Dependability and Fault Tolerant Computing

* Dr. K. C. Joshi
Department of Business Administration
M.J.P.Rohilkhand University, Bareilly
kc-joshi@indiatimes.com

ABSTRACT
Software is being used for building application requiring extreme dependability. Dependability is the trustworthiness of a computer system. Software fault tolerant of a system is the heart of building trustworthy software. Trustworthy software is stable software. It does what it is supposed to do and can repeat that action time and again hereby always producing the same kind of output from the same kind of input. The software used in safety system of nuclear power plants, electronic banking, semiconductor industry are example of trustworthy software. Trustworthiness is a holistic property, encompassing security, safety and reliability. It is not sufficient to address only one or two of these diverse dimensions, nor is it sufficient to simply assemble components that are themselves trustworthy. Integrating the components and understanding how the trustworthiness dimensions interact is a challenge. Because of the increasing complexity and scope of software, its trustworthiness will become a dominant issue.

Software fault tolerant of a system is the heart of building trustworthy software. Microsoft claims to have undertaken a trustworthy computing initiative. Bill Gates sent a memo to his entire workforce demanding, “…. Company wide emphasis on developing high-quality code that is available.” Information Week, Jan. 21 2002, issues 873. p.28.

Software dependability is achieved by fault avoidance techniques (structured programming, software reuse, and formal methods) to prevent software faults or fault removal techniques (including testing, verification, and validation) to detect and delete software faults. Software fault tolerance is concerned with all the techniques necessary to enable a system to tolerate software faults remaining in the system after its developments. These software faults may or may not manifest themselves during systems operations, but when they do, software fault tolerance techniques should provide the necessary mechanisms of the software system to prevent system failure from occurring. From a business perspective, operational failures caused by software faults can translate into loss of potential customers, lower sales, higher warranty repair costs, and losses due to legal actions from the people affected by the failures.

In the present paper we will discuss in details the attributes of dependability, means to attain dependability, its role in software fault tolerance and also we study how it is a necessary component in order to construct the next generation of highly reliable and available computing systems from embedded system to data warehouse systems.

KEY WORDS
Software Faults, Dependability, Ultradependability, Embedded system, Trustworthiness

INTRODUCTION
As the faults in computer systems are unavoidable as the system complexity grows, computer systems used for critical applications are designed to tolerate both software and hardware faults by the configuration of multiple software versions on redundant hardware systems. The complexity and size of current and future software systems, usually embedded in sophisticated hardware architecture, are growing dramatically. As our requirements for and dependencies on computers and their operating software increase, the crises of computer hardware failures also increase. The impact of computer failures to human life ranges from inconvenience (e.g., malfunctions of home appliances), economic loss (e.g., interceptions of banking systems) to life threatening (e.g., failures of flight systems or nuclear reactors).

The dependability constitutes can be represented under the form of a tree as shown as:

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Means</th>
<th>Impairments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Fault Tolerance * Faults</td>
<td></td>
</tr>
<tr>
<td>Availability</td>
<td>Fault Prevention * Errors</td>
<td></td>
</tr>
<tr>
<td>Safety</td>
<td>Fault Removal *</td>
<td></td>
</tr>
<tr>
<td>Integrity</td>
<td>Fault Forecasting</td>
<td></td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Maintainability</td>
<td></td>
</tr>
</tbody>
</table>

Failures:
A system may fail because it does not comply with the specification, or because the specification did not adequately describes its function.
Faults:
An error is that part of the system state that may cause a subsequent failure: A failure occurs when an error reaches the service interface and alters the service. A fault is the adjudged or hypothesized cause of an error. A fault is active when it produces an error otherwise it is dormant.

System Failures:
The failures in computer system can be elaborated as following figure:

- Software faults
- Hardware faults
- * Design faults
- * Specifications Mistakes
- * Implementation Mistakes
- * Component defects
- * External Disturbances

The faults in computer system at hardware and software level can be seen in four ways as fault avoidance, fault masking, fault removal and fault forecasting.

<table>
<thead>
<tr>
<th>Faults Methods</th>
<th>Chip design and Hardware level Fabrication level</th>
<th>Software Level (program &amp; Implementation level)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault avoidance</td>
<td>Do not use faulty chip, test the chips well</td>
<td>Testing for removal of all bugs</td>
</tr>
<tr>
<td>Fault masking</td>
<td>Use redundancy so that the faults do not propagate</td>
<td>Use alternate algorithms</td>
</tr>
<tr>
<td>Fault removal</td>
<td>Detect fault, locate faulty chip and replace or reconfigure</td>
<td>Patch bugs</td>
</tr>
<tr>
<td>Fault forecasting</td>
<td>Through fault simulation determine unacceptable behaviors</td>
<td>Determine the effect of out rigorously incorrect results</td>
</tr>
</tbody>
</table>

The purpose of fault tolerance is to increase dependability by providing a system that delivers the required service or some degraded version of it in the face of some specific classes of faults. The types of fault tolerated can be hardware faults, hardware design faults or software design faults.

The use of hardware fault tolerance in order to achieve increase reliability dates back to some of the very early digital computers. This has been very successful and was perhaps even more important years ago when computer hardware was more unreliable than usually is nowadays.

Attributes of Dependability:
Dependability is a quality measure encompasses the concepts of reliability, availability, safety, performability, maintainability and testability.
- Reliability of a system is defined as the function that maps time duration onto the probability that the system will perform correctly for periods of such duration.
- Availability is the probability that a system is performing successfully, according to specifications at a given point in time.
- Safety is the probability that the system will perform in a non-hazardous way.
- Performability is the probability that the system performance will be equal to or greater than some particular level at a given instant of time.
- Maintainability is the probability that a failed system will be returned to operation within a particular time period.
- Testability is a measure of the ability to characterize a system through testing.

Assumptions are that all rates are constant, independent of time and that failures occur independently of each other.

\[ R(t) = e^{-\lambda t} \]  
where \( \lambda \) is called the failure rate of the system.

Expected life or MTTF (Mean time to failure) is a commonly used measure to specify the reliability of a system, and is given by

\[ E(X) = \int_0^\infty t f(t) \, dt = \int_0^\infty R'(t) \, dt \]

(where \( l = \infty \))

\[ = \int_0^\infty R(t) \, dt = \int_0^\infty e^{-\lambda t} \, dt = 1/\lambda. \]

MTTF = 1/\( \lambda \).

Similarly if \( \mu \) is the repair rate, then MTTR (Mean time to repair) = 1/\( \mu \)

Availability = \( \text{MTTF}/(\text{MTTF} + \text{MTTR}) \)
For a system $\text{sys}$ representing a combination of components $C_1, C_2, C_n$ (without redundancy), we have

$$\lambda_{\text{sys}} = \sum \lambda_{Ci}$$

Hence

$$\text{MTTF}_{\text{sys}} = \frac{1}{\sum \lambda_{\text{MTTFi}}}$$

For a system $\text{sys}$ in which components $C_1, C_2, \ldots, C_n$ are redundant, we have

$$\lambda_{\text{sys}} = \Pi \lambda_{Ci}$$

Techniques of Dependability:

**Fault Prevention:** It is achieved by formal design, quality control and access control. It is attained by a quality control techniques employed during the design and manufacturing of hardware and software. They include structured programming, information hiding, modularization etc for software and rigorous design rules for hardware.

**Fault Tolerance:** A system is said to be fault-tolerant that may continue to provide its services with faults. It is generally implemented by error detection and subsequent system recovery.

**Fault Recovery:** Recovery transforms a system state that contains one or more errors and faults into a state without detected errors and faults that can be activated again. It consists the following:

* Isolation, rollback, reboot, compensation, fail-over.
* Masking, redundancy (voting, N-version programming, redundant hardware)

**Fault removal:** It is performed both during the development phase, and during the operational life of a system. Fault removal during the development phase of a system life cycle consists of three steps: verification, diagnosis, and correction. Verification is the process of checking whether the system adheres to given properties, termed the verification conditions. If it does not, the other two steps follows: diagnosing the fault(s) that prevented the verification conditions from being fulfilled, and then performing the necessary correction. Fault removal consists:

* formal verification of implementation with respect to specifications.
* validation of specifications with respect to real environment
* fault injection and testing

**Fault Forecasting:** Fault forecasting is conducted by performing an evaluation and prediction of the system behavior with respect to fault occurrence or activation. Evaluation has two aspects:

* Qualitative or ordinal evaluation, which aims to identify, classify, rank the failure modes, or the event combinations that would lead to system failures.
* Quantitative or probabilities evaluation which aims to evaluate in terms of probabilities the extent to which some of the attributes of dependability are satisfied: these attributes are then viewed as measures of dependability.

Embedded systems pervade modern society. From consumer electronics, to automobiles, to satellites, they represent the largest segment of the computer industry. Since they are so pervasive, our society has come to depend on these systems for its day-to-day operation. The disruption caused by the outage of the Galaxy IV satellite slows how the failure of one system will be affect society. Clearly, if modern society is going to continue to depend on these systems, steps must be taken to assure they will be available when we need them. One way to make the system more dependable is to look into ultra dependability technology. The ultra dependability means that a system has been design to operate for such a long period of time without defects that testing become impractical. Automobiles and jet planes are two systems that may need ultra dependability technology. Many jet planes fly everyday, and they dependent on many embedded systems to fly, and failure of any of the plane's embedded systems may cause a major disaster. With operating lives of up to 20 or 30 years and 10,000's of planes flying, the number of hours the fleet operates becomes enormous. When faced with an enormous number of operating hours, the chances of their being an error during operation go up dramatically. Institutions like these, where the system must be as close to flawless, as possible, embedded systems designers must look to make their systems ultra dependable.

**Challenges for Future Scope:**

The difficulties in applying dependability techniques in current software engineering are quite often cultural rather then technical, a matter of a vast gap of in comprehension between most software experts and dependability experts. For an actual improvement in engineering practice, it is necessary to bridge this gap. This may require more than just goodwill, but research into its economic, cultural and psychological causes and how to deal with them. For dependability minded customers, like the safety critical industries, quality of COTS products is now a major concern. Increasing dependence on software, increases the costs of undependability or of not matching dependability to needs. Some current trends, like that towards using more COTS components, create both opportunities and technical challenges for this progress. Improving the state of art in dependable computing is crucial for the very future of science and technologies of information and communication. Computer failures cost to the society tens of billions of dollars each year, and the cumulative cost of cancelled software developments is of the same order of magnitude.

**NEED OF MORE RESEARCHES**

1. Because of our present inability to produce software, software fault tolerance is and will continue to be an important consideration in software systems.
2. It is expected that software fault tolerance research will benefit from this research by enabling greater predictability of the dependability of software.
3. It would seem that many of design tasks involved in achieving high levels of overall reliability from large and complex hardware/software systems will continue for a long time to require large measures of creative skill and experience on the part of the designers.

4. Software fault tolerance research has drawn more and more focus now a days, as the majority of system defects are shown to be software defects.

5. It is reasonable to assume that the incidence of faults within the software systems will remain a problem for the foreseeable future, despite current advances in software engineering methodologies such as object-oriented design therefore alternative methods need to be investigated; one such method is that of fault most of the mathematics used in reliability engineering has been applied to software reliability engineering.

CONCLUSION:
With the increasing use of computers in commercial applications a strong economic motive emerged: the very visible and costly interruptions of service to business customers. This may have been the major factor that brought fault tolerance to the market place, with several established and new companies developing a variety of fault tolerant computers, both for internal use and for commercial market. A major strength of the dependability concept is its integrative nature that enables to put into perspective the more classical notions of reliability, availability, safety, security and maintainability that are seen as attributes of dependability.

REFERENCES: