# Toward User Oriented Dependability Standard for Future Embedded Systems

Yoshiki Kinoshita
*yoshiki@m.aist.go.jp*

Yutaka Matsuno*
*yutaka.matsuno@aist.go.jp*

Hiroki Takamura
*takamura-h@aist.go.jp*

Makoto Takeyama
*makoto.takeyama@aist.go.jp*

Research Center for Verification and Semantics (CVS),†
National Institute of Advanced Industrial Science and Technology (AIST).

JST, CREST. ‡

## Abstract

*We outline our new project* User Oriented Dependability*, a project within the research program "Dependable embedded operating systems for practical use (DEOS)" in Core Research for Evolutional Science and Technology (CREST) program of Japan Science and Technology Agency (JST). Our first goal is to establish a concept of dependability in the era of extremely complex networks of evolving and open embedded systems. The next goal is to develop it into an international standard, together with guidelines for assessors, developers, and users. We intend that the standard will advance the dependability of future society where (1) users will have objective criteria for the dependability of the system they work with and (2) developers will be able to provide the dependability of their products as an objective added value backed by the standard. We aim to add new aspects of "user oriented dependability" to the concept of dependability. One of the aspects is that users of dependable systems actually understand its services and can cope with various expected and unexpected difficulties that inevitably occur during the whole life cycle of the systems. We outline some basic ideas, current activities, and plans toward the goals.*

## 1. Introduction

We outline our new project "User Oriented Dependability", a project within the research area "Dependable embedded operating systems for practical use (DEOS)" in Core Research for Evolutional Science and Technology (CREST) program of Japan Science and Technology Agency (JST).

Systems are everywhere in daily life: home electronics, cars, mobile phone services via Internet, closed-circuit television, etc. Such systems around us have three significant properties:

- Systems evolve. Systems are modified, updated, and replaced all the time during their use phase. This happens whether users and developers like it or not, in order for the systems to continue to function in their changing environments.

- Systems are connected. Computers embedded in various systems are interacting with each other in unprecedented scales. A car engine controller could be communicating with a far away server via an automated cruise system and a navigation system. Users themselves may form links between unexpected combinations of systems.

- Systems are open. Components of open systems dynamically change via interaction with the external environments; their boundaries are ambiguous; and they can provide services by varieties of different behaviors (this property is called equifinality[16]. ) Tokoro [15] pointed out that a new form of science for open system is needed that integrates and fuses analysis, synthesis, and management.

Evolvability allows systems to be flexible to user requirements and the rapidly changing environments, and connected systems provide unified services that can not be provided by disconnected systems.

*Corresponding author
†Address: 5th floor, Mitsui Sumitomo Kaijo Senri Bldg., 1-2-14 Shin-Senri Nishi, Toyonaka, Osaka, 560-0083, Japan
‡Address: 5, Sanbancho, Chiyoda-ku, Tokyo, 102-0075, Japan

978-0-7695-3572-2/09 $25.00 © 2009 IEEE
DOI 10.1109/STFSSD.2009.37

However, these characteristics make recent systems extremely complex, often to the extent that we can no longer fully comprehend them. We believe that a new concept of dependability must be developed for the era of extremely complex networks of evolving open embedded systems. The existing dependability concepts are not enough nor adequate because they tend to be contemplated from developers' standpoints. They mainly focus on "how" some dependability of a system should be achieved by developers. For example, although the paper [1] considers a wide variety of threats to dependability, most of the means to attain dependability discussed there are accessible only to developers. The paper does give certain emphasis on the issues of users and operators. However, they mostly appear as parts of use environments, as something attached to the central entities that is the systems, and as conceived by developers. It is natural, for developers, that a dependability case (the argument for why a system can be depended on) to be based on the system's construction and the assumptions / requirements on its environments. It is then natural that dependability cases to become ever more complex and interconnected as the complexities of systems and environments grow. Current approaches seem to give even less help for users to understand this complexity than they do for developers to build up the complexity.

We believe that users should be treated as the central actors: it is their desire that prompts the development of complex and evolving systems, and it is they who must deal with the world full of such systems. We believe that new reasoning principles (or at least presentation methods) should be developed for the construction of dependability cases that are to convince users. After all, we as users do demand, and more or less obtain, the ease of use familiar from simple systems of the past, from today's complex systems; why should not we do the same for ease in trusting the systems?

Our first goal is to establish a concept of dependability for future society, where systems are evolving, networked, and open. We aim to add new aspects of "user oriented dependability" to the concept of dependability. There are several attempts on developing the concept of dependability focusing on users [4, 2, 12]. However, such attempts are limited because they only considered cases when users are end users in specific environments. The next goal is to set up a standard based upon the concept, and to develop it into an international standard, together with two guidelines accompanying the standard; one for the conformity assessment and the other on how to manage system life cycle (design, development, operation, maintenance, and disposal) according to the standard. We aim to develop a standard that based on users' viewpoint, incorporating the aspects of previous concepts. We also believe that such a standard is important in comparing different concepts of dependability.

Figure 1 shows the roles of assessors, developers, and users. We intend that our standard will advance the dependability of future society where (1) users will have objective criteria for the dependability of the system they work with and (2) developers will be able to provide the dependability of their products as an objective added value backed by the standard.

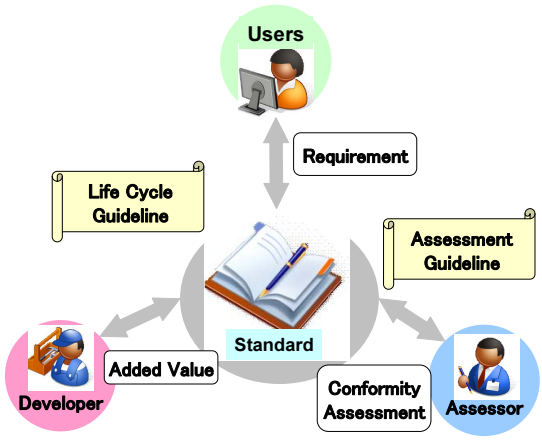


**Figure 1. Roles of User, Assessor, and Developer**

The rest of the paper is structured as follows. In Section 2, we consider what dependability properties are needed for future embedded systems by a few examples, including our main target: the DEOS architecture. Section 3 summarizes our current activities and plans.

## 2. Basic Ideas

The original definition of dependability is the ability to deliver service that can justifiably be trusted [1]. Several other definitions also have been proposed. An alternative given by the authors of [1] is that "the ability to avoid service failures that are more frequent and more severe than is acceptable." Their definition assumes the inevitability of failures and provides a criterion for deciding whether or not, in spite of failures, a system is still to be regarded as dependable. Tokoro[14] states that dependability is "a property that provides with users their expected services as safely and continuously".

However, in these definitions, users are not given an active role. The definition of [1] has little emphasis on the fact that it is the (human) users who decide a system they use can be trusted; the users in [1] are just some other systems in the environment, and do not have the character of active, central actors that we mentioned earlier. We believe that users should (be able to) decide if a system is depend-

able, and demand more than avoiding failures. A possible formulation is:

> Users of dependable systems actually understand its services and can cope with various expected and unexpected difficulties that inevitably occur during the whole life cycle of the systems.

We emphasise the following:

- **User oriented dependability.** Users are the central actors for coping with difficulties. The systems and the developers, maintainers, etc are there to help.
- **Look for unexpected.** Difficulties are mostly unexpected, specially in case of evolving, networked, and open systems. The problem of unexpected events has been widely recognized, for example in [10]. "Look for unexpected" is a phrase by Jones [11]. That unexpected things happen is a basic premise in Tokoro [14].
- **Continuity** It is usual to say dependability of a system should be sustained continuously throughout its lifecycle. Continuity is particularly crucial for infrastructure systems such as operating systems that must support changing applications in changing environments. Continuity is one of the keywords emphasized in [14].

### 2.1 Aspects of "User oriented dependability"

We list certain aspects of "user oriented dependability" using examples.

- Electronic Devices: TVs and mobile phones. TVs and mobile phones are the typical examples of today's networked, evolving, and open systems.

  TV: Almost all of us can use its services: turn on TV, change channels, watch TV show, change volume, turn off TV, etc. If it does not work, the user would consult to some electronic shop for repair or buy new one. It may sound trivial, but it means that the user can deal with a failing TV during its whole use-phase.

  How about new digital TVs which are connected to the Internet and hence are evolving and open? For example, The US Federal Communications Commission (FCC) is announcing that, in the US, after February 17, 2009, full-power TV stations will broadcast only in digital. FCC says "the switch will free up the airwaves for police, fire, and emergency rescue communications, ... and allow for advanced wireless services."

  Will a user be able to understand those advanced services including their limitations? When she comes to rely on those services in her daily life, will she be able to cope with service failures?

  Mobile phones: Recent mobile phones have many kinds of features: calling, e-mail, web shopping (such as Amazon.com, which is addressed below), music, digital TV, and electronic money. As more and more people get access to those advanced features, more and more important services are done through those features, making things ever more convenient. However, are those services made more dependable? How about some elderly people and others who have difficulty in using them? Can we cope when there is a network failure at a mobile carrier?

- Amazon.com. Amazon.com is one of the largest e-commerce operation in the world. Many users trust Amazon.com: the users send their personal information such as their credit card number via Internet. We consider that some of the reasons why they are willing to do so are: Amazon.com is easer to use than other web shopping system; the quality of service is high: ordered goods can be reached within a few days; and the fact that many other users use Amazon.com make users feel safe to use Amazon.com. Amazon.com's technology of reliability is shown in [3]. Even though most of the users do not know details of the technology, they trust Amazon.com and continue to use it. What properties of Amazon.com, besides the above reasons that are somewhat fuzzy, make users to trust it? Do we not need more rigorous, verifiable criteria of dependability to place trust in some services?

- Systems with human components. Humans do not appear only as the users, outside of the boundary of a system under consideration. Indeed, most systems contain humans as components inside the system boundary to provide its services. So our "user oriented dependability" should give some guidance for the users (outside the system boundary) on how to cope with possibly failing human components within the system boundary. Considering that a system and its user form a larger system (making him a component of the larger system), we also should think about how he should behave when he himself failed.

### 2.2 DEOS architecture

Our main target is the DEOS architecture[1] which is being developed in the research program "Dependable embedded operating systems for practical use (DEOS)" in Core Research for Evolutional Science and Technology (CREST) program of Japan Science and Technology Agency (JST).

[1] http://www.dependable-os.net/index-e.html

DEOS is based on Linux, and its main concept is described as follows [6].

> On the Unix-like operating systems, such as Linux, Free BSD, and Solaris, developers are able to extend OS kernel features by dynamically-loadable kernel modules. However, it is not an easy task because it is not easy to understand, not portable, not extensible and not reliable. To solve these, the DEOS architecture introduce a a new layer between a kernel and kernel extensions called *P-Bus*. P-Bus provides abstracted, well-defined, and well-exported interfaces for the extensions. Kernel extensions on the P-Bus are named *P-Components*. P-Components are required to use P-Bus API in order to interact with operating system kernels.

Some of P-Components being developed in the DEOS project are: power-saving parallel distributed computing, program verification by typed assembly language, high performance home electronics, micro ubiquitous technology, highly secure OS, security weaver, and real time parallel distributed computing. DEOS is designed as an OS for various embedded systems such as mobile phones, office and home electronics, robot operation system, and network server. Currently members of the "DEOS Core team" are discussing how such P-Components can convince the users (developers of mobile phone, office and home electronics, ...) that the DEOS architecture is valuable in their development. The requirements of developers may differ in their target products. For example, real time execution is crucial in robot operation systems, but may not for the others.

### 2.3 Sustaining DEOS dependability

Our main concern on DEOS is how to sustain its dependability during the whole life cycle. However, the life cycles of evolving, networked, and open systems are much more complicated than the model given in [1] consisting of just development and use phase.

DEOS will be continually updated due to bug fixes and requirement changes, so the use and development phases of different versions of DEOS will overlap. Different DEOS versions will coexist in systems. The life-cyle of DEOS must be considered in those of application systems based on DEOS.

Compatibility among different DEOS versions is an expected problem, and there will be more serious problems in sustaining dependability through such a complex life cycle. We believe that some novel management process should be established.

"The DEOS promotion board" represents the targeted users of DEOS, consisting of major electronic companies. Those (middle-)users are at the same time developers of application systems using DEOS, which themselves have their end-users. Therefore, our user oriented dependability is related to their activities in dual senses (Figure 2). As in this case, systems around us form developer-user chains. At each level, we consider its dependability from users' viewpoint so that the whole systems are dependable.

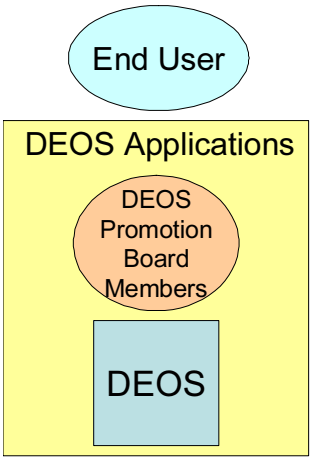


**Figure 2. Users of DEOS and Developers using DEOS are both DEOS promotion board**

### 2.4 What is dependability?

Let us summarize our discussion in above examples. We consider the following three points are crucial.

- Dependability is not only about conventional attributes such as reliability. It may include comfortableness, resource, energy consciousness [14], supportability, satisfiability, and sustainability.
- Dependability of a system should be sustained during the whole life cycle of the system. If a failure occurs, then the system should be recovered as soon as possible, and the failure should never occur again [14].
- Dependability should be a notion dependent on the class of users under considerations and we must be clear about that class.

## 3. Our Activities

Our current and future activities include the following.

### 3.1 Conceptualizing "user oriented dependability"

IEC TC 56 dependability committee of Japan is also studying new concept related to dependability. Its concept includes integrity, dependability (in the meaning of IEC 60300 [5], which is different from that of [1]), operability, and sustainability. Together with existing concepts such as in [1], we study the relations among our concept and others, and combine them as a new dependability concept. Such a new concept will relate to existing concepts such as accountability, traceability, etc.

### 3.2 Analysis of life-cycle processes

A good examples of threats at each phase in the life cycles of systems are given by the DEOS promotion board. The DEOS promotion board summarized a matrix of threats indexed by kinds of threats and life cycle phases in which the threats occur. The phases are: specification, design, implementation and unit test, integration test, distribution, operation, maintenance and update, and disposal and recycle. The threats are: in environments (operation environment, development environment, etc), in hardware, in human mistakes, and by human attacks. A part of the matrix is shown in Table 1.

Based on the threats matrix, as a step toward (whole) life cycle guidelines for systems, we list up threats which may occur at each development phase defined in IEC 12207 :2008 [8] and IEC 15288:2008 [9], and consider ways of coping with them.

Currently we are mapping each threat in the threats matrix to the software processes such as IEC 12207 or IEC 15288, which are to prevent or mitigate those threats. For example, using "Software Life Cycle Processes-Japan Common Frame 2007"[13] based on IEC 12207:1995[7],

- For the threat "misunderstanding of specification", we consider user documents(system operating manual, management manual, etc). In process 1.7.4.1 of [13], it is written that according to user documents, system should be operated under the environment intended by users.

- Process 1.6.12 of [13] corresponds to the threat "installation error." In 1.6.12.1 of [13], it is said that careful plan documents for software installation should be written. Installation in use phases is done by the maintainer according to 1.7 of [13].

The mapping will be used to critically analyse, or "debug" the existing life cycle processes, which are meant to prevent / reduce / mitigate those threats but are less than perfectly successful. It may be that the "specification" of a process is lacking, e.g., not describing a way forward when systems specification cannot be determined before implementation starts. It may be that the "program" of a process (tasks / activities) is not adequate, e.g., producing outputs which turn out to be insufficient at a later stage. In any case we need to clarify various concepts appearing in those life cycle processes. We intend to do so by analyzing dependencies among them, e.g., def-use relationship. Findings from this critical analysis will be used in PDCA-cycles on processes themselves as suggested by Tokoro [14], as well as making the standard we propose to be a more effective one.

The next steps of our current activities on the threat matrix by the DEOS promotion board are: developing guidelines and using them for the DEOS distributions and the DEOS with applications; analyzing the effects of services by applications on DEOS, to the dependability of the whole system; and developing methodologies for documentations at all life cycle phases.

### 3.3 Community formation

For a standard to become widely accepted, it is important to form a community around it involving a variety of relevant groups. We have started or plan to start:

- Visits to leading sites for dependability and standardization, both in universities and industry, for face-to-face discussions on ideas.

- Participating in relevant committees and groups: Our project members are currently involved in various capacities in IEC/TC56 (Dependability), ISO/TC199 (Safety of Machinery), OIML/TC5/SC2(International Organization for Legal Metrology, TC5 electronic instruments and software, SC2 software), and REAJ LCC study group (Reliability Engineering Associations of Japan, Life Cycle Costing). ISO/IEC JTC1/SC7 (Software and System Engineering) is another relevant committee, in which we plan to participate shortly.

We intend to develop our ideas through diverse interactions with others and feed back to communities.

## 4. Conclusion

In this paper we have introduced our new project, User Oriented Dependability. We hope that our standard will benefit and be flexible to all of assessors, developers, and users in the rapidly evolving, networked, and open world.

**Acknowledgements.** We thank other members of our team: Noriaki Izumi, Mitsuo Kishimoto, Satoshi Matsuoka, Daichi Mizuguchi, Toshinori Takai, Hiroshi Watanabe, and Yoji Yamada for their insightful comments. We thank

**Table 1. A Part of Threats Matrix by the DEOS Promotion Board**

| | Environment | Hardware | Human Mistakes |
|---|---|---|---|
| Implementation and UT | bugs in development tools, insufficient functionality of development environment, inconsistent versions of development environment, inadequate education system, insufficient verification of software tools, too frequent design changes | mismatches and delays in development schedule of target hardware, low quality yield, mismatches of specifications, HW bugs (that can / can not be fixed during the phase) | coding error, insufficient code review, algorithm errors, wrong choices of software libraries, integration mistakes, version mismatches, insufficient unit tests, copyright / patent infringement |
| Integration Test | bugs in testing tools, bugs in testing environment, too short testing time, too frequent implementation changes | insufficient implementation of testing functionality, remaining bugs in hardware | overlooked test items (insufficient test cases, mistakes in test execution), insufficient checking of test result, defective test items, insufficient testing review, misconfiguration of testing environment |
| Operation | aging, temperature environment, humidity environment, errors propagated from other systems' faults, system failure due to unexpected input data, input overload, impact shock, power failure(transient, fluctuation, outage, . . . | hardware degradation (mechanical, chemical, solid-state), contact failure(connectors, switches), power consumption too high, electromagnetic noise, heating | misunderstanding of specification, user errors due to carelessly designed user interface, operation errors (wrong function choice, wrong data input/choice), installation errors, configuration errors, data migration errors |

DEOS promotion board members for allowing us to introduce their threats matrix and Makoto Yashiro for carefully reading a draft of the paper.

## References

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.*, 1(1):11–33, January 2004.

[2] J. Bohn. *User Centric Dependability*. PhD thesis, ETH ZURICH, 2006.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *SOSP*, pages 205–220, 2007.

[4] G. Dewsbury, I. Sommerville, K. Clarke, and M. Rouncefield. A dependability model for domestic systems. In *SAFECOMP*, pages 103–115, 2003.

[5] IEC. *IEC 60300 series – Dependability management*.

[6] Y. Ishikawa. P-bus specification, 2008. http://www.il.is.s.u-tokyo.ac.jp/deos/pbus/spec/.

[7] ISO/IEC-IEEE. *ISO/IEC 12207:1995 Systems and software engineering – Software life cycle processes*. 1995.

[8] ISO/IEC-IEEE. *ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes*. 2008.

[9] ISO/IEC-IEEE. *ISO/IEC 15288:2008 Systems and software engineering – System life cycle processes*. 2008.

[10] D. Jackson, M. Thomas, and L. I. Millett. *Software for Dependable Systems Sufficient Evidence?* The National Academies Press, Washington D.C., 2007.

[11] C. Jones. Which system are we looking at: a dependability perspective, 2008. Presented at UK-Japan Symposium on Privacy and Security in the Information Society, British Embassy Tokyo, Japan (http://www.ukjapan2008.jp/events/20081111_000279e.html).

[12] B. Latronico, C. Martin, and P. Koopman. Analyzing dependability of embedded systems from the user perspective. In *Workshop on Reliability in Embedded Systems (in conjunction with SRDS)*, 2001.

[13] Software Engineering Center, Information-Technology Promotion Agency, Japan. *Software Life Cycle Processes-Japan Common Frame 2007(in Japanese)*. Ohmsha, 2007.

[14] M. Tokoro. Fundamental concept of dependability, November 2008. Presented at Symposium on System Verification in Tsukuba University, Japan (http://unit.aist.go.jp/cvs/symposium/SSV08/program2008.html).

[15] M. Tokoro. Towards open-systems science, August 2008. Presented at the Sony CSL 20th Anniversary Symposium in Shinagawa, Japan (http://www.sonycsl.co.jp/topics/2008/08/csl-20.html).

[16] L. von Bertalanffy. *General Systems Theory*. George Braziller, New York, 1968.