

## Software Reliability

### Outline of Lecture

1. Discussion of reliability
2. Reliability Models
3. Tools and Methods
4. Relationships with other attributes
5. Conclusion

## Definition of reliability

- Defined as “the probability that a system functions without failure for a specified time or number of natural units in a specified environment” – Musa [8]
- A failure refers to an incorrect output caused by an anomaly in the source code.
- A natural unit refers to a unit other than time
  - E.g. Pages of output, transactions, etc.

## Why is reliability important?

- Reliable software → customer satisfaction
  - Goal is to adhere to the user’s requirements
  - Unreliability may cause deaths or destruction
- Reliable software → success for the company
  - Satisfied customers become returning customers
  - Reliability saves time and money

## How did reliability become so important?

- Advanced technology has allowed software to permeate our lives
  - Comfortable and convenient lifestyle
  - Used increasingly in medical and military systems
- Software is developed by humans
  - We are not perfect
  - There is a need to strive for perfection
- Consumers have high expectations for software
  - Realized the power of software
  - A huge market has developed
- Therefore...
  - Reliability must be taken seriously to ensure the safety and satisfaction of the public, along with the success of the company

## Is reliability always a top priority?

- NO
  - Other qualities may be more important to the developer
    - E.g. Maintaining a scheduled release
- Does this mean that the product is unreliable?
- NO
  - It may simply be less reliable
  - There may be no noticeable differences in quality to the user

## Does reliability always guarantee company success?

- NO
  - Software success depends on the use of the product
  - Software with faults can still become popular
    - e.g. Microsoft
- Software that does not demand reliability
  - Video games
  - Home applications
- Software that does demand reliability
  - Military defense systems
  - Hospital patient systems

## How is reliability addressed?

- Reliability can be addressed in two major practices
  - Models
    - Predict the reliability of a system based upon its failure data
  - Tools
    - Implement reliability models using software

## Reliability models

- Reliability models are mathematically intense calculations that use data from the software to determine the reliability.
  - Some well known models are:
    - Jelinski-Moranda Model
    - Shooman Model
    - Musa Model
    - Littlewood-Verrall model
    - Goel-Okumoto Model
    - Musa-Okumoto Model

## Jelinski-Moranda Model

- Assumes faults before testing which will probably cause faults during testing
- Once fault found; removed with certainty
- Uses a hazard rate which is proportional to number of faults remaining
- Shooman Model is very similar to the Jelinski-Moranda Model except uses a failure rate

## Musa Model

- Also known as the "Execution Time Model"
- Uses execution time instead of calendar time
- Assumptions
  - There are no failures at the beginning of testing
  - "The failure rate will decrease exponentially with the expected number of failures experienced"[8]

## Models can be classified into three groups

1. Times between failures models
  - Time between  $i-1$  and  $i$ th failure
2. Failure count models
  - number of failures in specified time
3. Fault seeding and input domain models
  - Faults "seeded" into program
  - Time independent

## Steps in determining which model to use

- Step 1: Study software failure data
- Step 2: Choose model
- Step 3: Obtain estimates of model parameters
- Step 4: Use the model by inserting estimated values
- Step 5: Conduct evaluation to determine how well it works with system
- Step 6: compute important numbers
- Step 7: use the model to make decisions

## NASA and reliability models

- There are 11 steps defined by the AIAA *Recommended Practice for Software Reliability*. NASA uses 7 of them[17]
  1. Characterizing the operational environment
  2. Selecting tests
  3. Selecting models
  4. Collecting data
  5. Estimating parameters
  6. Validating models
  7. Performing analysis

## ...Dealing with selecting models.

- When dealing with the selecting models there are seven criteria
  1. Predictive validity
  2. Ease of parameter measurement
  3. Quality of assumptions
  4. Capability
  5. Applicability
  6. Simplicity
  7. Insensitivity to noise

## What are the advantages of using a software tool?

- Simplifies the reliability models
  - Software reduces tedious calculations and automatically graphs results
  - Software reduces the chance of human error
- Reduces time spent on reliability evaluation
  - Sometimes nearly half of software development is spent on testing
  - Higher efficiency leads to lower overall costs
- Assists in making critical decisions
  - Answers many questions, such as:
    - How much time should be spent testing this module?
    - Will we meet the deadline at this testing rate?
    - Have we met the appropriate level of reliability?

## What are the disadvantages of using a tool?

- Initially may be difficult to use
  - Requires prior experience with the reliability model
  - Mathematical knowledge required
- Many tools only address reliability in the testing phase
  - May be too late and too costly to change a requirement
- Not all reliability models fit the testing data
  - Experience needed to determine a suitable model
  - Inappropriate models will give misleading results

## In which phases can tools and methods be used?

- System testing phase
  - Evaluates software reliability on the system as a whole or as individual units
- Component integration phase
  - Addresses the reliability of relationships between system modules
- In all phases of software lifecycle model
  - Uses methods and planning to assess and track reliability in every phase

## Addressing reliability in the testing phase

- 3 resources are used to demonstrate reliability evaluation in this phase
  - FIABLOG: a software reliability tool and its application to space projects by Vallee, F. Ragot, A
    - Provided 3 reliability models
    - 3 basic steps to evaluating reliability
      - Observe trends in the failure process
      - Use diagrams to select the appropriate model
      - Allow the tool to predict reliability in terms of failures to come, failure rate, and mean-time-to-failure

## Addressing reliability in the testing phase – cont.

- Software reliability assessment tool based on object-oriented analysis and its application by Yamada, S.; Kimura, M.;
  - Offered 7 reliability models
  - 3 step process
    - A model is chosen
    - Determines if the model is appropriate by using the Kolmogorov-Smirnov test
    - The reliability measures are estimated using the reliability tool

## Addressing reliability in the testing phase – cont.

- Local exhaustive testing: a software reliability tool by Wood, T.; Miller, K.; Noonan, R.;
  - Proposed a technique called "local exhaustive testing" for uncovering faults in less time
  - To utilize this method, "critical points" must be determined
    - Critical points are points in the software where inputs have a higher probability of uncovering a fault
  - Using these points can find faults quicker and save overall testing time and costs

## Addressing reliability in the component integration phase

- 3 resources are used to demonstrate this idea
  - Developing a Low-Cost High-Quality Software Tool for Dynamic Fault-Tree Analysis by Dugan, J.; Sullivan, K.; Coppit, D.
    - Utilized fault-trees to represent the system failures visually
    - 2 types of sub-trees
      - Dynamic – determines if the order that components fail can lead to a system failure
      - Static – determines which components can potentially affect the other components through failures

## Addressing reliability in the component integration phase – cont.

- The support tool for highly reliable component-based software development by Matsumoto, M.; Futatsugi, K.;
  - Two main features of this tool:
    - Automatic refinement verification
    - Automated connector generation

## Addressing reliability in the component integration phase – cont.

- Framework of a software reliability engineering tool by Sanyal, S.; Shah, V.; Bhattacharya, S.
  - Two primary features of this tool:
    - Event-control flow graphs
    - Program dependency graphs

## Addressing reliability in all phases of the lifecycle

- 2 research papers were used:
  - Developing more reliable software faster and cheaper by Musa, J.D.
    - Introduced a reliability method called Software Reliability Engineering
    - 6 important steps:
      - List all of the components of the system that must be tested independently
      - Define the "just right" level of reliability for the product
      - Develop operational files
      - Use the operational files to prepare for testing
      - Conduct tests in order to determine failures
      - Prepare for the software's release by tracking reliability growth

## Addressing reliability in all phases of the lifecycle – cont.

- SREPT: **software reliability estimation and prediction tool** by Ramani, S.; Gokhale, S.; Trivedi, K.
  - Offers two approaches to predicting software reliability
    - Black-box-based approach
    - Architecture-based approach

## What software quality attributes are enhanced by reliability?

- Security
  - Defined as: how difficult it is for users to gain access to classified information
  - High reliability  $\longrightarrow$  higher security
  - Must protect legitimate users from corrupting the system
  - Must not allow unauthorized users access
    - May take advantage of system's unreliability
- Safety
  - Defined as: preventing the software from harming the user
  - High reliability  $\longrightarrow$  safer system
  - Unreliability may cause a system to perform undesired operations
  - Unsafe systems could be lethal
    - Military systems
    - Hospital systems

## What software quality attributes improve reliability?

- Testability
  - Defined as: how much effort must be put into testing
  - High testability  $\longrightarrow$  high reliability
  - Testing makes up a large portion of reliability testing
  - Less effort in testing will result in a quicker evaluation and release of the product
- Manageability
  - Defined as: how well the system support changes
  - A well-managed system  $\longrightarrow$  high reliability
  - Reliability assessment requires many changes
  - Smooth transitions  $\longrightarrow$  less time and money spent evaluating reliability

## What did we learn about this topic as a team?

- Evaluating reliability is a very time-consuming activity, however its benefits far outweigh the overall costs
  - Getting it correct the first time saves time and money in the long run
- Assessing reliability has developed into a science
  - Formal methods and models have been created to assess the failure data
- Software tools can be very useful, if you know how to use them, in predicting reliability
  - The difficulties in the models are simplified

## Conclusion

- To summarize
  - Reliability has recently become necessary and desired in many software systems
  - Since people are not perfect, much time must be spent on evaluating and increasing reliability
  - Software tools and models facilitate this daunting task
  - Certain software quality attributes enhance reliability, others are improved by it

## Resources

- [1] Dugan, J; Sullivan, K; Coppit, D.; **Developing a Low-Cost High-Quality Software Tool for Dynamic Fault-Tree Analysis**. IEEE Transactions On Reliability, Vol. 49, No. 1, March 2000. Page(s): 49-59 -10 Pages
- [2] Ehrlich, W. K.; Emerson, T. J.; **Modeling software failures and reliability growth during system testing**, Proceedings of the 9th international conference on Software Engineering March 1987.
- [3] Gogeva-Popstojanova, K.; Mathur, A.P.; Trivedi, K.S.; **Comparison of architecture-based software reliability models**, Software Reliability Engineering, 2001. ISSRE 2001. Proceedings. 12th International Symposium on, 2001 Page(s): 22 -31.
- [4] Kadifeli, Fedon; **A Reliability Model for a Large-Scale Software System**, Master of Science Thesis Approved by Dr. Oguz Tosun, 1987, Pages 1-94
- [5] Matsumoto, M.; Futatsugi, K.; **The support tool for highly reliable component-based software development** Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific, 2000 Page(s): 172-179 -7 Pages
- [6] Moawad, Ramadan; **Comparison of concurrent software reliability models**, Proceedings of the 7th international conference on Software engineering March 1984

## Resources: continued

- [7] Mohanty, Siba N.; **Models and Measurements for Quality Assessment of Software**, ACM Computing Surveys (CSUR) September 1979 Volume 11 Issue 3.
- [8] Musa, J.D.; **Developing more reliable software faster and cheaper** Engineering of Complex Computer Systems, 1999, ICECCS '99, Fifth IEEE International Conference on, 1999 Page(s): 162-176 -14 Pages
- [9] Musa, J. D.; Okumoto, K.; **A logarithmic poisson execution time model for software reliability measurement**, Proceedings of the 7th international conference on Software engineering March 1984.
- [10] Ohba, M., Chou, X. M.; **Does imperfect debugging affect software reliability growth?**, Proceedings of the 11th international conference on Software engineering May 1989.
- [11] Ramani, S.; Gokhale, S.; Trivedi, K.; **SREPT: software reliability estimation and prediction tool**. Performance Evaluation 39(1-4). 2000. Pages: 37-60 -23 Pages
- [12] Sanyal, S.; Shah, V.; Bhattacharya, S.; **Framework of a software reliability engineering tool** High-Assurance Systems Engineering Workshop, 1997., Proceedings , 1997 Page(s): 114-119 -5 Pages
- [13] Vallee, F.; Ragot, A.; **FIABLOG: a software reliability tool and its application to space projects** Software Reliability Engineering, 1994, Proceedings., 5th International Symposium on , 1994 Page(s): 296-302 -6 Pages

## Resources: continued

- [14] Wallace, D.R.; **Practical software reliability modeling** Software Engineering Workshop, 2001. Proceedings. 26th Annual NASA Goddard, 2001. Page(s): 147-155 -7 Pages
- [15] Wood, T.; Miller, K.; Noonan, R.; **Local exhaustive testing: a software reliability tool**. ACM Southeast Regional Conference, Proceedings of the 30th annual conference on Southeast regional conference, Raleigh, North Carolina, 1992. Pages: 77-84 -7 Pages
- [16] Yamada, S.; Kimura, M.; **Software reliability assessment tool based on object-oriented analysis and its application**. Annals of Software Engineering 8. 1999. Pages: 223-238 -15 Pages
- [17] American Institute of Aeronautics and Astronautics, **Recommended Practice for Software Reliability**, ANSI/AIAA R-013-1992, February 1993