Available online at www.sciencedirect.com
SCIENCE DIRECT®
The Journal of Systems and Software 79 (2006) 107–119
The Journal of Systems and Software
www.elsevier.com/locate/jss

A practical framework for eliciting and modeling system dependability requirements: Experience from the NASA high dependability computing project

Paolo Donzelli [a,*], Victor Basili [a,b]

[a] Department of Computer Science, University of Maryland, College Park, MD 20742, USA
[b] Fraunhofer Center for Experimental Software Engineering, College Park, MD 20742, USA

Received 9 December 2004; received in revised form 21 March 2005; accepted 21 March 2005
Available online 29 April 2005

Abstract

The dependability of a system is contextually subjective and reflects the particular stakeholder's needs. In different circumstances, the focus will be on different system properties, e.g., availability, real-time response, ability to avoid catastrophic failures, and prevention of deliberate intrusions, as well as different levels of adherence to such properties. Close involvement from stakeholders is thus crucial during the elicitation and definition of dependability requirements. In this paper, we suggest a practical framework for eliciting and modeling dependability requirements devised to support and improve stakeholders' participation. The framework is designed around a basic modeling language that analysts and stakeholders can adopt as a common tool for discussing dependability, and adapt for precise (possibly measurable) requirements. An air traffic control system, adopted as testbed within the NASA High Dependability Computing Project, is used as a case study.

Keywords: System dependability; Requirements elicitation; Non-functional requirements

1. Introduction

Individuals and organizations increasingly use sophisticated software systems from which they demand great reliance. "Reliance" is contextually subjective and depends on the particular users' needs. Therefore, in different circumstances, the users will focus on different properties of such systems, e.g., availability, real-time response, ability to avoid catastrophic failures, capability of resisting adverse conditions, and prevention of deliberate intrusions, as well as different levels of adherence to such properties. For example, while safety-critical applications development strives to guarantee absence of failures (with higher costs, longer time to market and slower innovations) (Knight, 2002; Littlewood and Stringini, 2000), everyday software (mobile phones, PDAs, etc.) must provide cost effective service with reasonably low failures rates (i.e., to be "sufficiently correct" rather than "correct") (Boehm et al., 2004; Boehm and Huang, 2003; Shaw, 2002).

The concept of *dependability* enables these various concerns to be subsumed within a single conceptual framework. The International Federation for Information Processing WG-10.4 (IFIP) defines dependability as *the trustworthiness of a computing system that allows reliance to be justifiably placed on the services it delivers*.

Achieving systems dependability is a major challenge that has spawned many efforts at the national and international level, including, for example, the European Initiative on Dependability (European Union, 2002), the

* Corresponding author. Tel./fax: +1 3014052740.
E-mail addresses: donzelli@cs.umd.edu (P. Donzelli), basili@cs.umd.edu (V. Basili).

0164-1212/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2005.03.011

US National Strategy to Secure Cyberspace (US, 2003), and the Critical Infrastructures improvement and protection initiatives adopted by various countries (Moteff et al., 2003; Wenger et al., 2004). The work we present here is part of the High Dependability Computing Project (HDCP, 2002), a five-year cooperative research agreement between NASA and various universities and research centers to increase NASA's ability to engineer highly dependable software systems. The Project involves: (a) understanding NASAs dependability problems; (b) developing new dependability modeling mechanisms, engineering practices, and technologies to address such problems; (c) empirically assessing (and iteratively improving) the capabilities of new practices and technologies, using realistic testbeds; (d) transferring technologies to technology users with clear indications about their effectiveness under varying conditions.

HDCP brings together, under the common goal of improving systems dependability, a large and heterogeneous group of actors, from government and academia alike, with various perspectives and different (sometimes even conflicting) needs:

- The system users, who are concerned mainly about the final system's behavior, and who need to understand whether or not, and to what extent, they can depend upon a system to achieve their goals.
- The system developers (or technology users), who need to know which processes and or technologies should be selected to meet the system users' needs in the most efficient and effective way.
- The technology researchers/developers, who focus on specific approaches to develop dependable systems and need to know where new technological capabilities are necessary.
- The empirical researchers, who act as ''observers'', to measure and make explicit what is achievable and what has been achieved, in order to improve the transfer of knowledge (needs, opportunities, technologies' capabilities and limits) among the other actors.

The success of the project depends on the synergistic collaboration of all these actors. For this reason, at the University of Maryland we have developed a framework for modeling dependability (hereafter referred to as UMD) that the different HDCP actors can adopt as a common language to specify, communicate, and understand dependability requirements and dependability achievement of individual systems. This, we believe, could be beneficial to all the involved actors, enabling them to better focus their specific activities.

UMD is designed around a basic modeling language that stakeholders and analysts may use to identify and make explicit (i.e., measurable) the dependability properties a system must possess in order to satisfy the specific needs of its application context.

This paper is organized as follows. Section 2 introduces UMD, by illustrating its underlying concepts in the context of the current view of dependability provided by the literature. This section concludes with a brief description of the web-based tool developed to implement it. Section 3 describes the case study, its implementation and the obtained results. The goal is twofold: perform a feasibility analysis of the UMD concept while eliciting and modeling the dependability requirements for an air traffic control system, adopted as testbed within HDCP (Asgari et al., 2004; Dennis, 2003; HDCP, 2002). Finally, Section 4 concludes and provides an outline of future work.

2. The dependability-modeling framework

Dependability is commonly recognized as an integrative concept that encompasses different attributes (Avizienis et al., 2001; Basili et al., 2004a,b; Boehm et al., 2003; Littlewood and Stringini, 2000; Melhart and White, 2000); little consensus, however, has been reached on which attributes should be considered. For example, Littlewood and Stringini (2000) suggest that dependability comprises reliability, safety, security and availability. Laprie et al. (Avizienis et al., 2001; Laprie, 1992) define dependability as a combination of availability, reliability, safety, confidentiality, integrity, and maintainability. Here, in line with the security community's view (US DOD, 1985), security is not seen as a separate attribute, but is defined as a combination of availability (i.e., availability for authorized users only), confidentiality (i.e., absence of unauthorized disclosure of information), and integrity (i.e., absence of unauthorized system alterations). In addition, the attribute of maintainability is introduced, defined as the system's ability to undergo repairs and modifications (i.e., corrective, perfective and adaptive maintenance).

Other authors, although recognizing the relevance of maintainability (the ability of the system to be easily and quickly repaired is, in fact, strictly related to availability, and faults are often introduced into a system as results of maintenance problems), consider it a static system property and thus distinct from the other dimensions of dependability (reliability, safety, security, and availability). To express the capability of a system to be quickly repaired, thus minimizing the disruption caused by a failure, the attribute reparability is sometimes suggested (Sommerville, 2004).

Maintainability is not the only property to appear occasionally among the dependability attributes. For example, accuracy and performance are suggested in (Boehm et al., 2003); responsiveness is indicated as a key attribute of dependability in (Walkerdine et al., 2004); real-time performance and interoperability are introduced in (Weinstock et al., 2004).

Another dependability attribute subject to alternate vicissitudes is survivability. According to Laprie et al. (Avizienis et al., 2001; Laprie, 1992), in fact, survivability represents another name for dependability (as well as trustworthiness); for other authors (Melhart and White, 2000; Sommerville, 2004) it instead plays an independent role, highlighting potential threats to the system (deliberate or accidental attacks). In particular, Sommerville (2004) establishes a link between security and survivability, both taking into account external attacks. The situation is further complicated by the fact that each of these dependability attributes is defined in the literature in a variety of ways. In particular, differing research communities (reliability, safety, dependability, critical infrastructures (Wenger et al., 2004), security, requirements engineering (Chung et al., 2000), etc.) use different definitions and terminologies (Randel, 1998). Moreover, when referring to different attributes, definitions often overlap, resulting in different communities claiming their right to specific problems (Bass et al., 2003). For example, in (Knight, 2002) it is observed that a security failure in an information system could lead to considerable potential losses, becoming in this way a safety-critical issue as well.

UMD attempts to overcome the difficulty of defining dependability requirements by introducing a modeling language that adopts a small set of basic dependability concepts to facilitate stakeholders' identification and precise formulation of their needs.

2.1. UMD basic concepts

In order to begin our analysis to derive UMD, let us consider the attributes: reliability, availability, safety, security, and survivability. It is important to note that this choice is purely arbitrary, and any other set could have been adopted. In the following (Section 2.6) we will show that our results are independent from the selected set. As mentioned above, for each of these attributes different definitions are available in the literature, and any of them could have been chosen. Again, it is important to note that our choice is purely arbitrary. In the following we will show that UMD is independent from the selected definitions. On this basis, we have selected the following definitions:

- *Reliability* is the continuity of correct service (a service is correct when it implements its specification) (Avizienis et al., 2001).
- *Availability* is readiness for correct service (Avizienis et al., 2001).
- *Safety* is freedom from those conditions (hazards) that can cause death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment (US DOD, 2000).
- *Security* is the ability of the system to deliver its required service without unauthorized disclosure or alteration of sensitive information, or denial of service to legitimate users by unauthorized persons (Mellor, 1992).
- *Survivability* is the ability of a system to continue to deliver its services to users in the face of deliberate or accidental attack (Sommerville, 2004).

Availability and reliability emphasize the absence of failures. A failure occurs when the delivered service deviates from correct service (Avizienis et al., 2001; Sommerville, 2004). As stated in (Avizienis et al., 2001), reliability and availability are closer to each other than they are to safety and security. They represent different views of the same phenomenon, with availability taking the repair time into account.

Safety emphasizes the avoidance of specific situations (hazards) that could have serious consequences on the user(s), properties or the environment. Safety and reliability are related but distinct (Avizienis et al., 2001). In general, reliability is a necessary but insufficient condition for system safety: reliability is concerned with conformance of the system to its specification, while safety is concerned that the system cannot cause damage irrespective of whether or not it conforms to its specification. Hazard and failure are not exclusive. A failure may be a hazard (i.e., a failure that could have serious consequences on the user(s), properties or the environment), but a hazard can also occur without a failure occurring (Sommerville, 2004).

Security reflects the system's ability to protect itself from accidental conditions or deliberate external attacks that could lead to alteration or disclosure of sensitive information, or to normal services being unavailable or significantly degraded. Security is a pre-requisite for reliability: security emphasizes the avoidance of specific types of failures (i.e., disclosure or alteration of sensitive information, incorrect or degraded service) due or not to external attacks (e.g., denial of service attack).

Finally, survivability represents the ability of the system to provide its service despite external attacks. Like security, survivability is a prerequisite for reliability. Survivability, in fact, emphasizes the avoidance of a specific class of failures, i.e., failures resulting from deliberate or accidental attacks.

All the above definitions share some common concepts. First, each of these definitions emphasizes the absence of issues (or of a specific class of issues) that the system could cause to the users. An *issue* may be a failure and/or a hazard. For example, reliability emphasizes the absence of any class of failure; survivability emphasizes the absence of a specific class of failures (i.e., failures resulting from deliberate or accidental external attacks); safety emphasizes the absence of any class of hazard. Then, an issue may be characterized by its *scope*,

given that it may concern the whole system or only a specific service. Finally, as pointed out by the definitions of survivability and security, an issue could also be the result of an external *event*, i.e., any deliberate or accidental external condition or action harmful for the system.

Due to the commonality across the various definitions, we have decided to adopt the concepts of issue, scope and event as the basic elements of our dependability modeling language, allowing stakeholders to express their dependability needs in terms of issue, scope and event. In particular, through UMD, stakeholders operationalize their *dependability requirements* by specifying the actual issue (or class of issues) that should not affect the system or a service (scope). In addition, by using the concept of event, stakeholders may also specify (when appropriate) external events that could be harmful for the system and describe their possible impact on the system in terms of resulting issues and corresponding scope.

Example. An example dependability requirement built using UMD for an on-line bookstore is (Fig. 1): "The search service (scope) should not have a response time greater than 10 s (issue) in case of denial of service attack (event)."

2.2. Using UMD to elicit requirements

While for a stakeholder it could be difficult to define dependability, or to provide a clear definition of what a dependability attribute means for a specific system, it is easier to think in terms of issues, scope and events. Put another way, the concepts of issue, scope and event reduce the complexity of the problem, so that stakeholders, rather than dealing with abstract entities (dependability and its attributes), can organize their thoughts about dependability by focusing on the issues that should not affect the system or specific services (scope), together with the possible triggering external events. UMD thus transfers the dependability definition problem towards a more concrete level, as schematized in Fig. 2.

The expressiveness of issue, scope and event stems from the fact that they are basic concepts that stakeholders can easily grasp and associate with entities proper their application domain. For example, while referring to an office automation application, stakeholders can associate the UMD concept of event with situations that are typical of an office environment, such as power outages, internet breakdowns, novice users, and so on. The case for an avionics application would be completely different. Here external events could be adverse weather conditions, lack of on-ground support, communication breakdowns, and so on.

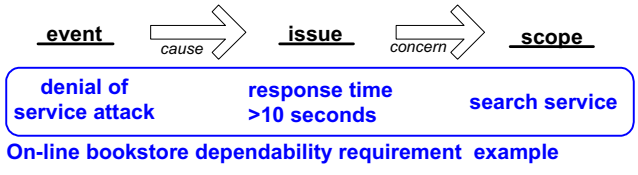

Fig. 1. UMD basic concepts.

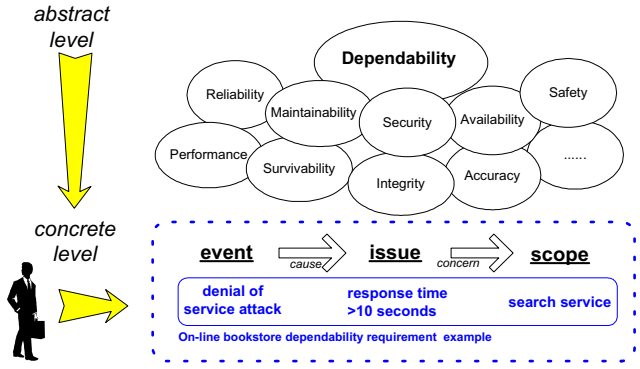

Fig. 2. UMD makes the "dependability definition" problem more concrete.

2.3. Decomposing UMD concepts to improve guidance to the stakeholders

Although the concepts of event, scope, and issue bear a "sufficiently" precise meaning to help stakeholders focus their analysis, we may further refine them to provide better guidance and allow for the definition of more precise requirements. For example, to help stakeholders identify what issues should not affect the system or a specific service, we may suggest the different types of failures that could occur (e.g., response time failures, accuracy failures, disclosures of sensitive information, etc.). Similarly, to support stakeholders in better expressing their concerns, we may further characterize issues according to their severity. For example, we may distinguish failures according to their impact on the utility of the system or service for the stakeholders (e.g., high and low severity), or according their impact on the availability of the system or service (e.g., stopping and non-stopping failures). The same can be repeated for the hazards, as we could distinguish among different severity levels depending on the potential consequences (e.g., hazards could concern only properties or people as well). Then, we can characterize the scope so stakeholders can focus on the whole system or a specific service, but also specify operational profile (e.g., workload, transaction distribution, etc.), for the whole system or a specific service, during which a specific issue should be avoided. Finally, characterization can also be adopted for the concept of event, to enable stakeholders to better recognize and specify external situations that could harm the system. For example, we could classify exter-

nal events into attacks and adverse conditions, to separate deliberate from accidental ones.

The characterizations of the concepts of issue, event and scope (e.g., failures types and severity levels) represent more sophisticated elements of our modeling language that stakeholders may adopt to better specify their needs (e.g., using failures types), and their concerns (e.g., using failures severity levels).

It is important to note that these characterizations depend on the specific context (project and stakeholders), and can be customized to better reflect the application domain needs, and better guide the stakeholders. Stakeholders play a crucial role in this customization process. In fact, for example, while identifying and describing issues and events on the basis of the available types, they may also add new types that better reflect their needs.

As illustrated in Fig. 3, UMD consists of both *invariant concepts* (i.e., issue, scope, and event) and *customizable concepts* (i.e., the characterizations of the invariant ones) that can be introduced as further modeling language items to support the elicitation activity.

2.4. Quantifying issues

UMD in its basic structure allows stakeholders to specify the issues they do not want to occur. This, however, is not enough; we need an operational definition of dependability. For this reason, it is important to allow stakeholders not only to identify the undesired issues, but also to quantify what they assume could be the tolerable corresponding manifestations.

The concept of *measure* introduced into UMD (Fig. 3) achieves this purpose. While measure is an invariant concept of our modeling language, it can be further refined to better support stakeholders in formulating their needs. In Fig. 3, for example, the following types of measurement models have been introduced:

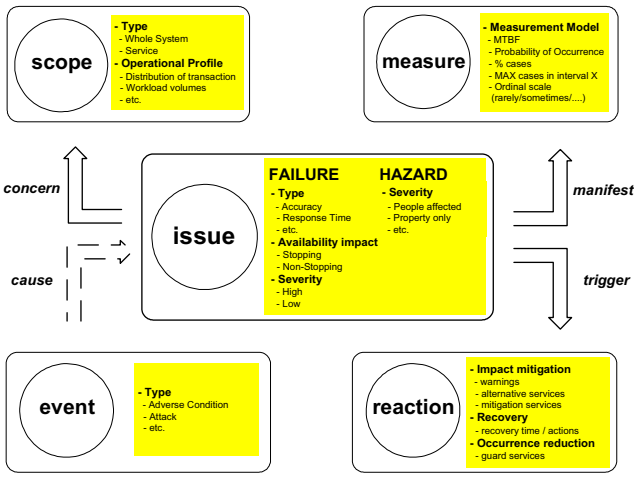

Fig. 3. UMD concepts and relationships.

- *Ratio and probabilistic measures*, such as mean time to failure (MTTF), probability of occurrence (e.g., in next time unit or transaction), percentile of cases.
- *Ratio and deterministic measures*, such as maximum number of occurrences (in a given time frame).
- *Ordinal and probabilistic measures*, for example an ordinal scale such as "very rarely/rarely/sometimes".

Example. By extending the example in the previous section, the dependability requirement will not simply be "the search service (scope) should not have a response time greater than 10 s (issue) in case of denial of service attack (event)", but, more precisely, stakeholders could say that "the search service (scope) should not have a response time greater than 15 s (issue) in case of denial of service attack (event) more often than 1% of the cases (measure)".

2.5. Improving dependability by specifying the desired "system reaction"

UMD allows stakeholders to express their views of dependability in terms of acceptable manifestations of issues. But it also gives stakeholders the opportunity to provide ideas to improve dependability. For this reason, the concept of *reaction* is introduced. In this way, stakeholders may indicate what they assume are the services the system should provide (as reaction to the issue) in order to become more dependable. Again, while reaction is an invariant concept of our modeling language, it can be further refined to better support stakeholders in formulating their requirements. In Fig. 3, for example, the following classification is proposed:

- *Warning services:* to warn users about what happened or is happening (e.g., in case of response time greater than 15 s, warn the user about the delay).
- *Mitigation services:* to reduce the impact of the issue on the users (e.g., in case of response time greater than 15 s, suggest a better time to try again).
- *Alternative services:* to help users to carry on their tasks regardless of the issue (e.g., suggest user call customer service).
- *Guard services:* to act as guard against the issue, i.e., may reduce the probability of occurrence (e.g., preventing delay due to system saturation or trashing by rejecting incoming requests). This idea can be extended to capture any possible suggestion the stakeholder might have to prevent the issue from happening: suggestions about modifications of existing services, design changes, etc.
- *Recovery behavior:* the time necessary to recover from the issue (e.g., expressed as mean time to recover—MTTR) and the kind of required intervention (e.g., user, technician, or automatic).

Example. By completing the example introduced in the previous sections, stakeholders will not only state that "the search service (scope) should not have a response time greater than 10 s (issue) in case of denial of service attack (event) more often than 1% of the cases (measure)", but also that if the failure occurs, the system should provide (reaction) a warning service: "user should be notified about the problem", a mitigation service: "user should be able to save the completed work", and be able to recover in 1 h.

2.6. "Expressiveness" of UMD

In order to build UMD, we have begun our analysis by considering a precise set of dependability attributes (reliability, availability, safety, security, and survivability), and choosing for each one of the many definitions available in the literature.

In this section, we want to show that our framework is independent from these initial choices; in particular we want to show that UMD is capable of expressing dependability requirements formulated according to: (a) different definitions of the dependability attributes already taken into account; (b) definitions of other dependability attributes not included in the initial set.

Let us start by considering different definitions for the same dependability attributes. More examples are discussed in (Basili et al., 2004a,b).

- *Availability* is the capability to maximize the amount of time during which the system will provide stakeholder-desired levels of service with respect to a system's operational profile (probability distribution of transaction frequencies, task complexities, workload volumes, others) (Boehm et al., 2003).
- *Security* is the capability of the system to minimize the expected value of the information, property, human life and health losses due to adversarial causes (Boehm et al., 2003).
- *Survivability* is the capability of a system to accomplish its mission despite a man-made hostile environment, i.e., the system's ability to detect and withstand an attack (Melhart and White, 2000).

We can observe that these definitions can be built around the invariant UMD concepts of issue, event, and scope. For example, we can say that the definition of availability emphasizes the avoidance of degradation of services (issues), regarding the whole system or a specific service (scope). Similarly, the definition of survivability emphasizes the capability of the system or one of its services (scope) to perform as expected (absence of issue) also in case of hostile external conditions (event). UMD allows model requirements expressed according to any of these definitions. In particular, each of these definitions emphasizes specific new types of issues and external events. For example, the definition of survivability clearly states that only man-made attacks should be taken into account, while the definition considered in our initial set referred to both deliberate and accidental attacks. We have already pointed out that many definitions of the same attributes are available in the literature, and each of them could be considered correct. It is up to the stakeholders to make the most appropriate choice according to their needs.

Here, we observe that our framework is capable of capturing these differences among the various definitions, allowing stakeholders to express their dependability needs using concepts as close as possible to their domain. UMD, in fact, can embed these differences (and above all make them explicit) through its customizable concepts. In other terms, it is possible to adopt different characterizations of the invariant concepts (e.g., different types of failures, external events, etc.) to reflect these needs, for example, by introducing the "operational profile description" as a further element of the scope's characterization. Such extension allows for the accommodation of different definitions of dependability attributes where the use conditions or the operation profile for the system or a service are taken into account (see the definition of availability above).

At this point, to complete our evaluation of the UMD expressiveness, we take into account additional attributes of dependability. Other examples are discussed in (Basili et al., 2004a,b).

- *Robustness* is the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions (Melhart and White, 2000).
- *Performance* is a static or dynamic system's capability (response time, throughput, etc.) defined in terms of an acceptable range (Sommerville, 2004).
- *Accuracy* is the ability of the system to minimize the differences between delivered computational results and the real world quantities that they represent (Boehm et al., 2003).
- *Integrity* is the absence of improper system state alterations (Avizienis et al., 2001).

Again, we can observe that these definitions can be built around the invariant UMD concepts of issue, event, and scope. For example, we can say that the definition of performance emphasizes the absence of failures related with response time and throughput (issue), concerning the system or any of its services (scope). Similarly, the definition of accuracy emphasizes the capability of the system or one of its services (scope) to minimize differences between computational results and the represented real world quantities. Finally, the

definition of robustness emphasizes the ability of the system or one of its components (scope) to function correctly (absence of issue) also in presence of invalid inputs or stressful environment conditions (event). Thus UMD is capable of expressing dependability requirements formulated according to any of these definitions. The new types of issues (e.g., response time, throughput, and accuracy failures) and external events (e.g., invalid inputs or stressful environment conditions) can be accommodated by the UMD customizable concepts to provide better guidance to stakeholders.

2.7. The UMD tool

To implement the UMD concepts, we developed a web-based tool (Basili et al., 2004a,b). The tool is organized around two main tables:

The table "scope" (see Fig. 5), which allows stakeholders to identify all the services of the system for which dependability could be of concern. For the system and each identified service, the stakeholder has to provide an identifier (name), and a brief description.

The table frame "issue" (see Figs. 6 and 7), which allows users to specify their dependability needs by defining, for the whole system or a specific service, the potential issues (failures and/or hazards), their tolerable manifestations, the possible triggering external events, and the desired reactions.

The tool produces an *MS Access* database that can support both graphical and numerical analysis, as illustrated in Section 3 (Figs. 8–10). Graphical analysis, in particular, is performed by using the visual query interface (VQI) tool, developed at the Fraunhofer Center Maryland and based on the idea of the Starfield display (Jog and Shneiderman, 1995).

3. The case study: Applying UMD

As discussed in the introduction, the NASA High Dependability Computing Project aims to increase NASA's ability to engineer highly dependable software systems. A key HDCP strategy is to accelerate adoption of new software engineering technologies by employing common-use technology evaluation testbeds representative of NASA software (HDCP, 2002). One such testbeds is the tactical separation assisted flight environment (TSAFE) (Erzberger, 2001), in a version developed by Dennis (2003), and then instrumented and adopted by University of Maryland and Fraunhofer Center Maryland as part of a larger technology evaluation effort (Asgari et al., 2004).

In order to be adopted as a testbed to evaluate technology usefulness in terms of dependability, a system must have clearly defined dependability requirements. For TSAFE, although a set of functional requirements defining the system was available (Dennis, 2003), there were no precisely stated dependability requirements. For this reason, we have decided to adopt TSAFE as a case study for UMD with the goal of performing a feasibility analysis of the UMD concept while identifying and modeling the dependability requirements for TSAFE.

In the following, we describe TSAFE, the organization of the case study, its implementation and the obtained results.

3.1. TSAFE

The Tactical Separation Assisted Flight Environment is a software system designed to aid air traffic controllers in detecting and resolving short-term conflicts between aircraft.

In the current air traffic control system, air traffic controllers maintain aircraft separation by surveilling radar data for potential conflicts and issuing clearances to pilots to alter their trajectories accordingly. Under this system, only part of the airspace capability is exploited. Exploiting the full airspace capacity requires a new approach, the Automated Airspace Concept (Erzberger, 2001), within which automated mechanisms play a primary role in maintaining aircraft separation. The role of TSAFE is to act as a reliable independent safety net from inevitable imperfections in this new model. Its aim is to detect conflicts somewhere between 2 and 7 min in the future and issue avoidance maneuvers accordingly. TSAFE plays a specific role among the existing conflict avoidance systems, which are designed to perform long-term conflict prediction (on the order of 20–40 min ahead), or to detect very short-term conflicts (only seconds away).

TSAFE provides the air traffic controller with a graphical representation of the conditions (position, planned route, forecasted synthesized route) and of the status (conformance or not conformance with the planned route) of selected flights within a defined geographical area. Fig. 4 provides a snapshot of the TSAFE display (Dennis, 2003).

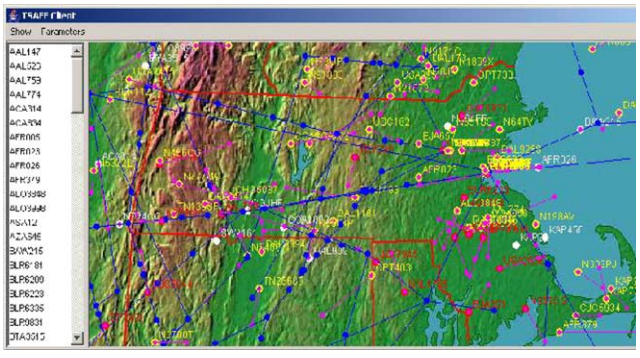

Fig. 4. The TSAFE display.

Table 1
Guidelines for measure definition

Availability level	Example values
Mission critical availability (99.99995%)	MTBF = 2.0E6 MTTR = 1 h
Very high availability (99.99%)	MTBF = 1.0E4 MTTR = 1 h
Medium/high availability (99.95%)	MTBF = 2.0E3 MTTR = 1 h
Low availability (99.9%)	MTBF = 1.0E3 MTTR = 1 h
Availability = MTBF/(MTBF + MTTR)	

3.2. Organization of the case study

For the case study, a small group of computer science researchers and students acted as stakeholders (specifically as air traffic controllers), after being given a short introduction to TSAFE and its goals. The primary goal of this initial case study was to evaluate the feasibility of the suggested approach, rather than identify the "correct" dependability requirements for TSAFE. In addition, in order to better evaluate the UMD tool capabilities, and represent real-life situations during which the stakeholders might be unfamiliar with automatic tools, all the acting stakeholders have interacted with the UMD tool through an analyst. Finally, to better support the stakeholders in quantifying the tolerable manifestation for the identified issues, the analyst was provided with the guidelines in Table 1, derived from the availability levels defined for the European Air Traffic Management System (Eurocontrol).

3.3. Requirements elicitation

UMD has been applied in two main steps, scope definition and requirements modeling.

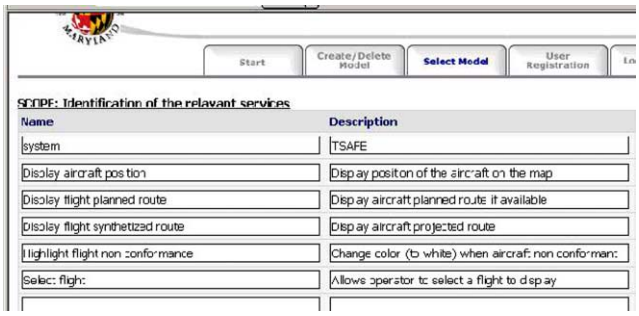

Fig. 5. The UMD tool "scope" table.

3.3.1. Scope definition

By analyzing the already available functional requirements, all the stakeholders, working together and supported by the analyst, have selected the TSAFE main services for which they believed dependability could be relevant. The identified services, as shown in the scope table (Fig. 5) are: "display aircraft position", "display aircraft planned route", "display aircraft projected route" (i.e., the route computed by TSAFE according the current aircraft status), "highlight flight conformance", which changes the color of the position dot when the aircraft is not on its planned route, and "select flight", which allows operators to select the flights to display.

3.3.2. Requirements elicitation and modeling

Each stakeholder, supported by the analysts and guided by the structure provided by the tool, has filled as many tables as necessary to define her/his dependability needs (see Figs. 6 and 7). As discussed in Section 2.3, the characterizations of the invariant UMD concepts have provided a useful guidance to the stakeholder: The stakeholder has, in fact, used the characterizations

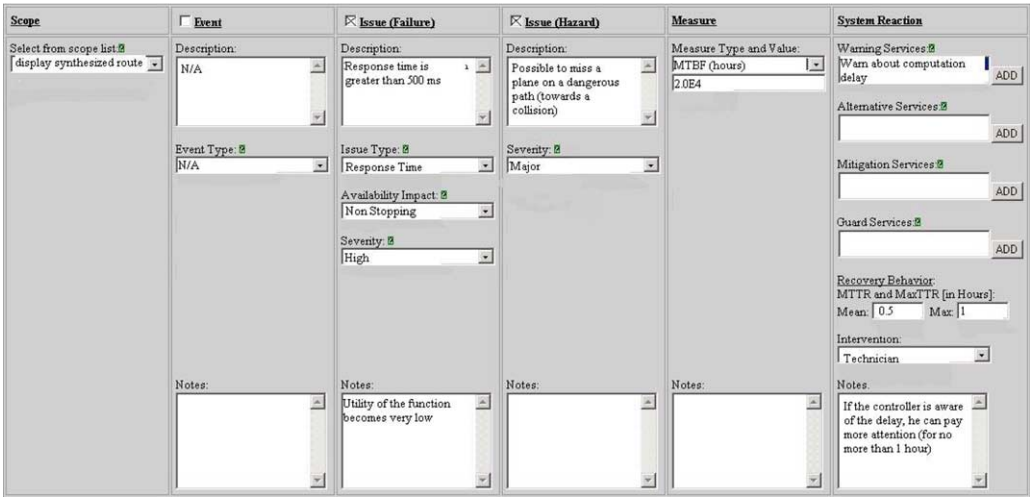

Fig. 6. UMD tool—issue not related to external event.

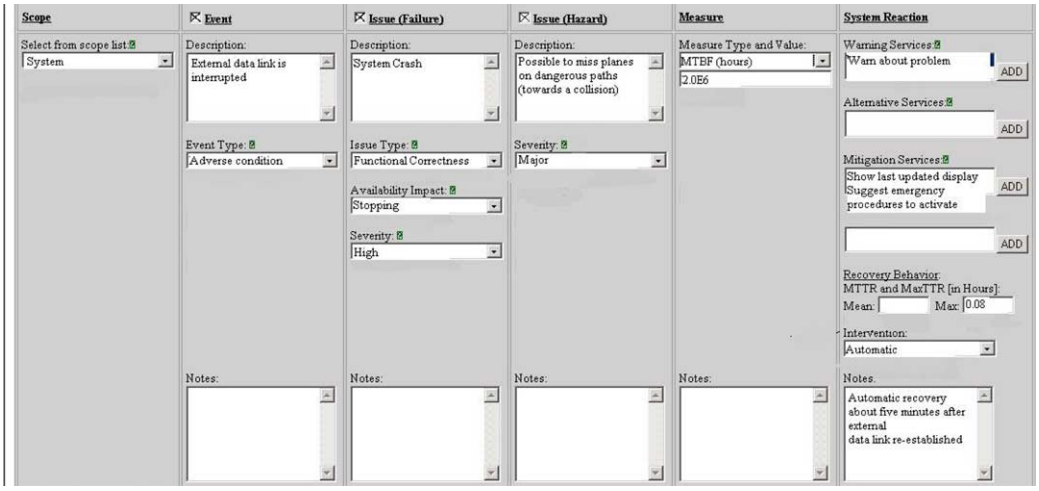

Fig. 7. UMD tool—issue related to external event.

already available (introduced by the analyst at the beginning of the project or by other stakeholders using the tool earlier), or, whenever necessary, has extended it with his/her own definitions. The characterizations

Table 2
UMD customization for TSAFE

Failure characterization

Failure types

- Functional correctness: system or service does not work or it does not implement the functional requirements
- Throughput: average or peak number of items (aircraft, routes, etc.) per unit of time dealt with by the system or service is less than expected
- Response time: response time of the system or the service greater than expected
- Peak load: max number of items handled by the system or the service is less than expected
- Accuracy: the accuracy (lateral, longitudinal, vertical) of the aircraft position or trajectory is less then expected
- Data freshness: the frequency of data updating is less than expected

Failure impact over availability

- Stopping: failure makes the system or service unfit for use
- Non-stopping: failure does not make the system or service unfit for use

Failure severity

- High severity: failure has a major impact on the utility of the system for the operator
- Low severity: failure has a minor impact on the utility of the system for the operator

Hazard characterization

Hazard types

- Catastrophic: risk of total aircraft destruction
- Severe: risk of serious damage to the aircraft, serious emergency situation, loss of human lives possible
- Major: risk of emergency situation, high stress on cockpit crew

Event characterization

- Adverse condition: any unintentional event that could have some effect on the system
- Attack: any intentional action carried out against the system

Measure characterization

Measurement models

- Mean time between failures (MTBF)
- Percentage of cases

Reaction characterization

Services types

- Warning services: warn user about the situation
- Alternative services: provide alternative ways to perform same tasks
- Mitigation services: reduce issue impact on the user
- Guard services: reduce probability of occurrence of the issue

Recovery behavior

- Mean time to recover (MTTR) max time to recover (MaxTTR)

Table 3
Example TSAFE requirements obtained with UMD

Scope	Event	Failure	Hazard	Measure	Reaction
System	Memory board fault	"Peak throughput less than 100 flights/h" Type: throughput Availability impact: non-stopping Severity: high	N/A	MTBF = 2.0E3 h	Warn about diminished performance Suggest flights to leave out MTTR = 0.5 h
System	External data link is interrupted	"System crash" Type: functional correctness Availability impact: stopping Severity: high	"Possible to miss a plane on a dangerous path" Severity: major	MTBF = 2.0E6 h	Warn about the problem Show last display Suggest emergency procedures to activate MaxTTR = 0.08 h
Display synthesized route	N/A	"Response time is greater than 500 ms Type: response time Availability impact: non-stopping Severity: high	"Possible to miss a plane on a dangerous path" Severity: major	MTBF 2.0E4 h	Warn about computation delay MTTR = 0.5 h MaxTTR = 1 h
Display synthesized route	N/A	"Accuracy: position error > Horizontal 0.25 NM Vertical 300ft" Type: accuracy Availability impact: stopping Severity: high	"Possible to miss a plane on a dangerous path" Severity: major	MTBF = 2.0E6 h	Warn about problem MaxTTR = 0.5 h

obtained for TSAFE (reconciled among the different stakeholders) are summarized in the Table 2. It is worth noting that some of the failure types and hazard types have been customized to the domain. In particular, the hazard types are based upon the hazard severity classification suggested in the RTCA standard (RTCA, 1992).

As examples of the collected data, we describe two of the tables filled by the stakeholders.

Fig. 6 illustrates an example of an issue not related to an external event. The stakeholder signals a potential failure for the service "display flight synthesized route" when the response time is greater than 500 ms. This is a response time, non-stopping, high severity failure, given the high impact on the service's utility for the operator. For the stakeholder, this failure is also a hazard (Major Hazard), given that he thinks he could miss spotting a plane on a dangerous path. This could lead to an emergency situation and possibly cause high stress on the cockpit crew, required to perform sudden escape maneuvers by the very short-term conflict avoidance systems. The stakeholder sees this failure as a very critical one, leading the analyst to suggest MTBF of 2.0E4 (between the values suggested for very high and mission critical availability in Table 1). In order to be more confident in the system, the stakeholder asks for a warning service that will advise in case the computational time becomes greater than 500 ms. In this way, he will know when to pay more attention. Finally, the stakeholder asks for the recovery to be performed within 1 h by a technician. If this failure condition lasts more than 1 h, he feels he would be unable to properly perform his duties, due to the need to maintain a higher than usual level of attention.

An example of issue related to an external event is illustrated in Fig. 7. The stakeholder is concerned that an interruption of the external data link (adverse condition) could lead to a failure affecting the whole system, a system crash. This is a functional correctness, stopping, high severity failure, given the high impact on the utility of the system. Again, the stakeholder sees also this as a hazard (major hazard), given that she will not be able to spot planes on a dangerous path. As a reaction, the stakeholder asks the system to warn about the problem (warning service), and to provide two mitigation services: First, that the system should continue displaying the data displayed prior to the interruption, so she will be able to better activate emergency procedures, knowing the planes' last positions; and, that the system should suggest what emergency procedures should be activated. Then, asked what a tolerable manifestation might be, the stakeholder feels this failure seriously impacts the capability of the system to perform its mission and suggests a high MTBF (transformed by the analysts into a MTBF of 2.0E6). Finally, she thinks that the recovery should be automatic and within approximately 5 min from the time the external data link connection is re-established.

As an example of dependability requirements collected with UMD, an extract of the requirements expressed by one of the stakeholders are described in Table 3.

3.4. Requirements analysis and refinement

As introduced in Section 2, the UMD tool can support both graphical and numerical analysis. Graphical analysis allows analysts and stakeholders to visualize how the identified issues are distributed around the different services highlighting, with different symbols colors, labels and sizes, the properties of interest (e.g., failure type, availability impact and severity; hazard severity; type of external event, etc.). Fig. 8, for example, illustrates a portion of the UMD Tool display showing how failures are distributed around the services (y-axis) according their type (x-axis). In addition, two colors are used to differentiate high severity (black) from low severity failures (gray), while labels are used to mark stopping failures.

In terms of numerical analysis, the measures expressing the tolerable manifestation for each of the identified issues have been combined to provide "aggregate values of dependability", for example, the aggregate MTBF of all the failures (or of all the failures that were also stopping failures) for each service, as shown in Fig. 9. Similarly, having the MTBF of the stopping failures, and knowing the corresponding desired recovery time, we could compute the desired availability for the whole system or a specific service, as shown in Fig. 10.

Once the stakeholder finished expressing her requirements, the analyst presented her with the results to date. Depending on her level of satisfaction, the stakeholder decided to introduce changes; while the analyst, on the other side, pointed out possible areas of risk. For example, the issue distribution in Fig. 8 (see the dotted circle)

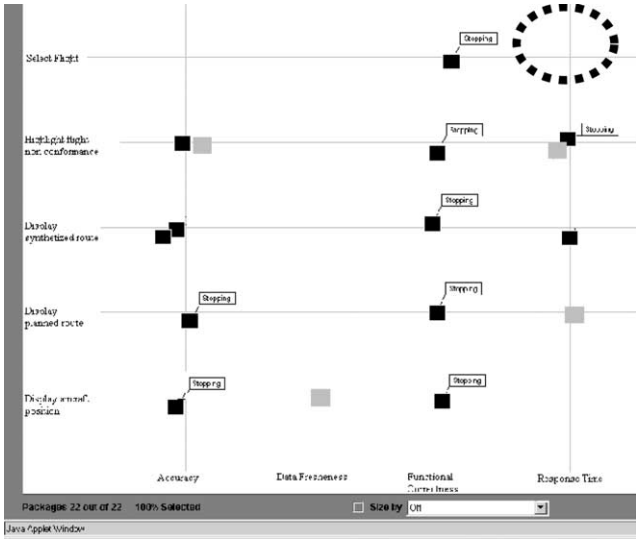

Fig. 8. TSAFE data graphical analysis.

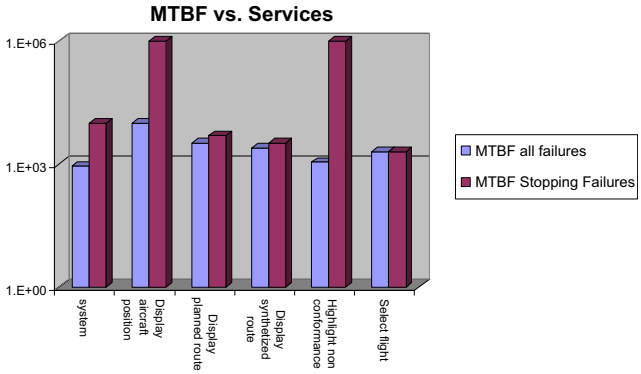

Fig. 9. MTBF for the different services.

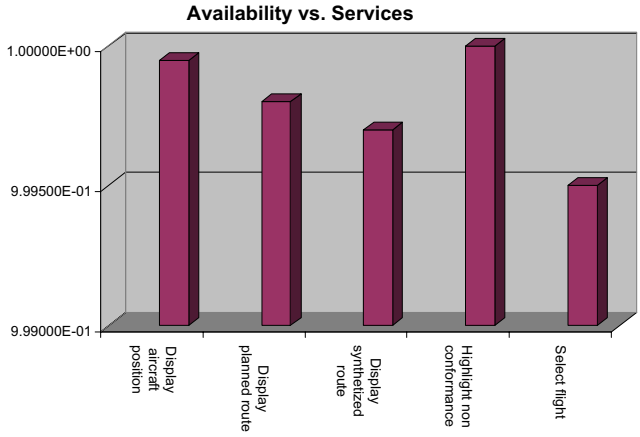

Fig. 10. Computed availability.

showed that a particular type of failure (i.e., response time) had not been taken into account for a specific service (i.e., select flight), so the analyst asked the stakeholder whether or not he confirmed such a choice.

Similarly, other times, the computed availability (Fig. 10) was insufficient for the stakeholder, so he decided to revise some of the choices he made while filling the UMD Tool tables.

The assessment led to further refinement of the already elicited requirements, and the identification of new ones. The iteration ended when both the analyst and the stakeholder felt confident about the results.

At this point, the analyst had to reconcile the needs emerging from the different stakeholders. Sometimes these could be merged, for example, when two or more stakeholders had expressed requirements concerning the same service, but identified different classes of failures. In other cases, some negotiation was necessary, for example where the stakeholders asked the system to behave in different and incompatible ways (e.g., asking the system to react to the same failure by stopping or by providing an alternative service). At the end, the analyst produced a set of agreed-upon dependability requirements representing the dependability desired for TSAFE (i.e., the TSAFE dependability model).

4. Concluding remarks

The case study results have increased our confidence in UMD's ability to provide a common modeling language that the different HDCP actors could adopt to specify, communicate and understand dependability requirements of individual systems. The concept of issue (failure and/or hazard) has appeared to be a very efficient and intuitive elicitation mechanism that stakeholders could easily use to concretely identify the most critical properties (response time, accuracy, etc.) of the system and its services. At the same time, the concepts of reaction and event have allowed stakeholders to take a constructive approach towards dependability, by suggesting active reactions to issues or indicating potential harmful external situations to take into account. The major difficulty was the selection of the measurement model to express the tolerable manifestation of an issue and the definition of the corresponding value. Here the analyst was needed. Finally, the clarity of the language has been a key factor of the elicitation and modeling process. On the one hand, the stakeholders could straightforwardly express themselves and understand each other's positions; on the other, the analyst facilitated in understanding, combining and reconciling the requirements of the different stakeholders.

UMD can be used in different ways, from improving requirements of an existing system, to eliciting dependability requirements for a new system. UMD allows stakeholders to deal with different aspects of the system that are usually dealt with in isolation rather than as part of a single framework: For example, delivered services, quality properties that are relevant for each service, failures modes and acceptable failure rates, failures severity and potential hazards, recovery time and system reaction, etc. We have found that the resulting synergy is beneficial to whole process of requirements elicitation and analysis. As recognized by others (He et al., 2004; Sommerville, 2003), gaining a more complete understanding of the system and its desired behavior in context allows stakeholders to better identify and clarify their expectations.

Besides supporting the requirements elicitation and analysis process, the application of UMD results in a set of detailed dependability requirements that supports subsequent system development activities. By precisely knowing what stakeholders consider as more or less critical (in terms, for example, of failure modes and availability of each service), project managers and system developers may more efficiently select the most appropriate strategies to attain dependability (e.g., fault prevention, removal, and tolerance techniques (Laprie, 1992; Littlewood and Stringini, 2000)). For example, at a planning level, project managers need to efficiently allocate project resources to the various competing development activities, e.g., to achieve dependability. UMD requirements allow for better-informed decisions as they permit project managers to focus project resources (e.g., testing time) on stakeholders' needs. In particular, it is worth noting that UMD can act as forerunner for value- and dependability-based software engineering approaches (Boehm et al., 2004; Boehm and Huang, 2003; Shaw, 2002).

Similarly, at the design level (Bass et al., 2003), system developers can better match requirements with available system resources (i.e., computational power, memory, etc.), both under normal and exceptional (e.g., unusually high workloads, internal faults, etc.) circumstances. They can focus on stakeholders' priorities in terms of quality properties and system functionalities to perform better tradeoff analysis. For example, in designing a fault tolerant system, they can focus on the failures most relevant to the stakeholders.

Future work will develop in three main directions. First, additional empirical assessments are necessary. We need to understand how to better employ UMD to support requirements elicitation, definition, and early verification for a system, possibly in combination with other tools and approaches, such as the Win-Win model (Boehm et al., 1998). These activities can, in fact, be performed in many different ways, depending on the system application domain, but also on the specific needs and experience of the users and developers. Second, we want to investigate the possibility of adopting UMD in the context of UML-based development approaches (Fowler, 2004). The outcome of the UMD analysis, in terms of critical non-functional properties for the system and its services, can be linked to information required to apply UML profiles for dealing with non-functional requirements during system development, such as (OMG, 2001, 2004). Finally, we want to investigate the knowledge available in the literature that could be incorporated into UMD to guide stakeholders during requirements elicitation and formulation. This could be both domain independent and domain specific and take any format, from measurement models, to classifications concerning dependability issues, such as failures, hazards, and related defects and faults (IBM Research; Mellor, 1992; RTCA, 1992), to empirical results that could help stakeholders focus on the concepts of issue, scope, measure, event, and reaction, enabling them to more effectively transform their dependability needs into precise requirements.

Acknowledgments

The authors wish to acknowledge support from the NASA High Dependability Computing Project under cooperative agreement NCC-2-1298.

The authors wish to thank the researchers on the HDCP project team for their insights and suggestions and Jennifer Dix for proof reading this paper.

References

Asgari, S., Basili, V., Costa, P., Donzelli, P., Hochstein, L., Lindvall, M., Rus, I., Shull, F., Tvedt, R., Zelkowitz, M., 2004. Empirical-based estimation of the effect on software dependability of a technique for architecture conformance verification. In: Proceedings of the ICSE 2004 Workshop on Architecting Dependable Systems, Edinburgh, UK.

Avizienis, A., Laprie, J.C., Randell, B., 2001. Fundamental concepts of dependability. Research Report N01145, LAAS-CNRS, France.

Basili, V., Donzelli, P., Asgari, S., 2004a. The unified model of dependability: Putting dependability in context. IEEE Software 21 (3), 19–25.

Basili, V., Donzelli, P., Asgari, S., 2004b. Modeling dependability-the unified model of dependability. Technical Report CS-TR-46-01, University of Maryland College Park, MD, US.

Bass, L., Clements, P., Kazman, R., 2003. Software Architecture in Practice, second ed. The SEI Series in Software Engineering. Addison Wesley.

Boehm, B., Huang, L., 2003. Value-based software engineering: A case study. IEEE Computer 36 (3), 34–41.

Boehm, B., Egyed, A., Kwan, J., Shah, A., Madachy, R., 1998. Using the win-win spiral model: A case study. IEEE Computer 31 (7), 33–44.

Boehm, B., Huang, L., Jain, A., Madachy, R., 2003. The nature of information system dependability—a stakeholder/value approach. Technical Report, University of Southern California, CA, US.

Boehm, B., Huang, L., Jain, A., Madachy, R., 2004. The ROI of software dependability: The iDave model. IEEE Software 21 (3), 54–61.

Chung, L., Nixon, B., Yu, E., Mylopoulos, J., 2000. Non Functional Requirements in Software Engineering. Kluwer Academic Publisher.

Dennis, G., 2003. TSAFE: Building a trusted computing base for air traffic control software. MSc thesis, MIT, Boston, MA, US.

Erzberger, H., 2001. The automated airspace concept. In: Proceedings of the 4th USA/Europe Air Traffic management R&D Seminar, Santa Fe, New Mexico, US.

Eurocontrol—European Organization for Safety of Air Navigation. Available from: <http://www.eurocontrol.be/home.html>.

European Union, 2002. European Initiative on Dependability: Towards Dependable Information, Report, Bruxelles. Available from: <http://deppy.jrc.it>.

Fowler, M., 2004. UML Distilled, third ed. Addison-Wesley.

HDCP—High Dependability Computing Project, 2002. Available from: <http://hdcp.org>.

He, J., Hiltunen, M., Schlichting, R., 2004. Customizing dependability attributes for mobile service platforms. In: Proceedings of the International Conference on Dependable Systems and Networks, Florence, Italy.

IBM Research. Orthogonal Defect Classification. Available from: <www.research.ibm.com/softeng/ODC>.

IFIP—International Federation for Information Processing WG-10.4. Available from: <www.dependability.org>.

Jog, N., Shneiderman, B., 1995. Starfield information visualization with interactive smooth zooming. In: Proceedings of the IFIP 2.6 Visual Databases Systems, Lausanne, Switzerland.

Knight, J., 2002. Safety critical systems: Challenges and directions. In: Proceedings of the International Conference on Software Engineering, Orlando, FL, US.

Laprie, J.C., 1992. Dependability: Basic Concepts and Terminology, Dependable Computing and Fault Tolerance. Springer-Verlag.

Littlewood, B., Stringini, L., 2000. Software reliability and dependability: A roadmap. In: Proceedings of the ACM Future of Software Engineering conference, Limerick, Ireland.

Melhart, B., White, S., 2000. Issues in defining, analyzing, refining, and specifying system dependability requirements. In: Proceedings of the 7th IEEE Conference on the Engineering of Computer Based Systems, Lund, Sweden.

Mellor, P., 1992. Failures, faults and changes in dependability measurement, 1992. Information and Software Technology 34 (10), 640–654.

Moteff, C., Copeland, J., Fisher, S., 2003. Critical infrastructures: What makes an infrastructure critical? Technical Report RL31556, The Library of Congress, DC, US.

OMG—Object management Group, 2001. UML Profile for Schedulability, Performance, and Time Specification. Available from: <www.omg.org>.

OMG—Object Management Group, 2004. UML Profile for Modeling Quality of Services and Fault Tolerant Characteristics and Mechanisms. Available from: <www.omg.org>.

Randel, B., 1998. Dependability, a unifying concept. In: Proceedings of Computer Security, Dependability and Assurance: from Needs to Solutions, York, UK and Williamsburg, VA, USA.

RTCA—Radio Technical Commission for Aeronautics, 1992. Software Considerations in Airborne Systems and Equipment Certification, DO-178B Standard.

Shaw, M., 2002. Everyday dependability for everyday needs. In: Supplemental Proceedings of the 13th International Symposium on Software Reliability Engineering, Maryland, US.

Sommerville, I., 2003. An integrated approach to dependability requirements engineering. In: Proceedings of the 11th Safety-Critical Systems Symposium, Bristol, UK.

Sommerville, I., 2004. Software Engineering, seventh ed. Addison-Wesley, UK.

US, 2003. National Strategy to Secure Cyberspace, Report, Washington DC. Available from: <www.whitehouse.gov/pcipb>.

US Department of Defense, 1985. Trusted Computer System Evaluation Criteria (Orange Book), DOD 5200.28-STD.

US Department of Defense, 2000. Standard practice for system safety, MIL-STD-882D.

Walkerdine, J., Melville, L., Sommerville, I., 2004. Dependability within peer-to-peer systems. In: proceedings of the ICSE 2004 Workshop on Architecting Dependable Systems, Edinburgh, UK.

Weinstock, C.B., Goodenough, J., Hudak, J., 2004. Dependability cases. SEI Technical Report CMU/SEI-2004-TN-016, PA, US.

Wenger, A., Metzger, J., Dunn, M., 2004. International CIIP Handbook, Swiss Federal Institute of Technology Zurich. Available from: <www.isn.ethz.ch/crn/_docs/CIIP_Handbook_2004_web.pdf>.