On Engineering Standards Based Carrier Grade Platforms

Francis Tam
Nokia Research Center
P.O. Box 407
FI-00045 NOKIA GROUP
Finland

francis.tam@nokia.com

ABSTRACT

The remarkable pace of advancement in communications technologies and the exponential growth of the market have pressured network equipment providers into producing more features in products in a much faster rate at lower costs. The strategy of buying constituent components instead of building one's own has shown promises in achieving these goals. In this paper, we articulate the needs for following standards, and discuss the impact and the required changes for engineering a standards based carrier grade platform. The focus is on the introduction of an availability management middleware, in the form of an off-the-shelf component, and its impact on the product life cycle. By applying and adapting a selection of research results from the dependability community, we show that the telecommunications industry can benefit and achieve its target of reducing development costs.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management – *life cycle*; K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software maintenance, software process, software selection*; C.4 [**Performance of Systems**]: *reliability, availability, and serviceability.*

General Terms

Design, Reliability, Standardization.

Keywords

Robustness, Upgrade.

1. INTRODUCTION

The modern telecommunications market is huge and can be thought of being made up by three groups consisting of the end users, operators and network equipment providers. Figure 1 shows how the never-ending challenges of expectations funnel down from the end users as requirements onto the operators and then to the network equipment providers.

At the top of the model are the end users. The typical expectation from this group is to have new features frequently introduced by the operators. These services should also be affordable and dependable.

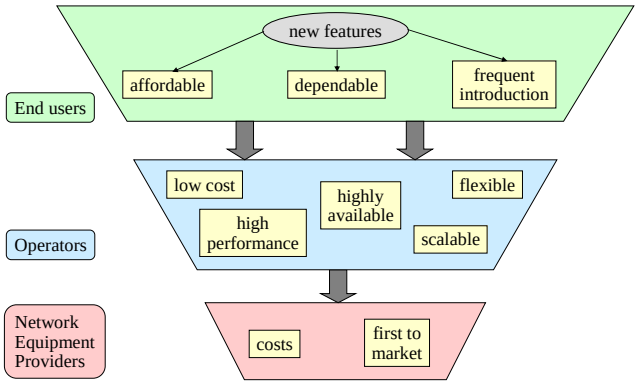

Figure 1 Expectations and requirements

The second layer represents the operators. They must have low implementation and operating costs in order to make their services affordable to the end users. The expectation is that the underlying communications infrastructure equipment must have high performance to cater for the potentially large number of service subscribers. Another expectation is the flexibility of the network equipment so that new features can be deployed very quickly. Scalability, both upwards and downwards, is important for the expanding and emerging markets respectively. Last, but not least, the network equipment must be highly available because it is one of the key factors that has a direct impact on the revenue generation for this group. The challenge for this group is to be able to respond, in a timely manner, to the diverse market needs.

At the bottom of the model are the network equipment providers. Since their customers are the operators, they have to satisfy the long list of requirements and meet the high expectations that have been passed downwards. As if this is not enough, the fierce competition in this industry pushes this group to have an ever shorter time to market, at lower development costs, in order to win businesses.

There are many ways to achieve the goals of reducing development costs. For example, by using advanced software technology, development process improvement, automatic code generation, standardised components are some of the few solutions commonly used in the industry. In this paper, we discuss the latest trend of exploiting standardised components in the carrier grade base platforms construction. We show how the product development process has to be adapted, taking into account the need for integrating off-the-shelf components into the final product. This paper focuses on the software side, in

EFTS'07, September 4, 2007, Dubrovnik, Croatia.

particular the high availability middleware for carrier grade base platforms. This also serves as a case study of how the telecommunications industry can apply and benefit from the research results produced by the dependability community.

2. CARRIER GRADE BASE PLATFORM

The term carrier grade base platform is widely used in the network equipment provider community. It refers to a class of systems that possess a notable characteristics of high availability. This carrier grade quality can be traced back to the early telephone switching systems in which continuous operation for 99.999% of the time was expected. Other properties such as high performance and scalability are also attached to this term. As the name base platform indicates, this class of systems serve as a foundation on which higher level of functionality are implemented. In essence, it is an attempt to capture, as much as possible, a common set of features that are required for a broad range of network element products. Note that a carrier grade base platform on its own does not provide any specific telecommunications capabilities. The application that runs on the base platform would determine what the network element is.

Figure 2 illustrates an architecture for a carrier grade base platform that is commonly used in the industry. At the hardware level, redundant communication networks are used. For example, redundant interconnect such as Ethernet is used in loosely coupled systems, while replicated backplane bus structure is utilised in closely coupled implementations. The interconnected nodes typically contain hardware resources such as CPUs, Network Processors, Digital Signal Processors, disk storage and switches.

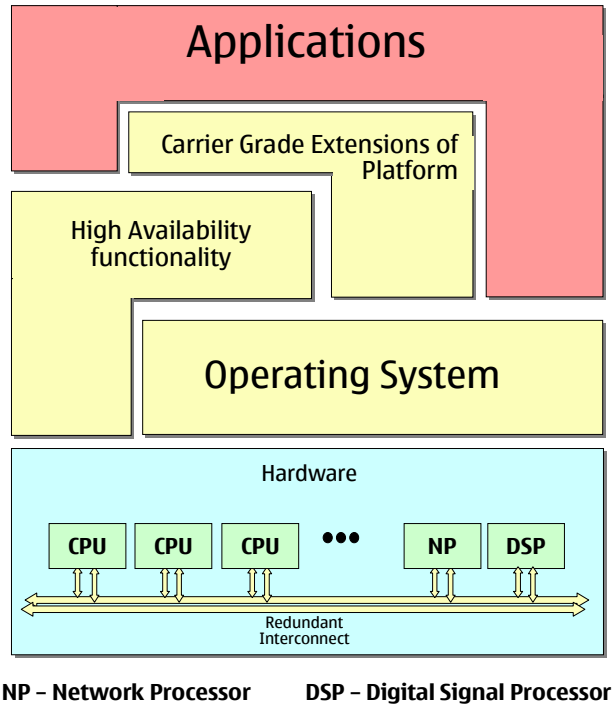

Figure 2 Carrier grade base platform architecture

The operating system controls all the hardware resources and provides an abstract view upon which software can be written. One of the key benefits of having the operating system layer is that the software above this level can be decoupled from the hardware. This is of particular advantage when we consider the typical scenario that members of a product family essentially have a similar set of features, and yet due to the usage in the field the capacity requirements and expected price range for the customers could be very different. The operating system layer ensures that the software does not tie to a specific kind of hardware. As and when a less powerful, hence less expensive hardware is used for a particular configuration of a product, the higher level software can be ported to the new hardware environment with relative ease.

The high availability function provides support for the capabilities such as the provisioning of protective redundancy in different models, for example active standby, N+M depending upon the application requirements. Another group of functions is fault management, which involves the detection of faults and sending of notifications to the concerned parties for possible recovery. Finally load balancing and sharing are typically considered in this category as most of these schemes tend to exploit the inherent redundant resources in the system during the error free periods of operation. This collection of features essentially delivers the carrier grade characteristics to the platform.

The carrier grade extensions usually contains those functions that are commonly used in the telecommunications domain. For example, signalling protocol stacks, alarm management, performance management, configuration management, directory services, to new just a few. Although these extensions provide the base platform with telecommunications system specific functions, the particular application that run in the platform would turn the base platform into a concrete network element. For example, if the application is a Serving GPRS (General Packet Radio Service) Support Node (SGSN), the resulting platform would therefore become a SGSN network element in the packet switched domain in the Public Land Mobile Network. On the other hand, if the software is a Gateway GPRS Support Node (GGSN) the final product would become a GGSN network element instead.

3. PRODUCT LIFE CYCLE

3.1 Simplified Model

We have adopted a product life cycle model from [2] to facilitate discussions in this paper. It has four phases: specification, design, production and operation. It is assumed that the requirements, mainly expressed in natural languages as needs, have been captured prior to the specification phase.

The outcome of the specification phase is essentially a contract defining the mission (functional characteristics) describing the desired service and its duration of operation, and the non-functional characteristics dealing with product dependability requirements such as reliability, availability etc.

Design is a process consists of successive steps of transforming a specification into a model that represents an abstraction of the future product. Each step refines the model from the previous step, for example, from a general design to detailed design. Sometimes the final step of realisation is included to complete the design phase. Hardware and software designs have similar processes and means. There are three design levels. First, the behavioural level formalises the specification on what the product needs to do. Second, the structural level breaks a system down

into an interconnection of sub-systems. Third, the technological level materialises the designed model to an executable system.

The production phase is the integration of the realised design systems with other required products, which are either created by the same organisation or bought from outside. The outcome of this phase is the created product.

The operation phase is the duration of time when a product is placed in the final environment for use according to its mission. Maintenance is needed when actions are performed on the product structure during its useful life, either as corrective actions or improvements. As a result, the specification may be modified and lead to the design of a new version of the initial system. The maintenance phase is generally underestimated, especially in products where the life duration are longer than their creation time. This is especially true in some telecommunication deployment.

We have identified three areas in the product life cycle that require attention when engineering a standards based carrier grade platform. We are particularly interested in the impact of incorporating the carrier grade characteristics on the overall product life cycle. In the design stage, the chosen standards must naturally be reflected in the overall architecture. In the production stage, steps must be taken to ensure that high availability is maintained even when off-the-shelf components are used. In the operation stage, means must be made available to reduce the mean time to repair. We examine each area in turn in the following sections.

3.2 Architecture

The architecture of a carrier grade base platform shown in figure 2 is not the result of an accident. It has been refined over the years. In the early stages of development, a company typically constructed their products in a vertical fashion. In other words, a product was built from the application all the way down to the hardware. Needless to say, this approach took a considerable amount of development time and costs. One popular solution as witnessed in the industry is to buy the constituent parts whenever possible, instead of building one's own. In order to prevent from being locked in by a single vendor who supplies such a part, the use of standardised components is regarded as a risk management strategy because there tend to be more than one vendor to choose from. In addition, it is very often that one could select the best product available.

As the development of technology used in constructing carrier grade base platforms progresses, we have already seen the replacement of 1) custom-built processors by their commercially available counterparts, 2) proprietary hardware by standards based computing elements such as the PICMG's AdvancedTCA [11], and 3) in-house development of systems software by standards operating systems such as Carrier Grade Linux [3]. This trend of using more and more standardised components continues to move upwards to the high availability functionality level. To date, the Service Availability Forum's standards [12] are predominant and is striving to become the de facto standard of high availability middleware in carrier grade base platforms.

The Service Availability Forum is an industrial coalition formed in December 2001 by a group of communications and computing companies working together on open standards for availability middleware in the telecommunications domain. Its mission is to foster an ecosystem that enables the use of off-the-shelf building blocks in the creation of highly available network infrastructure products, systems and services. Thus far, the Forum has already released several versions of the Application Interface Specification and Hardware Platform Interface.

Figure 3 shows the architecture of the service availability middleware. The Hardware Platform Interface (HPI) primarily deals with the monitoring and controlling the physical components of a carrier grade computing platform. By abstracting the platform specific characteristics into a model, an HPI implementation provides the users with standard methods of monitoring and controlling the physical hardware via this abstraction. The Application Interface Specification (AIS) defines the capabilities of a high-availability middleware, interfacing with applications and the underlying carrier grade hardware platform. The AIS abstracts the high-availability characteristics into a model upon which an implementation provides standard methods to the application developers to respond and manage events such as failures. AIS defines an extensive Application Programming Interface for both threaded and non-threaded applications, supporting synchronous and asynchronous interfaces.

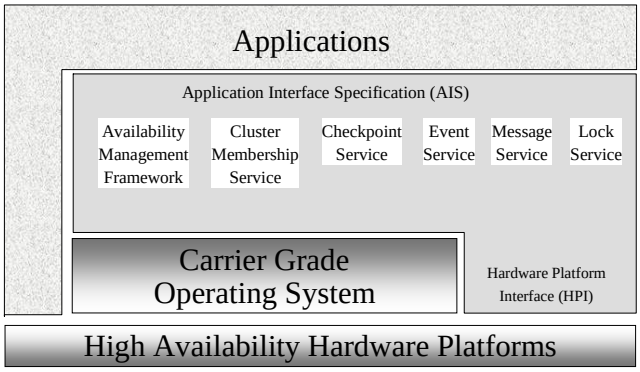

Figure 3 Positioning of Service Availability Forum middleware

The availability management and control capability are captured in the Availability Management Framework (AMF). AMF is a software entity that provides service availability by coordinating redundant resources within a logical cluster consisting of cluster nodes. From the AMF perspective, an application is structured as logical entities known as components. Each of these components implements state models and callback interfaces according to the standard.

Additional services, known as AIS Services, have also been defined in AIS to support the development of a carrier grade system. Cluster Membership Service provides applications with the current cluster node membership information. Checkpoint Service provides a means to record checkpoint data incrementally in order to protect an application against failures. Event Service provides an asynchronous communication means between multiple publishers and multiple subscribers over an event channel. Message Service provides a buffered message passing system based on the concept of a message queue, which is expected to preserve messages during a switch-over. Lock Service provides a means to synchronise access to shared resources among application processes on different nodes in a cluster.

By using AIS as the high availability middleware, an application can now focus on the functionality of the network element. In addition, as and when the same application is required to be ported to another hardware platform, the needed changes would be minimal.

3.3 Dependability Benchmarking

Fault prevention and fault removal techniques are generally used throughout the product life cycle. In the production stage, this means that some forms of assessment must first of all be carried out to determine if the carrier grade requirements have been met in the currently realised subsystems. If there are still residual faults they must be removed before proceeding to creating the final product. In software, this involves not only the application, but also the run-time environment and other off-the-shelf software components. Since the latter are third party software, their development cycle are however outside of our control. As such, other means must be used in order to ensure that the selected products are indeed dependable.

The benefit of having open standards is the choices available from different vendors. In practice however, when one chooses a product from a pool of suppliers robustness is very often over shadowed by performance. The "seems to work well" criterion is typically the most commonly used without any further consideration of the robustness of such an implementation. Worse still, the crashing or hanging of such a third party software causes the whole system to fail, too. The challenge here is to develop the appropriate technology for measuring and comparing robustness of off-the shelf software in order to support the product selection process.

Figure 4 shows the vision of a dependability benchmark machine that supports the selection process for a suitable third party availability management middleware. The principle is to have the dependability benchmarking machine executing a set of benchmark suites on a number of off-the-shelf availability management middleware products according to some pre-defined fault-load and workload. The benchmark suites contain test cases that are designed to uncover how the system under test react to exceptional inputs and stressful environmental conditions. The fault-load and workload are configurable according to the requirements exhibited by the specific network element that runs on the base platform. The benchmark results for each vendor's implementation can then be compared.

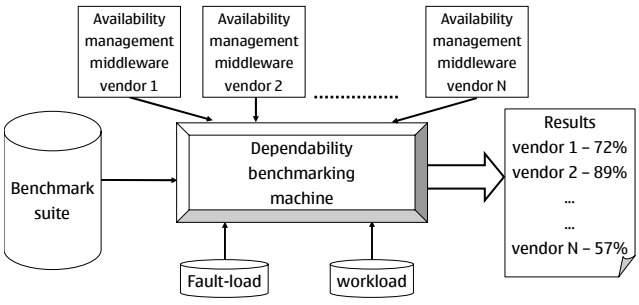

Figure 4 Vision of dependability benchmarking

We have designed and implemented a prototype [5] of this vision and experimented on three versions of Service Availability Forum's AMF implementations. The test cases are primarily for stretching the middleware with exceptional inputs. We have automated the generation of test cases directly from the interface specification by means of a template-based approach. In addition, we have incorporated mutation-based testing into the benchmark as a means of improving fault detection on more complex scenarios, which usually involve the accumulation of state information during a series of calls to the availability management middleware. Note that the orderly sequencing of certain calls to the availability management layer is a must because the middleware is the machinery holding the entire system's current state information. The preliminary results have shown that different test methods were able to detect different failures successfully. Future work includes establishing a representative, configurable workload and fault-load for the dependability benchmarking machine for each class of network element application. This allows for the tailoring of different measurement criteria for different products.

3.4 Online Software Upgrade

During the operation stage, maintenance of a network element product is one of the key concerns. Nowadays hardware technologies have advanced to a point where replacement of physical components can be performed without powering off the system, thus interruption of service can be avoided. However, changing software, whether it is for bug fixing or version upgrade, may lead to a potential stoppage due to the need of restarting or reloading of data in some cases. Currently, this kind of unavailability of service is usually covered by the scheduled downtime as stated in the service level agreements between the operators and network equipment providers. Due to fierce competitions, there is a growing trend for the operators to remove this kind of scheduled maintenance time periods altogether. This has put demands on the network equipment providers to deploy online software upgrade in the field during a product's operation stage.

Upgrading software online with minimum or no service interruption is difficult to achieve. It requires a considerable level of support to be put in place. For example, software image management for the distribution, installation and configuration; version management for the software; control and monitoring of the upgrade procedure; and recovery from errors during the upgrade, to name just a few. Depending on the application requirements, other considerations may include how the upgrade should be carried out, for example, by using rolling upgrade or split-mode methods.

Intuitively, online software upgrade interrelates to the protective redundancy of a system because it is the redundant resources that keep the system running while some parts of it are being replaced. Since the carrier grade base platform in this paper uses the Service Availability Forum's availability management architecture, it is logical to anticipate that some kind of upgrade capabilities be defined by the Forum to work alongside. Indeed the initial direction of a Software Management Framework has been reported in [14]. It is based on the notion of extending the representation of the types of entities in the system with versioning information. An upgrade operation is essentially an action to change a particular entity from one version to another. Interruption of services can be prevented or minimised by cooperating with the Availability Management Framework when

needed. A means, in the form of XML schema, is provided to express how an upgrade should be conducted. The description includes what upgrade method should be used (e.g. rolling, split-mode), what kind of error recovery action should be taken etc. It is expected that the Service Availability Forum will release the Software Management Framework standards soon.

4. RELATED WORK

There are plenty of related work in the general area of high availability that have been reported. Due to the space limitation, this section only review works that are narrowly linked to the development process for standards based systems.

4.1 Architecture

In the standards architecture space, the Object Management Group's Fault Tolerant CORBA [6] is the only alternative to the Service Availability Forum's Application Interface Specification. Fault Tolerant CORBA is based on the notion of object group in which services are replicated according to a selected strategy such as request retry, redirection to an alternative server, passive (primary-backup) and active replication. Interfaces for the replication manager, fault manager (faults detection and notification), logging and recovery management have been defined. At this high level of abstraction, both Fault Tolerant CORBA and AIS provide a very similar solution. However, Fault Tolerant CORBA attempts to be generic across wide-ranging applications from the enterprise domain to the communications and mission critical systems, which have very different dependability requirements. As a result, the interface definitions do not contain the required support for a specific industry. For example, in telecommunications applications, significant amount of effort would have been needed to integrate a Fault Tolerant CORBA implementation. This is due to the need for an application to explicitly interact with the other well established standards such as state management, alarm reporting and event management in this domain.

4.2 Dependability Benchmarking

The Ballista [9] testing methodology was a pioneer in robustness testing of off-the-shelf components such as an operating system. It was first shown to compare the robustness of 15 POSIX operating systems. The approach was then applied to other software components such as the CORBA middleware [10] in order to test the robustness of the C++ client side exception handling capabilities on two major versions of three ORB implementations on two operating systems.

DBench [1] was a 3-year research project on dependability benchmarking funded by the European Commission under the Information Society Technologies Fifth Framework Programme. The project has developed a framework and guidelines for defining dependability benchmarks for off-the-shelf software components. The specific application areas covered were automotive control systems, onboard space control and enterprise systems.

The dependability benchmarking approach discussed in this paper has been, to a large extent, influenced by the fundamental approach of Ballista and an adaptation of the DBench framework and guidelines to the off-the-shelf availability management middleware for the telecommunications domain.

4.3 Online Software Upgrade

One popular means of upgrade used in standards based systems is RPM (Redhat Package Manager) under the Linux Standard Base specifications [4]. It defines the format which is necessary to resolve compatibility and dependency issues. Together with the corresponding utilities, it handles the distribution and installation of new software. However, it does not address any of the procedural aspects of upgrades.

The Open Group's Distributed Software Administration (XDSA) [8], which is based on the IEEE 1387.2 standard, goes beyond the issues tackled by RPM. The XDSA specification defines the package layout and addresses the software administration requirements of distributed systems and applications. The specification defines a set of administration utilities that enable management applications to control the packaging, distribution, installation, update and removal of software across multiple nodes of a cluster. XDSA still does not address the service availability issue in any way.

The Object Management Group's Online Upgrade specification [7] addresses the upgrade of a CORBA object implementation. Interfaces have been defined for an *upgrade manager* to prepare an object for upgrading; to perform the upgrades of one or more objects; to rollback upgrades of objects; and to revert an object from its new implementation to its old implementation. The first step of an upgrade is to put both the old and new implementations into an object group, followed by the query to the old implementation to determine whether it is safe to perform the upgrade. A vendor specific implementation of the *upgrade mechanisms* is expected to stop new messages being delivered to the old implementation, instead, it queues them for delivery to the new implementation. When the old implementation responds affirmatively to the upgrade request, the current state of the old implementation is transferred to the new one. The queued messages are then replayed to the new implementation, and all future messages for this object group are directed to the new implementation. The old implementation is then removed from the object group before the upgrade is committed. If some part of the upgrade fails, an implementation specific rollback and reverting an upgrade could be used to recover.

Some central features in the Online Upgrade specification are missing, for example, object implementations are not versioned and concurrent operation of the old and new implementations is not supported. In addition, it does not really support a true online upgrade because services provided by the object to be upgraded are not available during the upgrade, although the incoming messages for that object are preserved. To overcome this limitation, it has been recommended that the Fault Tolerant CORBA should be used in order to exploit the inherited redundancy for having minimum, or even no loss of service during an upgrade. However, there is no guidelines on how to integrate these two in a seamless manner.

5. CONCLUSIONS

In this paper, we have discussed the current directions of developing standards based carrier grade base platforms in the telecommunications industry. As a sanity check against our approaches, we have attempted to map the reported activities onto a recent study conducted by Siewiorek et al. [13], which suggests that the industrial trends and experimental research in

dependability are based on three observations: 1) Shifting error sources. 2) Explosive complexity. 3) Global volume. Figure 5 highlights where the discussed steps are positioned in the simplified product life cycle model. For trend 1, the shift to more software failures and less tolerance to planned outages have necessitated dependability benchmarking and online software upgrade respectively. For trend 2, the use of standards can result in some control over the explosive complexity issue. For trend 3, the use of online software upgrade is essential to improve the dependability in systems management. We also concur with the authors' conclusions that trends 1 and 2 are well underway with considerable body of research, whereas trend 3 is still relatively young.

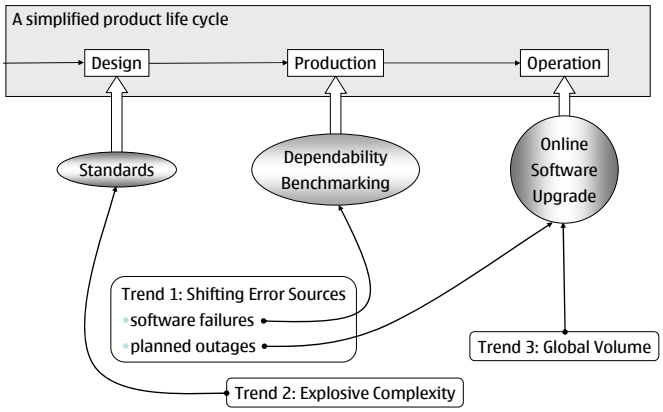

Figure 5 Mapping of research trends onto product life cycle

We believe in using standards based carrier grade base platforms can reduce development costs and achieve the required cycle time reduction goal. The additional steps identified in the product life cycle, together with the proposed supporting technologies, are considered to be critical if such an approach is adopted. When the technologies are matured and commercially viable to be deployed in the development chain, we can anticipate a considerable and quantifiable return on investment.

6. ACKNOWLEDGMENTS

The work presented in this paper had been carried out in the project "Highly Available Services: Standardisation and Technology Investigation" during 2001-2006. The project was funded by Strategy and Technology in Nokia Networks, which is now part of Nokia Siemens Networks.

7. REFERENCES

[1] The DBench project. *Dependability Benchmarking*. http://www.dbench.org.

[2] Geffroy, J-C. and Motet, G. *Design of Dependable Computing Systems*. Kluwer Academic Publishers, 2002.

[3] The Linux Foundation. *Carrier Grade Linux Workgroup.* http://www.linux-foundation.org/en/Carrier_Grade_Linux.

[4] The Linux Foundation. *Linux Standard Base Core Specification*. http://www.linux-foundation.org/en/Specifications.

[5] Micskei, Z., Majzik, I. and Tam, F. Comparing Robustness of AIS-Based Middleware Implementations. In *Service Availability. Lecture Notes in Computer Science 4526.* (Proceedings of the 4th International Service Availability Symposium, New Hampshire, U.S., 21-22 May, 2007). Springer, 2007, 20-30.

[6] Object Management Group. Fault Tolerant CORBA. In *Common Object Request Broker Architecture: Core Specification. Chapter 23, version 3.0.3, formal/04-03-21*. March 2004.

[7] Object Management Group. *Online Upgrade Specification.* SMSC review copy, smsc/05-10-02. November 2005.

[8] The Open Group. *Systems Management: Distributed Software Administration (XDSA)*. CAE Specification. Document number C701. 1998.

[9] Koopman, P. and DeVale, J. Comparing the Robustness of POSIX Operating Systems. In *Proceedings of the 29th Annual International Symposium of Fault-Tolerant Computing*. (Wisconsin, United States. 15-18 June, 1999). IEEE Computer Society 1999, 30-37.

[10] Pan, J., Koopman, P., Huang, Y., Gruber R. and Jiang, M. Robustness Testing and Hardening of CORBA ORB Implementations. In *Proceedings of the International Conference on Dependable Systems and Networks* (Sweden, July 2001). 141-150.

[11] PCI Industrial Computer Manufacturers Group. *PICMG 3.0 AdvancedTCA™ Base Specification.* http://www.picmg.org/v2internal/specifications.htm.

[12] Service Availability Forum. *Application Interface Specification*. http://www.saforum.org.

[13] Siewiorek, D. P., Chillarege, R., and Kalbarczyk, Z. T. Reflections on Industry Trends and Experimental Research in Dependability. *IEEE Transactions on Dependable and Secure Computing, 1, 2* (April-June. 2004), 109-127.

[14] Toeroe, M., Frejek, P., Tam, F., Penubolu, S. and Kasturi, K. The Emerging SAF Software Management Framework. In *Service Availability. Lecture Notes in Computer Science 4328*. (Proceedings of the 3rd International Service Availability Symposium, Helsinki, Finland, 15-16 May, 2006). Springer, 2006, 253-270.