CURRENT RESEARCH ACTIVITIES ON DEPENDABLE COMPUTING AND OTHER DEPENDABILITY ISSUES IN JAPAN

Yoshihiro Tohma[1] and Masao Mukaidono[2]
[1]Tokyo Denki University, tohma@sie.dendai.ac.jp; [2]Meiji University, masao@cs.meiji.ac.jp

Abstract: Current research activities on dependable computing and other related dependability issues in Japan are reviewed. When considering the dependable computing, an emphasis is put on architectural aspects of computing systems, though the dependable computing is not limited to them, but ranges very broadly. Not only technical issues but also some organizational activities are also touched.

Key words: fault tolerance; mobile computing; COTS; Internet; grid computing; feature interaction; fail-safe; organizational activities.

1. INTRODUCTION

Taking this opportunity of Topical Day for commemorating Prof. Al Avižienis's outstanding contributions to the advancement of fault tolerance and dependable computing, it is our pleasure and honor to present a paper, reviewing research activities on dependable computing and other dependability issues currently conducted in Japan. However, this paper is not intended to make a comprehensive survey, but to introduce some research outcomes simply based on the authors' view.

When looking at research activities, we first focused our attention to architectural aspects of dependable computing systems with new features, and further argued some extension of the application of dependability concept to other technical worlds.

2. NEW PARADIGMS OF DEPENDABLE COMPUTING

These days, every thing such as the configuration of computing systems, the requirement to them, and the computing paradigm are changing rapidly. Stand-alone computing systems make no longer any sense, but are almost networked. The scale of a computing system and its clients is ever exploding concurrently with its non-stop provision of services. It is required to provide different services of not only scientific computation but also support or assistance to the daily life of individuals. Mobile and/or ubiquitous computing are new demands. The way of computation is changing. Computing will be shared with servers distributed over a network(s) in a form of, say, grids.

Facing these trends, dependable computing is becoming more significant in different ways.

2.1 Fault tolerance in mobile computing

In mobile computing environment, computing unit in such equipments as cellular phone, PDA, etc. must be light and small for portable purposes. Therefore, LSI's in such equipments are produced by sub-micron fabrication technologies. This means that those computing units are becoming very vulnerable to cosmic ray and/or artificially made radiation such as alpha particle and prone to suffer transient errors. Since continuous and real-time services are mandatory in mobile computing, it is required and becomes more important to implement countermeasures against transient errors, which can recover the computing units quickly.

T. Sato proposed to implement the fault detection and recovery mechanism in superscalar processors (Sato 2003) with the at most 8-way dynamic scheduling, focusing on transient errors in the logic and arithmetic units in the processor. Other parts can be protected effectively by means of ECC and/or parity check. The essence of error detection is simply the duplicated execution of an instruction and the comparison between the both results.

As shown in Fig. 1, instructions are first stored in the register update unit (RUU), of which primary role is to carry out the dynamic scheduling, and then dispatched to functional units. When an instruction in RUU is committed (with the necessary control and data ready), it is again dispatched. Its execution result is compared to that of the first execution of the instruction.

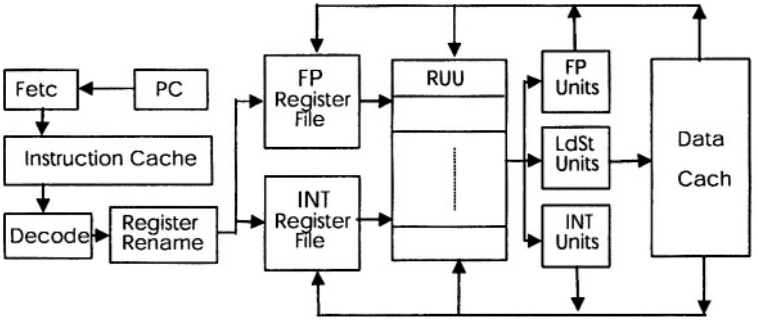

Figure 1. Error detection.

When the mismatch happens, the recovery by hardware (instead of relying on OS) is activated transparently. Regarding the occurrence of an error to be a misspeculation of data, the transparent hardware recovery here uses the existing mechanism to reissue an instruction upon the misspeculation of data.

What we are mostly concerned with is the performance degradation caused by the introduction of fault tolerance. The percentage increase of execution cycles in the error-free operation was measured for various benchmarks of SPEC2000 and MediaBench suites. Thanks to the superscalaring (Mendelson 2000), the overheads in SPEC200 and MediaBench are 44.2% and 44.7%, respectively, in average, even though every instruction is executed twice. The more detailed investigation to reduce the overhead revealed that earlier speculative update of branch prediction table at the stage of instruction decode together with the elimination of redundant memory access is effective. Finally, the overhead was reduced to about 30% in average by incorporating the above two techniques. He notes that the limitation of hardware resources greatly influences the performance degradation and therefore, it could be lessened further, if microprocessors would have sufficient hardware resources.

2.2 Use of COTS for dependable Internet services

As Internet extends not only to computer professionals but also to common people, various services which range from purely engineering computations to daily supports for individuals such as multimedia delivery, online entertainments, VoIP, e-commerce, e-government, and etc. are carried outd on it. Thus, Internet has become an important social infrastructure, and the importance of dependable computing on Internet is recognized even by common people.

On the other hand, the cost to implement dependability measure for Internet should not be much, because they should be incorporated widely and commonly. Traditional dependable computing systems in critical applications use proprietary hardware such as voting and fault isolation circuitries with proprietary OS. They are main sources of the cost. Instead of using such proprietary components, Mishima and Akaike proposed ways to implement fault tolerance measure (Mishima and Akaike 2003), using COTS components of commodity hardware (computers and networks) and software (OS and applications), which are not modified, nor re-compiled, and nor vendor-specific.

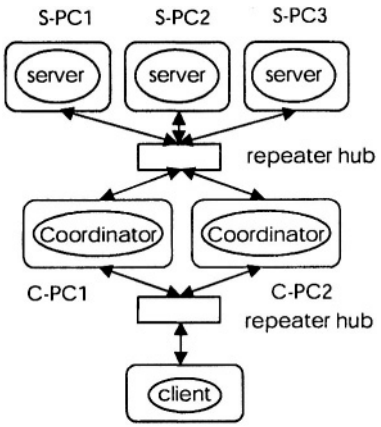

Figure 2. Fault tolerance measure for servers.

Although Internet services are carried out in the coordinated relationship of servers and clients, the malfunction of servers is much more serious than that of clients. Therefore, they simply considered ways to realize dependable servers. Since many services on Internet are real-time operations, the fault tolerance of servers based on the primary-backup is not applicable. Therefore, the fundamental idea is to use the active replication of servers as shown in Fig. 2, which needs less time to make the recovery than that of the primary-backup. Three servers on different PC's (denoted S-PC's) are employed to simply perform the TMR operation. However, the key issue here is to make the fault tolerance measure as much transparent as possible, because in the use of COTS components, the modifications of both hardware and software components in servers and clients should be kept unnecessary. To realize the transparency, a control function called *Coordinator* is inserted between a client and triplicate servers. In addition to IP address of Coordinator itself "CDR", it has another IP address "IPVD" (IP address as Virtual Server) by which the client views the Coordinator as a single virtual server. The communication between the Coordinator and the clients are

carried out by referring to the client's IP address "C" and "IPVD". Similarly, the Coordinator communicates to each of the replicate servers, using "CDR" and the server's IP address "S" in addition to "C", as shown in Fig. 3.

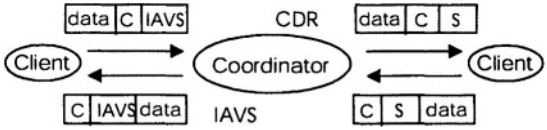

Figure 3. Packet flow control.

When a client communicates to the servers, the Coordinator must manage the loose synchronization among the replicated servers and make the adjudication. To reduce the response time to the client, it responds quickly to the client as soon as the first and the second responses from the replicated servers agree.

In order to tolerate the malfunction of Coordinator itself, the primary-backup and the active replication of Coordinators were considered. The experiments of the performance evaluation show that the former is better than the latter in terms of the round trip time. It is speculated that the active replication of Coordinators causes more congestion of packets, resulting in the performance degradation.

2.3 Fault tolerance in new computation paradigm

Today, computers exist ubiquitously, and are connected to each other through a network(s). However, they do not necessarily run always. Therefore, such an idea is a natural consequence that computing power distributed over a network should be shared by different users. They should not necessarily reside in a corporation but across enterprises. In a way, computers make a form of virtual grid over a network and they perform services in the coordination. This computation paradigm is called *grid computing*.

A computer over a network may participate in different grids. Further, it decides for itself whether it participates in a coordinated service or not. In this sense, the grid computing is a paradigm of *distributive* and *autonomous* computing.

Another incentive to such distributive and autonomous cbmputing is the continuous expansion of a computing system over network(s). The number of computing facilities on a network ever increases. Further, the territory of a computing system over a network is becoming more and more boundary-less.

The configuration and scale of a computing system over network(s) changes dynamically. In such circumstances, it is hard to manage whole computing system by a single centralized mechanism. We should rely on the autonomous computing distributed over the network.

A typical example of computational model of grid computing is as shown in Fig. 4 (Foster 2002), where computers of two categories are included, the *registry* and the (computing) *servers.* First, a user sends his/her inquiry to the registry about what service(s) he/her requests. The registry responds to the user's request, retuning ID'S of server(s) which can perform the requested service(s). Then, the user sends message(s) to such server(s), requesting the service(s). The requested server(s) execute the necessary operation(s) and return the result(s) to the user. If necessary, the requested server may sends its own request to other servers.

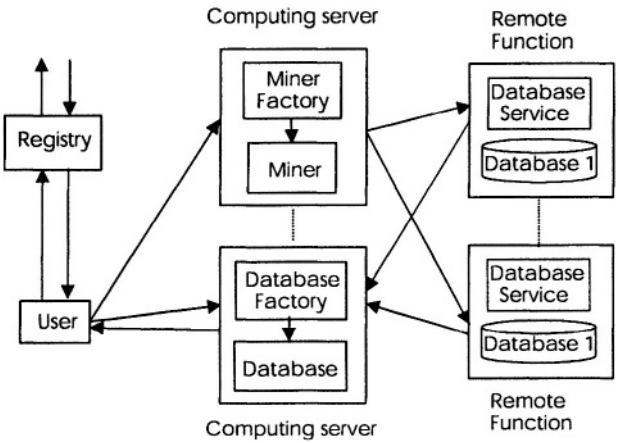

Figure 4. Operational model of the grid computing.

Issues concerned with the incorporation of fault tolerance into such environments are argued (Tohma 2003). First, the registry plays a key role to the fault tolerance in such a computational environment. It must tell the user the fault-free servers which can satisfy the user's request, excluding faulty ones. Therefore, it must also maintain the information of which servers are fault-free and which are faulty. The fault tolerance of the registry itself is crucial.

There may be many servers of similar functionalities in such an environment. Therefore, in contrast to the case of the registry, a faulty server can easily be replaced by another fault-free one.

Servers execute their computation in the message-driven way. Therefore, the most difficult problem is by what way the user or a requesting server can recognize the occurrence of faults in the requested servers. Or reversely by

what way can the requested servers notify it to the user or a requesting server? The concept of neighborhood and its consensus is advantageous in alleviating this difficulty. However, since the neighborhood consists of three servers of similar functionalities and the user or a requesting server must send a message to each of three servers in a neighborhood, the nine-fold of a message are transmitted through the network in the worst case. Further, each server in a neighborhood exchanges its computational result to each other. Thus, the communication overhead is considerable.

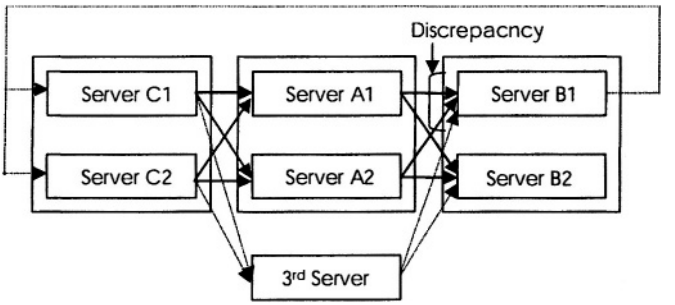

Figure 5. Duplicated operation in a pair.

Instead, it is proposed that the computation is normally duplicated in a pair of servers as shown in Fig.5. When the operation of a server in a pair is abnormal, a discrepancy between the paired computational results is found at the input of servers in the succeeding pair. Then, the server in the succeeding pair requests the re-computation by the third server which receives the necessary inputs from the preceding pair and provides its result to the succeeding pair. Since each server in the succeeding pair receives three-fold inputs, it can restore the correct input.

Servers in a pair are not necessarily the same one, and may reside remotely to each other on a network. The use of differently designed servers in pairs may benefit by the design diversity against the intrusion.

However, the grid computing needs more detailed investigation about how granularity or atomicity the functionality of grid computing needs, what language should be developed to describe services requested by users, etc.

2.4 New frontier of dependable computing

One of the clearest characteristic to contemporary computing systems is the continuous augmentation of their functionality. New functions and/or capabilities are added rather independently even under the continuation of services so far provided to users. Then, a new type of harassments arises,

that is, functions and/or capabilities added newly may conflict to other functions which already reside in the system and make some of them not to work properly. This is called the *feature interaction.*

Typical examples of feature interaction can be found in telephony systems. Consider, for example, that function OCS (Originating Call Screening) to make connection from subscribers A to C prohibited has been installed. However, if new function CF (Call Forwarding) to have telephone call from A to subscriber B automatically forwarded to C is added, and if A calls B, A is connected to C by way of B. The intention of OCS not to connect A to C is thus violated. In recent intelligent communication systems, the feature interaction has become one of real problems against providing dependable communication services.

The difficulty is how to find (detect) possible feature interaction, when functions and/or capabilities are added to a communication system without considering mutual interactions to each other.

Generally, a service is specified by the sets of subscribers, rules, predicates, and events. Further, to describe rules in a general form, variables are introduced and are instantiated, when rules are applied to actual situations. Rules of such forms as *r: pre – condition* [*event*] *post – condition* define post-conditions to which pre-conditions are reduced, when rules are applied to preconditions and the events are activated. Pre-conditions are represented by predicates and/or the negations of predicates, while post-conditions by predicates only.

A state is defined to be a set of predicates with their variables instantiated. For example, assume that subscribers are *A* and *B,* variables *x* and *y,* respectively. Further, the predicates are {*idle*(*x*), *dialtone*(*x*), *busytone*(*x*), *calling*(*x,y*), *talk*(*x,y*)}. Then, {*dialtone*(*A*), *dialtone*(*B*)} is a state. When *A* dials *B* , pre-condition {*dialtone(A)*,¬*idle*(*B*)} invokes such a rule *dialtone*(*x*),¬*idle*(*y*)[*dial*(*x, y*)]*busytone*(*x*) and post-condition *busytone*(*A*) results by the execution of event *dial*(*A,B*). Thus, state {*dialtone*(*A*), *dialtone*(*B*)} transfers to a new state {*busytone*(*A*), *dialtone*(*B*)} as shown in Fig.6.

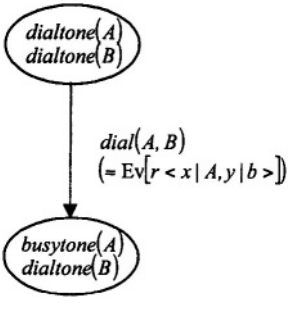

Figure 6. State transition.

In this way, all state transitions are defined for the set of rules of a service. The feature interaction can be detected by checking the existence of paths from the initial state to states (reachability check) where the feature interaction holds.

When analyzing real situations, the number of states becomes prohibitively large in general. To cope with this state explosion, the state space and the state transition relation as well as the condition of feature interaction are represented symbolically by Boolean functions, which are calculated by symbolic model checking tools. However, such Boolean formula is lengthy so that its simplification is still one of challenging issues.

Yokogawa, et al proposed a new way of representation (encoding) (Yokogawa 2003), noting that only a small fraction of state variables are usually involved in each state transition. Since the size of formula can be reduced, this new way benefits often in the possibility to explore larger state space. The considered services are as follows:

- Call Waiting (CW) allows subscribers to receive the second incoming call while they are already talking.
- Call Forwarding (CF) allows subscribers to have their incoming call forwarded to another address.
- Originating Call Screening (OCS) allows subscribers to specify in the screening list their outgoing calls to be either restricted or allowed.
- Terminating Call Screening (TCS) allows subscribers to specify in the screening list their incoming calls to be either restricted or allowed.
- Denied Origination (DO) allows subscribers to disable any call originating from the terminal. Only terminating calls are permitted.
- Denied Transmission (DT) allows subscribers to disable any call terminating at the terminal. Only originating calls are permitted.
- Direct Connect (DC) is the so-called hot-line service. When x subscribes to DC and it specifies y as the destination address, x is directly calling y simply by offhooking.

As the feature interactions, the *nondeterminism* and the *invariant violation* are considered. The nondeterminism is the situation such that two or more functionalities of different services can be activated simultaneously and therefore, which functionality should be actually performed is not determined. The invariant is a property to be held at any time.

The experiment was conducted to detect the nondeterminism for 11 combinations of services with four subscribers. The performance improvement is shown in Table 1. Times in column 'Trad.scheme' are measured by using an implementation of Chaff (Moskewicz 2001) and represented relative to those in column 'time', respectively: The latter are obtained by using the new method. The invariant violation was also checked very efficiently.

Table 1. Comparison of performance of detecting the nondeterminism.

Combination	time	Trad.scheme
CW+CF	1	1634
CW+DT	1	44.1
CW+OCS	1	113.8
CW+TCS	1	386.9
CF+DT	1	2676
CF+OCS	1	1966
CF+TCS	1	3255
DC+DO	1	43.5
DT+OCS	1	38.2
DT+TCS	1	93
OCS+TCS	1	101

3. EXTENDED APPLICATION OF DEPENDABILITY CONCEPT TO OTHER WORLDS

Modern dependable computing has many application fields from business information systems including financial, data communications, and etc. to online real time control systems including chemical process, manufacture, aerospace, etc. The concept of dependability has many aspects such as reliability, maintainability, safety, evaluation, etc. and fundamentally it covers many systems not only computing and structural systems but also social and human systems such as organizations and inspections. That is, we can extend the concept of dependability into many other worlds, and it has potentially huge application fields in the real world.

In these applications the most critical one will be a system, which affects a person's life, that is, a safety system. Safety has a strong relation with reliability but is fundamentally different concept from reliability. "Reliability" targets to maintain the given functions, but "safety" targets to avoid dangerous situations in which a sense of values of related person or current society is concerned. For example, if a bullet train is stopped from a fault of safety devise, the safety is maintained (high) but the reliability is lost (low).

In this section two activities on safety issues, which are currently conducted in Japan, are explained. One is an activity on the attempt to construct Map on Safety, which systematizes the safety concepts and safety technologies applied commonly to many safety fields. The other is harmonized activity on safety standard of machinery in Japan with international safety standards.

3.1 Map on Safety or Safety Mandala

This is an attempt toward establishing a new overall discipline on safety (which we would like to call newly "safenology") by unifying safety engineering and safety science with social and humanity sciences. For the first step of the purpose we list up many key words concerning or related to safety and cluster them into categories, which consist of three hierarchal levels and one appendix as shown in Fig. 7. We would like to call it the Map on Safety (first we called it safety map, but we prefer now to call it safety Mandala, which means, in Buddhism, a map illustrating the structure of conceptual essences) (Mukaidono 2002).

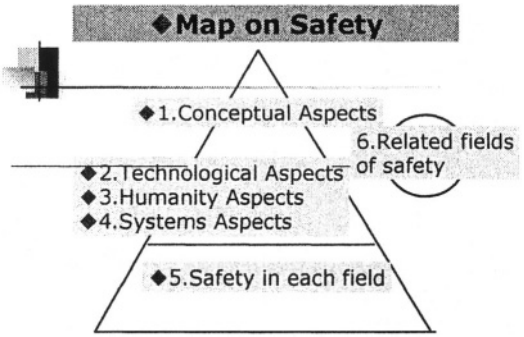

Figure 7. Map on safety.

The first level of the map on safety is (1) Conceptual Aspects, which includes fundamental concepts on safety. The second level is made of three categories (2) Technological Aspects, (3) Humanity Aspects, and (4) Systems Aspects, which are commonly used in many fields of safety. The third level of safety on map is (5) Safety in each field, which consists of each domain specific safety such as machine safety, chemical safety, nuclear power safety, etc. The appendix is (6) Related fields of safety.

The following are examples of key words clustered into each category.

(1) Conceptual Aspects
  (1-1) What is safety
    Definitions of safety, risk, tolerable risk, hazards, danger, safety target
  (1-2) A sense of values in safety
    Responsibility, safety versus cost/efficiency/ethics/convenience, safety culture
  (1-3) Humanity in safety
    Mistakes, habit, human's reliability
  (1-4) Structure of safety
    Defend what, from what, how, under what name ?

(2) Technological Aspects
Technologies on evaluation, prevention, maintenance, damage reduction, prevention, inherent safety design, fault tolerance, fail safe, fail soft, and fool proof
(3) Humanity Aspects
Human machine interface, miss uses, ergonomics, education, peace in mind
(4) Systems Aspects
Management, assessment, standardization, regulation and norm, certification
(5) Safety in each field
Machine safety, nuclear power safety, traffic safety, chemical safety, product safety, material safety, food safety
(6) Related fields of safety
Crisis management, security, insurance, court systems, law

3.2 Recent activities in standardisation on safety of machinery in Japan

Standards for safety of machinery started from Europe. EN292 (Safety of Machinery--Basic concepts, general principle for design) was originated from British and German safety standards came into effect on 1991, and used as harmonized standards of EC Machine Directives, which are mandatory. ISO/TC199 (Safety of Machinery) was established to discuss ISO12100 (Safety of Machinery—Basic concepts, general principle for design) based on EN292 and related many international standards of safety of machinery. Japan has JIS (Japan Industrial Standard) systems but not enough for machine safety. According to TBT (Technical Barriers to Trade) agreement, Japan started in 1995 to harmonize JIS with international standards, and the safety of machinery standards are introduced actively into JIS based on ISO/TC199. We started the research for developing international safety standards, for example, vision-based protective devices and electronic safety control circuit module based on fail safe technology. Furthermore, in the cooperation with **Asia-Pacific** countries, we are pushing researches and making efforts to propose the international safety standards to ISO and IEC.

The following is a short history of the safety machinery standards in Japan and Europe.

- 1989 (Europe) EC Machine directives
- 1990 (International) ISO/IEC guide 51: Safety Aspects - Guidelines for their inclusion in standards
- 1991 (Europe) EN292: Safety of Machinery - Basic concepts, general principle for design

- 1991 (International) ISO/TC199: Safety of Machinery
- 1992 (International) ISO/TR12100: Safety of Machinery - Basic concepts, general principle for design
- 1995 (International) TBT (Technical Barriers to Trade) agreement
- 1995 (Japan) Declaration to harmonize JIS (Japan Industrial Standard) with ISO, IEC within 5 years
- 1998 (Japan) TR B 0008, 0009 (corresponding to ISO/TR12100)
- 2001 (Japan) Guideline for comprehensive safety norm on Machinery (corresponding to ISO/TR12100) into effect by Ministry of Labor, Health and Welfare
- 2001 (Japan) Asia-Pacific Machinery Safety Seminar started every year (China, Korea, Thailand, Singapore, India, Philippines)
- 2001 (Japan) Research for developing International Safety Standards
  (1) Development of vision-based protective device
  (2) Electronic control circuit module
- 2003 (International) ISO12100: Safety of Machinery - Basic concepts, general principle for design
- 2004 (Japan) JS B 970 (corresponding to ISO12100)

4. CONCLUSION

As technological innovation emerges, we face always new challenges to dependable computing. This paper has reviewed researches as well as organizational activities currently conducted in Japan, noting their new characteristic and nature.

REFERENCES

Foster, I., Kessselman, S., Nick, J. M. and Tuecke, S., 2002, Grid Services for Distributed System Integration, Computers, *IEEE Computer,* Vol. 35, No. 6, pp. 37-46, June 2002.

Mishia, T and Akaike, T., 2003, PREGMA: A New Fault Tolerant Cluster Using COTS Components for Internet Services, *Transactions on Information and Systems, The Institute of Electronics, Information and Communication Engineers (IEICE),* Vol. E86-D, No. 12, pp. 2517-2526, December 2003.

Mendelson, A. and Suri, A., 2000, Designing high-performance & reliable superscalar architecture - the out of order reliable superscalar (O3RS) approach; *Proc. IEEE/IFIP International Conference on Dependable Systems and Networks,* pp. 447-481, 2000.

Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L, and Malik, S, 2001, Chaff: Engineering an efficient sat solver, *Proc. 39th Design Automation Conference,* 2001.

Mukaidono, M., 2002, The Map on Safety - Toward to establish a new overall discipline on safety, *Proc. 3rd International Forum on Safety Engineering and Science (IFSESIII),* 2002. (See http://www.sys.cs.meiji.ac.jp/~masao/kouen/safetymandala.file/frame.htm.)

Sato, T., 2003, A Transparent and Transient Faults Tolerance Mechanism for Superscalar Processors, *Transactions on Information and Systems, IEICE,* Vol. E86-D, No. 12, pp. 2508-2516, December 2003.

Tohma, Y., 2003, Consideration of Fault Tolerance in Autonomic Computing Environment, *ibid,* 2503-2507.

Yokogawa, T., Tsuchiya, T., Nakamura, M,, and Kikuno, T., 2003, Feature Interaction Detection by Bounded Model Checking, *ibid,* 2579-2587.