environment is constituted by the operating system, compilers, libraries, protocols, hypervisor (in case of virtual machine), stakeholders (users, administrator, etc., interacting with the system), underlying hardware resources, including networks and everything required for processing. In such case the external environment can significantly affect the system performance and reliability, and a change in the environment could have consequence on the overall system. In order to represent such aspect, it is necessary to take into account the effect of the environment on each dependent subsystem, explicitly specifying them in the quantities to be evaluated.

Good references on such specific topics are (Finkelstein 1999, 2008) dealing with this problem in reliability and availability contexts.

4 Dependability evaluation

From an operational point of view, dependability is an aggregate property composed of different attributes, at least including reliability, maintainability and availability. Reliability is associated with the *continuity* of service, *maintainability* concerns the possibility to modify or repair the system, while availability is related to the *readiness* of a system or a service (Avizienis et al. 2004). In critical systems a dependability attribute of strategic importance is safety, i.e., the property of preventing or avoiding catastrophic consequences of system faults or failures for users or the environment.

To achieve dependability goals in critical systems, specific policies based on redundancy and spare management such as standby or maintenance strategies are required, thus falling into the dynamic context above identified. The effects of such policies have to be therefore evaluated according to the global *dynamic system view*. In this context a subsystem or the overall system is identified as *static* or *dynamic* whether dependencies involves its subsystems or not, respectively. Thus, to evaluate the system reliability and availability it is requested to adequately investigate the subsystem dynamics, individuating and quantifying all the dependent behaviors above specified.

In order to evaluate a critical system dependability the corresponding system quantity, i.e. its reliability, and the associated lifetime random variable X are taken into account. Assuming this quantity is influenced by another event, either internal or external as discussed above, which can be represented and enumerated through a discrete number of *operating conditions* $c_X^i \in \mathbf{C_X}$ in the set $\mathbf{C_X}$, X can be characterized by a function $F_X^i : \mathbb{R} \to [0,1]$ stochastically describing its behaviour when the system is subjected to the ith operating condition. F_X^i is a generic, continuous, and strictly monotone function, the cumulative distribution function (CDF) associated with the system lifetime X random variable when the system works permanently in the ith operating condition c_X^i, in *isolation*.

Then, let C be another continuous random characterizing the operating condition evolution and dynamics. This way, X depends on C that triggers the operating condition switching thus affecting on X. More specifically, the values assumed by C are directly associated with the conditions $\forall c_X^i \in \mathbf{C_X}$, i.e. $\mathbf{C_X}$ is the domain of the C random variable. Thus, we aim at obtaining the *overall* function $F_X(t, c_X)$ that characterizes the quantity related to X when time t and condition c_X vary, starting from the corresponding functions in isolation of $\mathbf{F_X}$. Since the condition switching is related to the value of C, it could be considered as just a function of time t such that $F_X(t, c_X) = F_{X,C}(t)$.

The main requirement we impose on $F_X(t, c_X)$ is that it must be a continuous function in t also at operating condition c_X changing. Thus, the problem is to obtain $F_X(t, c_X)$ highlighting the relationships with $F_X^i(t) \in \mathbf{F_X}$ when condition c_X^i is applied is isolation.

To formally deal with such issue, only the case of a system characterized by two operating conditions for X, associated with the $F_X^1(t)$ and $F_X^2(t)$ functions in isolation, respectively, is considered. At the beginning the system is in condition c_1 and then switches to condition c_2 at $t = y$. Then, $F_{X,C}(t)$ can be formulated as

$$F_{X,C}(t) = \begin{cases} F_X^1(t) & t \le y \\ F_X^2(t-\tau) & t > y \end{cases} \tag{1}$$

where $\tau \in \mathbb{R}$ is a constant depending on the changing point y such that $t, y, t-\tau \in \mathbb{R}$ and $F_{X,C}(t)$ is continuous at $t = y$.

This way, the behavior at changing point has to be specified to quantify τ. Given that $F_{X,C}(t)$ must be continuous and $F_X^1(t)$ and $F_X^2(t)$ are strictly decreasing (and thus invertible), at $t = y$ by Eq. (1) we have that:

$$F_X^1(y) = F_X^2(y-\tau) \Rightarrow \tau = y - F_X^{2(-1)}\left(F_X^1(y)\right) \tag{2}$$

where $F_X^{2(-1)}(\cdot)$ is the inverse function of $F_X^2(\cdot)$, thus assuming it is invertible.

If the distribution of the operating condition changing process $F_C(t) = \Pr\{C \le t\}$ is given, where $C =$ *time to* $c_1 - c_2$ *switching* as in the example considering just one switch, it is possible to obtain $F_{X,C}(t)$ by applying the law of total probability:

$$\begin{aligned} F_{X,C}(t) &= \int_{-\infty}^{+\infty} \Pr\{X \le t | C = y\} f_C(y) dy \\ &= \int_0^t \Pr\{X \le t | C = y\} f_C(y) dy \\ &\quad + \int_t^{+\infty} \Pr\{X \le t | C = y\} f_C(y) dy \\ &= \int_0^t (1 - \Pr\{X > t | C = y\}) f_C(y) dy \\ &\quad + F_X^1(t)(F_C(y)|_t^{\infty}) \end{aligned} \tag{3}$$

where $f_C(t) = \frac{dF_C(t)}{dt}$ is the probability density function of C.

This way, considering $y \le t$, $\Pr\{X > t | C = y\} = 1 - F_X^2(t-\tau) = 1 - F_X^2(t + F_X^{2(-1)}(F_X^1(y)) - y)$ and thus by Eq. (3)

$$\begin{aligned} F_{X,C}(t) &= F_X^1(t)(1 - F_Y(t)) \\ &\quad + \int_0^t F_X^2\left(t + F_X^{2(-1)}\left(F_X^1(y)\right) - y\right) f_C(y) dy \end{aligned} \tag{4}$$

A graphical description of the specification above discussed and detailed by Eqs. (1) and (2) is shown in Fig. 1.

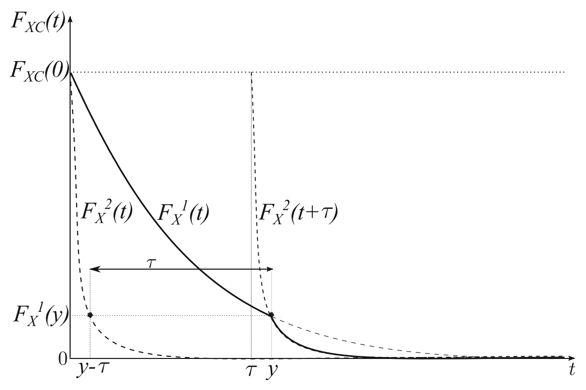

Fig. 1 Critical system reliability $F_{X,C}(t)$ of the example considering one changing point

5 Conclusions

This paper focuses on the dependability evaluation of critical systems, identifying the main aspects and behaviours affecting the related quantities. In particular non functional, dependent relationships among components and/or the external environment are taken into account. Starting from the concept of dependence and following the system reliability engineering approach, the decomposition into subsystems is first proposed. Then, the concept of *dependency* is specified as the basis of a modular approach to dynamic reliability by which complex dependent behaviours, mainly affecting or to be taken into account in critical systems assessment, can be expressed. This way, an analytical framework for dealing with simple dependencies is provided as a first analytical tool for the in depth evaluation of critical system dependability.

The application of this technique to real case study is work in progress. Further developments are focusing on numerical techniques for the solution of the analytical framework, also dealing with largeness in terms of number of components and subsystems and complexities in terms of dependent behaviours and dependencies.

Conflict of interest The author declares that he has no conflict of interest.

References

Avizienis A, Laprie J-C, Randell B, Landwehr C (2004) Basic concepts and taxonomy of dependable and secure computing. IEEE Trans Dependable Secur Comput 1:11–33. doi:10.1109/TDSC.2004.2

Barlow RE, Proschan F (1965) Mathematical theory of reliability. Classics in Applied Mathematics. Wiley, New York

Bolch G, Greiner S, de Meer H, Trivedi KS (2006) Queueing networks and Markov chains: modeling and performance evaluation with computer science applications, 2nd edn. Wiley-Interscience, Hoboken, NJ

Bouissou M, Bonc J-L (2003) A new formalism that combines advantages of fault-trees and markov models: Boolean logic driven Markov processes. Reliab Eng Syst Saf 82(2):149–163

Bozzano M, Villafiorita A (2010) Design and safety assessment of critical systems. CRC Press (Taylor and Francis), an Auerbach Book

Bruneo D, Distefano S, Longo F, Scarpa M (2010) Qos assessment of ws-bpel processes through non-markovian stochastic petri nets. In: Parallel and distributed processing (IPDPS), 2010 IEEE international symposium on, pp 1–12, IEEE

Chiacchio F, Cacioppo M, D'Urso D, Manno G, Trapani N, Compagno L (2003) A weibull-based compositional approach for hierarchical dynamic fault trees. Reliab Eng Syst Saf 109(0):45–52. doi:10.1016/j.ress.2012.07.005

Cox DR, Smith WL (1961) Queues. Wiley, New York

Distefano S (2009) How to capture dynamic behaviours of dependable systems. IJPEDS 24(2):127–150

Distefano S (2011) The standby engineering: classification and quantification of standby in reliability. Int J Syst Assur Eng Manag 2(4):333–341

Distefano S, Puliafito A (2009) Dependability evaluation with dynamic reliability block diagrams and dynamic fault trees. IEEE Trans Dependable Secur Comput 6(1):4–17

Distefano S, Trivedi KS (2013) Non-markovian state-space models in dependability evaluation. Qual Reliab Eng Int 29(2):225–239

Distefano S, Filieri A, Ghezzi C, Mirandola R (2011) A compositional method for reliability analysis of workflows affected by multiple failure modes. In: Proceedings of the 14th international ACM Sigsoft symposium on component based software engineering, pp 149–158, ACM

Dugan JB, Bavuso S, Boyd M (1992) Dynamic fault tree models for fault tolerant computer systems. IEEE Trans Reliab 41(3):363–377

Feller W (1968) An introduction to probability theory and its applications. Wiley, New York

Ficco M, Daidone A, Coppolino L, Romano L, Bondavalli A (2011) An event correlation approach for fault diagnosis in scada infrastructures. In: Proceedings of the 13th European workshop on dependable computing, EWDC '11, pp 15–20, New York, NY, USA. ACM. doi:10.1145/1978582.1978586

Finkelstein M (2008) Failure rate modelling for reliability and risk. Springer Series in Reliability Engineering. Springer Verlag, Berlin

Finkelstein MS (1999) Wearing-out of components in a variable environment. Reliab Eng Syst Saf 66(3):235–242. doi:10.1016/S0951-8320(99)00023-X

IEC 61508-4 ed2.0 (2010-04) Functional safety of electrical/electronic/programmable electronic safety-related systems—Part 4: Definitions and abbreviations, International Electrotechnical Commission

Marseguerra M, Zio E (2002) Basics of the MonteCarlo method with application to system reliability. LiLoLe Verlag, Hagen, Germany

Merriam-Webster Dictionary (2012) Merriam-Webster, Incorporated

Naylor AW (1981) On decomposition theory: generalized dependence. Syst Man Cybern IEEE Trans 11(10):699–713. doi:10.1109/TSMC.1981.4308590

Rausand M, Høyland A (2003) System reliability theory: models, statistical methods, and applications, 3rd edn. Wiley-IEE, New York

SAE International (1996) Guidelines for development of civil aircraft and system. http://standards.sae.org/arp4754a/

Sahner R, Trivedi KS, Puliafito A (1996) Performance and reliability analysis of computer systems: an exeample-based approach using the SHARPE software package. Kluwer Academic Publisher, Dordrecht

Vesely WE, Goldberg FF, Roberts NH, Haasl DF (1981) Fault tree handbook. U.S. Nuclear Regulatory Commission, NUREG-0492, Washington, DC

Walter M, Siegle M, Bode A (2008) OpenSESAME—the simple but extensive, structured availability modeling environment. Reliab Eng Syst Saf 93(6):857–873. doi:10.1016/j.ress.2007.03.034