Software Reliability and Dependability

Objectives

- What is reliability?
- The problems with understanding software reliability.
- What are the levels of reliability are currently achieved?
- Reliability assurance and measurement.
- What is dependability?
- Dependability assessment.

Why is reliability engineering needed?

- Reliability is being able to deliver usable services, while being able to satisfy the requirements of the system.
- Part of quality assurance.
- There is now an increasing usage of software in many aspects of society today. Especially in areas where the software is safety critical. For example nuclear station and medical systems.
- There is an increasing need to for people to see how much they can depend and rely the software.
- There is a market advantage for companies to demonstrate their reliability to customer.

Problems

- Unreliability is almost always due to design faults.
- Software used to be modelled on a manual process that already existed.
- Software has a tendency to be designed in order to take on tasks which are difficult for older technology.
- Complexity in software, have a trend of increasing.
- It is impossible to do exhaustive testing.
- Evidence of reliability is only obtained after the system has been operating for a long time, but there needs to be an assurance that the system is reliable before they are used.

Levels of reliability (Failure levels)

- The levels of reliability required differs, depending on the software usage e.g. a flight control system will require a higher level of reliability than an office software system.
- The levels are set to be a probability of a failure occurring over time.
- However figures from newer IT industries are harder to obtain.

Examples of levels of reliability achieved

- Flight control systems
  - Were specified to have a required tolerance of 10^{-9} probability of failures per hour. However in practice it was 10^{-7} to 10^{-8}, this required the FAA to produce 'Airworthiness directives'.
- Nuclear safety system
  - Only have a requirement of,10^{-4} probability of failures upon demand (PFD), much lower and less stringent because only called into action upon a dangerous system state.
- AT&T communication network
  - Their telecom network exhibited very high quality of service measures. This was a result due to the inclusion upon component reliability of extensive redundancy, error detection and recovery. An example are the 4ESS switches which ran at $5.7x10^{-6}$ unavailability.

Measurement and Assurance of Reliability 1

- Testing under operational conditions
  - Reliability growth modelling
    - Simulate the software's operational use noting times when failures occur and statically analyzing the data. The faults will then be removed.
    - Limitations
      - It is often difficult to create a test suite that is representative of operational use.
      - This model assumes successful fault removal, however it is possible that removing this fault will introduce new faults.
      - The data collected is quite limited and in order to demonstrate reliability levels the number of tests that would need to be executed to show a confidence in the PFD would be infeasible for a high level of reliability piece of software.

Measurement and Assurance of Reliability 2

- Static Analysis of the software
  - A formal proof or analysis that shows that a class of fault is not present in the software.
  - Limitations
    - The analysis tool can only be used, as long as the complexity of the software is not too great.
    - There is the question of measuring the increase in confidence we have in the software.
    - For example; Malapas Analysis
      - The safety system software of the Sizewell nuclear reactor, showed some problems but they were claimed to have no safety implications. However certain parts of the system due to their complexity defeated the analytic tool.

Measurement and Assurance of Reliability 3

- Structural model of reliability
  - The reliability of the various components of the system is know and therefore the overall system reliability is estimated from the components reliability.
  - This could also be used when Off the shelve components are used, and the reliability in previous systems are used to calculate the overall system reliability.
  - Limitations
    - The data used for obtaining the reliability of the components are often inaccurate and not sufficient in detail.
    - There needs to be a simple model to capture interaction between the components in order for it to be manageable.
    - The difficulty in estimating the effect of redundancy, due to the order complexity.
    - The reliability calculated previously for components, may not reflect the reliability of the context usage.

Dependability

- This includes; reliability, safety, security and availability, but the main topic here is reliability.

Awareness of Dependability

- Hindered by economic, cultural, psychological matters & public perception.
  - Y2K, when very few computer systems had failed the 'public' attitude was that a huge expenditure was wasted.
- Design culture
  - Professional developers often lack education in dependability, both academic and from work experience.
  - The RISKS Archive (www.csl.sri.com/risksinfo.html), this is a useful source for common problems.
- Management culture
  - Management do not usually have appropriate background knowledge.
  - Management claiming that the product has 10^{-9} PFD too easily with no/little evidence.

User Focused Dependability

- Too often, reliability is done to compliance of written specifications of the system, and not defined as its effect on the user.
- Measuring the reliability of individual components in a component based system is a convenient step towards assessing the users dependability requirements, but this is not the goal of reliability engineering.
- User-Orientated requirements have many dimensions.
  - Staggered bounds in failure rates i.e.:
    - Telephone companies with multiple targets for freq. of dropped calls or freq. of total duration of outages.
- Users can do the 'human factor' to the system through operation and maintenance of the system.

Dependability Assessment 1

- Validation
  - Design cultures often ignore the need for validation.
- Failure Prevention
  - Proof that certain event can't happen.
  - Separation of sub-systems, memory protection prevents interference and propagation between different processes.
    - Modern airliners require guaranteed separation in the powerful software services that fly/run them.

Dependability Assessment 2

- System Monitoring
  - Reporting of errors and faults, now starting to be more common.
    - Windows XP, you can annoy Microsoft every time an application crashes. Many other developers use this or are rolling it out.
    - It provides valuable real world usage and from this better test cases can be created.
  - Technical/Hardware monitoring.
    - This is the oldest form of monitoring as a poor computer system will cause any software to perform worse.

Judgment & Decision Making 1

- Engineering Approach
  - Often industry can't afford the extra effort on reliability. This view is limiting.
  - In achieving dependability, matching means to dependability targets is required.
  - Rational decisions must be taken taking into the affecting factors.
    - Overkill is a sensible strategy if the degree of cost of failure out ways the cost of overkill.
- Choice of Process
  - Most proposed methods for improving software reliability have intuitive appeal.
    - But measuring their whether the advantages are real and the cost is justified is seldom attempted.
  - The exploiting of available data to extract useful conclusions without overgeneralization poses demanding analytical problems.
  - However there is the need to base decisions on general laws that determine effectiveness of methods rather than on raw data.

Judgment & Decision Making 2

- Formalism & Judgment
  - For the modest dependability requirements, the techniques now available would be supported without process evidence.
    - However industry is reluctant to apply the effort required to test and data collect.
  - For some extreme requirements using all available evidence becomes essential.
    - Adherence to a process is usually a main part of the evidence.
    - But this is not sufficient for believing the software satisfies dependability requirements.
  - Rigorously assessing reliability has two main difficulties.
    - Scarcity of knowledge. The actual effect of process on dependability.
    - The difficulty of understanding how the evidence we need to use should be combined into an overall evaluation.
  - Matching formal descriptions to real world requires judgement and prudence, this can't be formalized.

Very Large-Scale Systems

- Often built using unplanned aggregation of computer-based systems. This creates new concerns.
  - Modelling complex systems without losing the ability to understand and solve the models.
  - Avoid the risk factors which previously negligible.
  - Detailed models are useless.
- Practices are often browed form other real world large-scale systems.
  - Communication/electrical distribution networks.
  - Modelling the evolution/propagation of disease breakouts.
- Grow without a general design.
- Emerging properties determine the user observed dependability.
- Not limited to the software side.
  - Economic, legal, environment & technological factors/developments.

Humans

- Cognitive workload & mental models.
  - Attention by designers is a clear requirement in most software developments.
- Strengths and weaknesses of human operators with machinery compared to automated machinery.
  - As yet there are no strong software engineering practices to define this.
- Cognitive psychology applied to the mechanisms of human error.
  - Understanding the effects of software processes.

Reference

- Software Reliability Guidebook.
  - Robert L. Glass (1979).
- Software Reliability and Dependability: a Roadmap.
  - Bev Littlewood & Lorenzo Strigini

Conclusions

- Approach to design must include dependability aspects.
  - Has been slow to adopt but is necessary for more efficient decisions.
- Increasing dependence on software.
  - Cost of undependability increasing.
  - Not matching the dependability to the needs.
- COTS components.
  - The trend towards this creates opportunities and challenges to overcome.
- Education of users & developers in better communication.
- Research challenges.
  - Learning more on the effects of different practices of achieving dependability.
  - Learning to better the judgement and decision making process.