SCHULICH School of Engineering

UNIVERSITY OF CALGARY

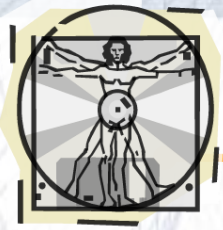
SENG 637
Dependability and Reliability of Software Systems

Chapter 10: Reliability Assessment Tools and Techniques

Department of Electrical & Software Engineering, University of Calgary

B.H. Far (far@ucalgary.ca)

http://people.ucalgary.ca/~far



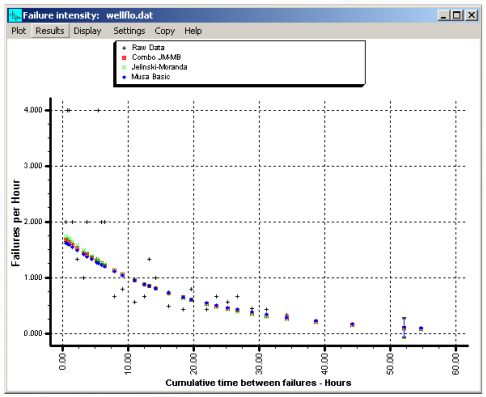
Contents

- Reliability assessment tools
- System reliability assessment
- Assessment criteria:
  - Certification testing using RDC
  - Reliability growth
  - Zero failure testing
  - Special cases

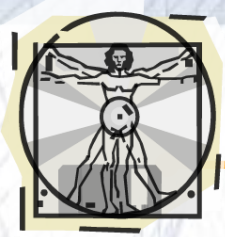
Reliability Assessment Tools

Failure data → **Reliability Assessment Tool** → **Output info**

Time(s)	Cumulative Failures	Failures in interval
30	2	2
60	5	3
90	7	2
120	8	1
150	10	2
180	11	1
210	12	1
240	13	1
270	14	1

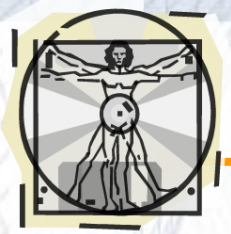


- Failure intensity plot
- Time-to-failure plot
- Reliability plot
- etc.



Tasks Handled by SRE Tools

- Collecting failure and test time information
  - Calculate MTTF, reliability, etc.
- Or
  - Select a reliability model; calculate estimates of model parameters; test to fit a model against the failure data
  - Use model parameters to make predictions of remaining failures, time to release, etc.
- Using the results for decision making



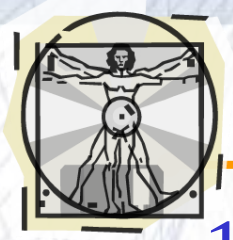
Input Data Specification /1

- All of the SRE tools use one of two basic types of failure data:
  - time-domain data

    (*i.e.*, time-between-failures data)

  - interval-domain data

    (*i.e.*, failure-count data)



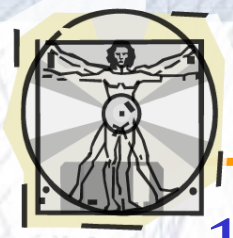
Input Data Specification /2

1) Time of failure
2) Time interval between failures
3) Cumulative failure up to a given time
4) Failures experienced in a time interval

(2) Is a.k.a Inter-failure times data

Time based failure specification

Failure no.	Failure times (hours)	Failure interval (hours)
1	10	10
2	19	9
3	32	13
4	43	11
5	58	15
6	70	12
7	88	18
8	103	15
9	125	22
10	150	25
11	169	19
12	199	30
13	231	32
14	256	25
15	296	40



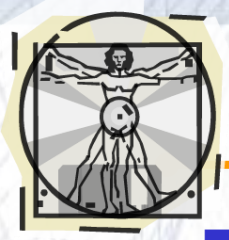
Input Data Specification /3

1) Time of failure
2) Time interval between failures
3) Cumulative failure up to a given time
4) Failures experienced in a time interval

Failure based failure specification

Time(s)	Cumulative Failures	Failures in interval
30	2	2
60	5	3
90	7	2
120	8	1
150	10	2
180	11	1
210	12	1
240	13	1
270	14	1

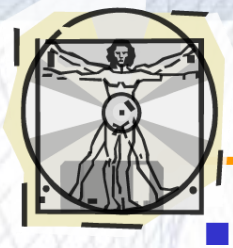
(4) Is a.k.a failure count data



SRE Tools

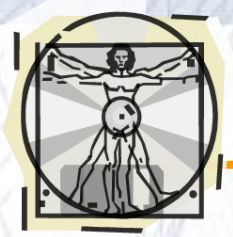
- CASRE
- SoftRel
- SMERFS
- SoRel
- SRMP
- ProConf
- Relex
- MEADEP (MEAsure and DEPendability)

etc.



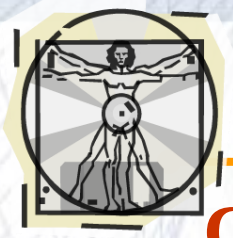
SRE Tools (cont'd)

- **ACARA II:** Availability, Cost, And Resource Allocation, Version 2 *(no charge per license)*
- **ARAM:** Automated Reliability/Availability/Maintainability, Version 2.0 *($400 source code license)*
- **ETARA:** Event Time Availability, Reliability Analysis *($200 source code license)*
- **GO:** Graphics Oriented Program *($150 source code license)*
- **HARP:** Hybrid Automated Reliability Predictor, Version 7.0 *($500 source code license, for Unix or PC)*
- **HARPO:** Hybrid Automated Reliability Predictor Output Graphics Display *($150 source code license)*
- **SPRPM:** Software Problem Report Metrics Program *(no charge per license, requires EXCEL)*



SRE Tools: SMERFS /1

- Statistical Modeling and Estimation of Reliability Functions for Software
- SMERFS is a public-domain software package designed and implemented at the Naval Surface Warfare Center
- SMERFS is a program for estimating and predicting the reliability of software during the testing phase
- The body of code is in Fortran



SRE Tools: CASRE /1

Computer-**A**ided **S**oftware **R**eliability **E**stimation Tool

- CASRE is copyrighted by NASA
- CASRE is a PC-based tool that was developed in 1993 by the Jet Propulsion Laboratories to address the ease-of-use issues of other tools
- CASRE requires the Windows operating environment
- It has a pull-down, menu-driven user interface and uses the same model library as the SMERFS tool with the additional feature of allowing linear combinations of models to create new ones at the user's discretion
- Four combined models are permanently available in CASRE
- CASRE ver. 3.0 is available (http://www.openchannelsoftware.org/discipline/Reliability_Analysis)

SRE Tools: CASRE /3

- CASRE provides operations to transform or smooth the failure data
- users can select and/or define multiple models for application to the data and make reliability predictions based on the best model

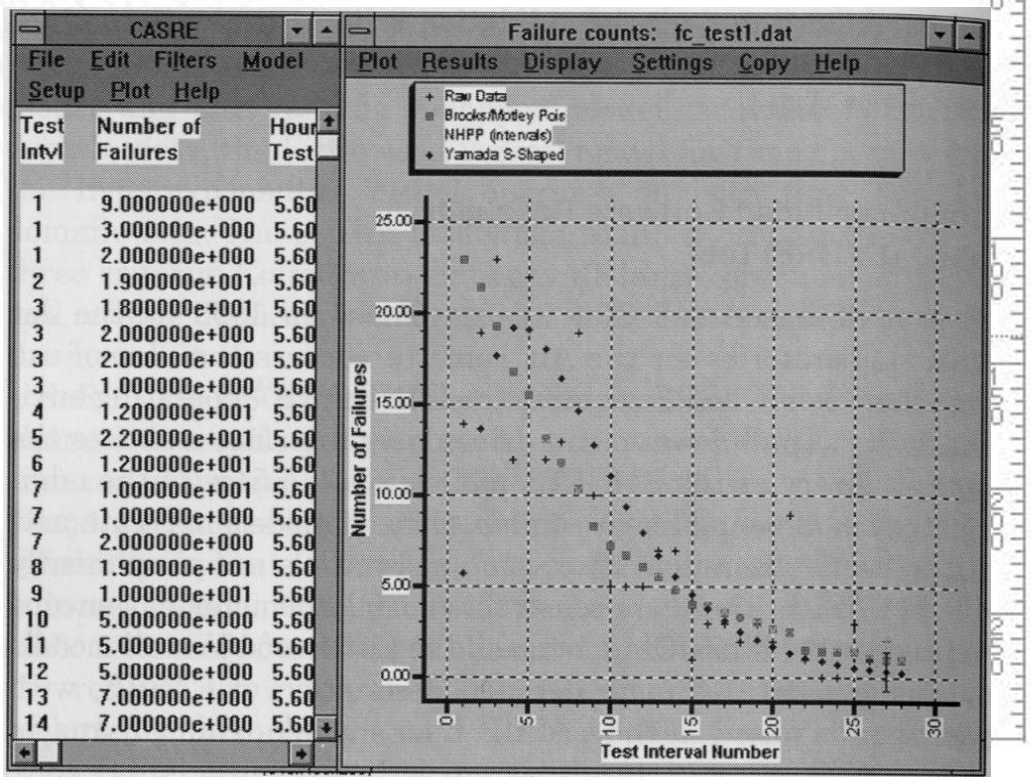
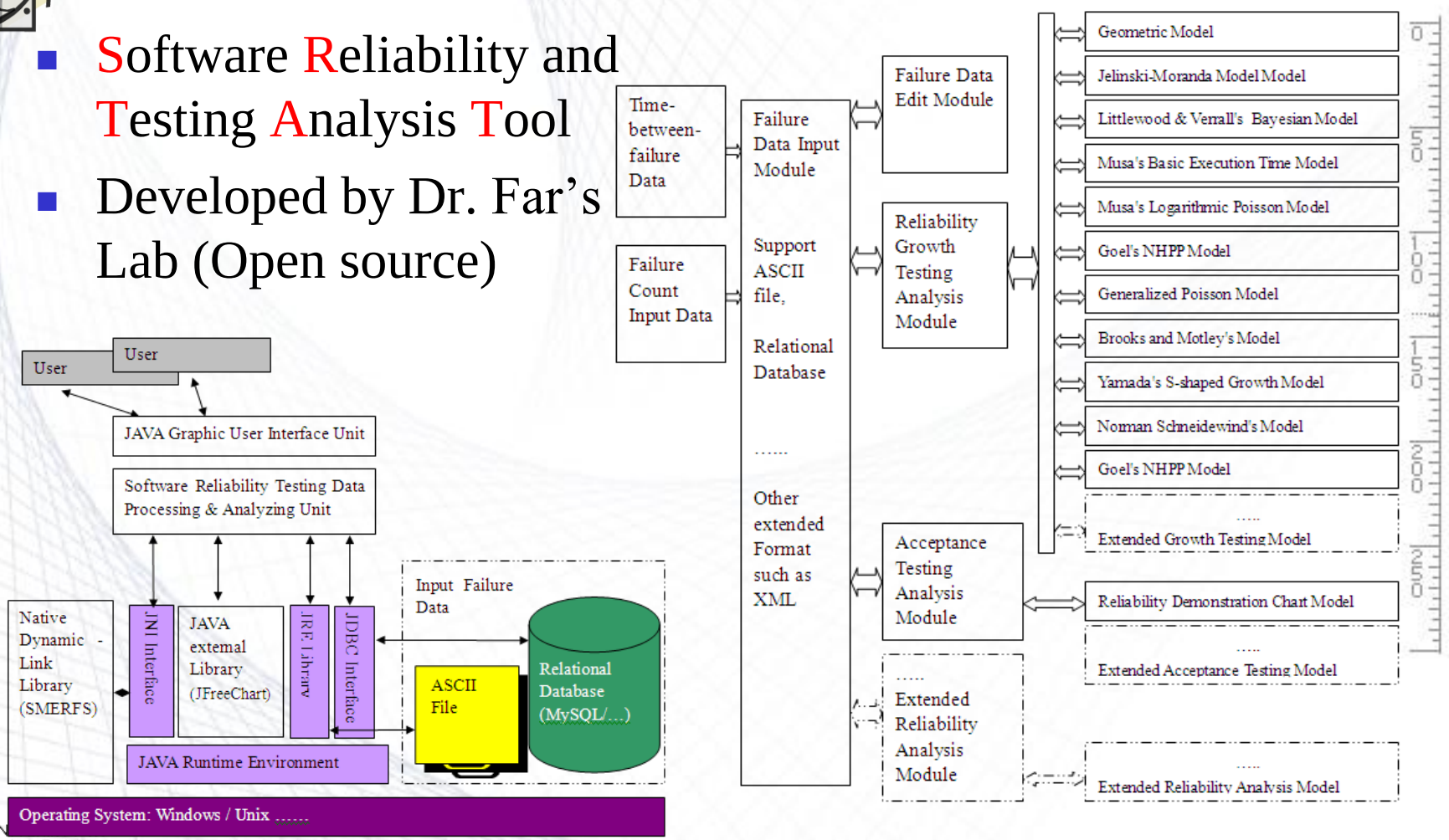


Figure from SRE Handbook

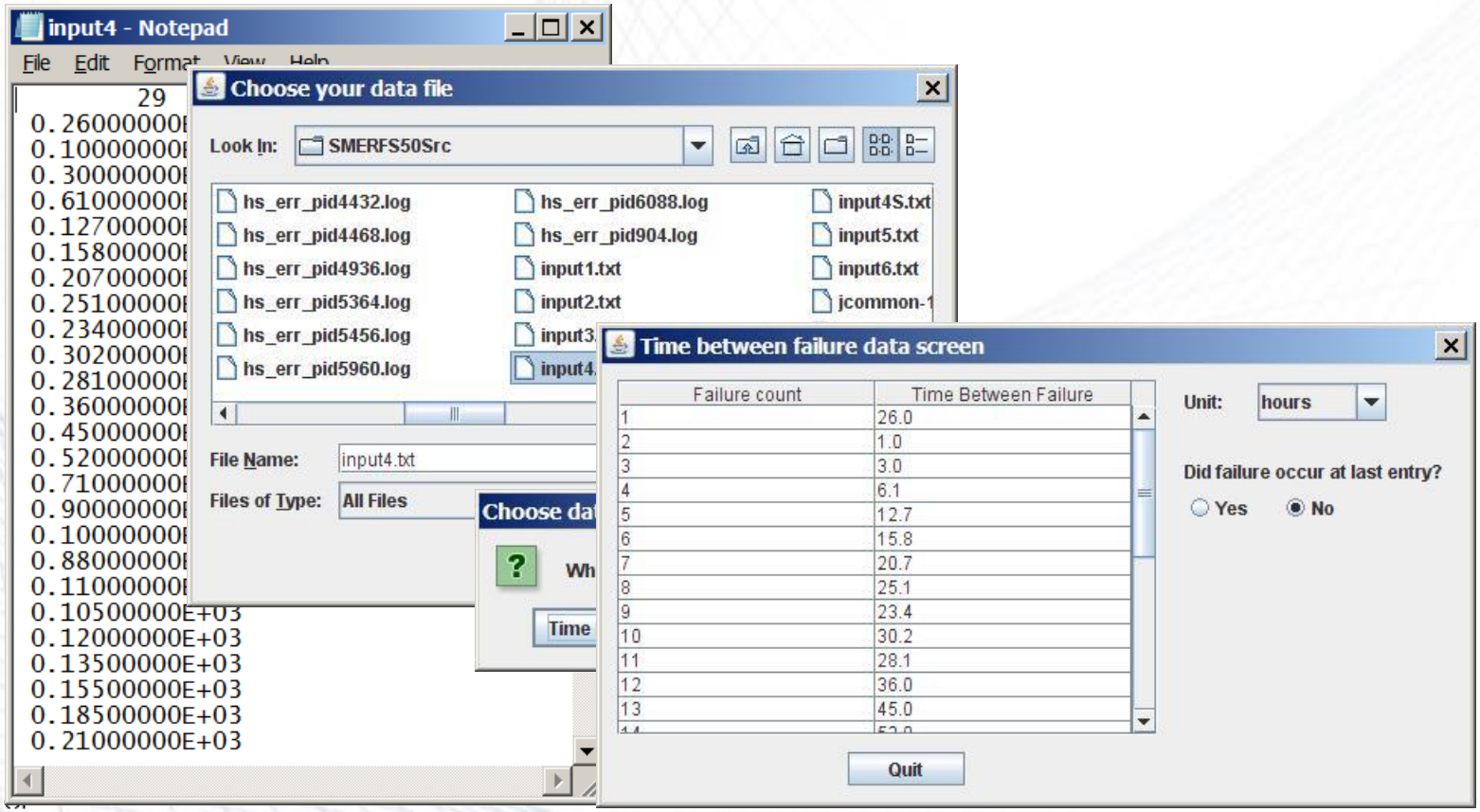
SRTATA

- Software Reliability and Testing Analysis Tool
- Developed by Dr. Far's Lab (Open source)



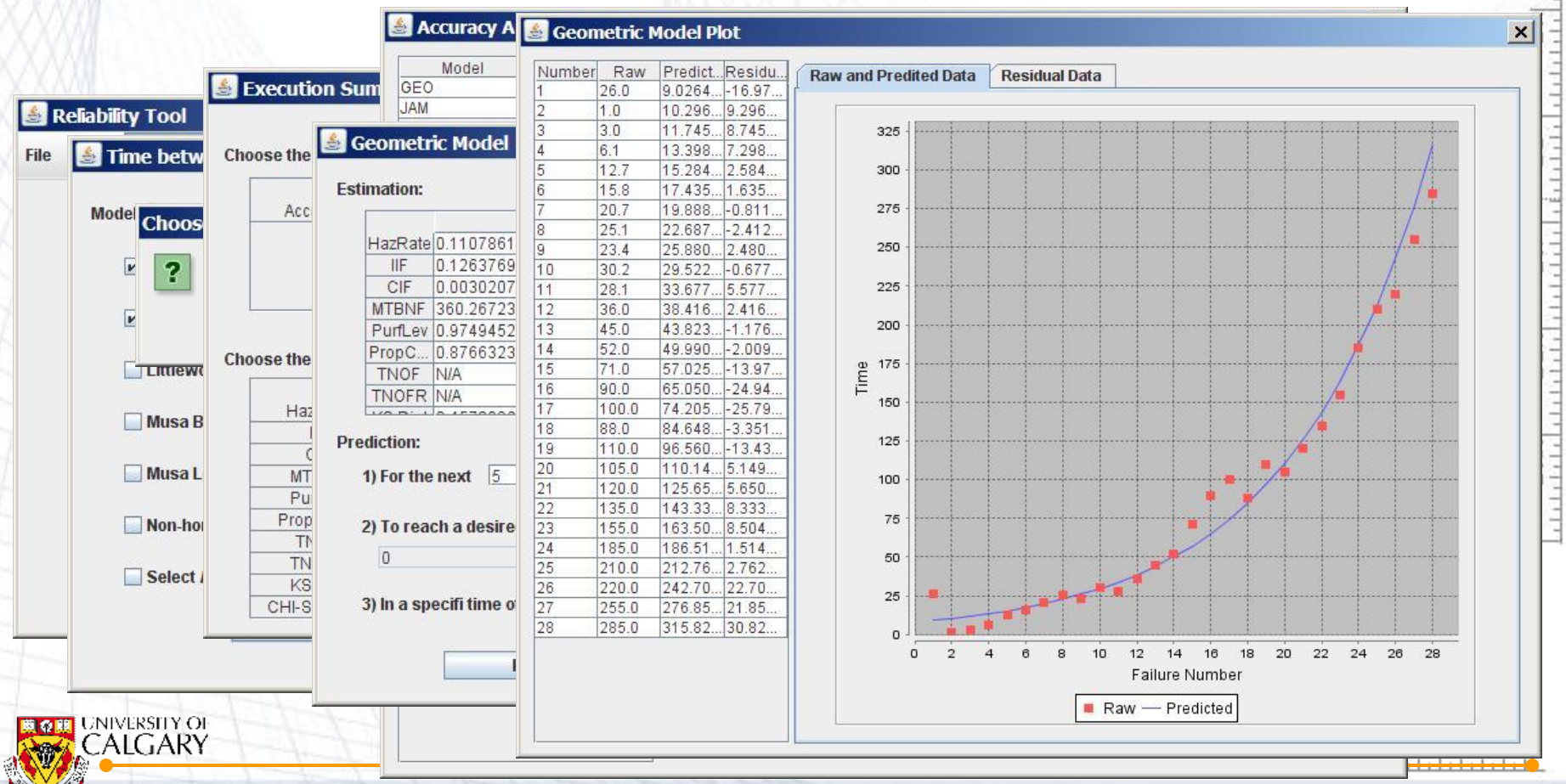
SRTATA /2

- Input and display: time-between-failure data



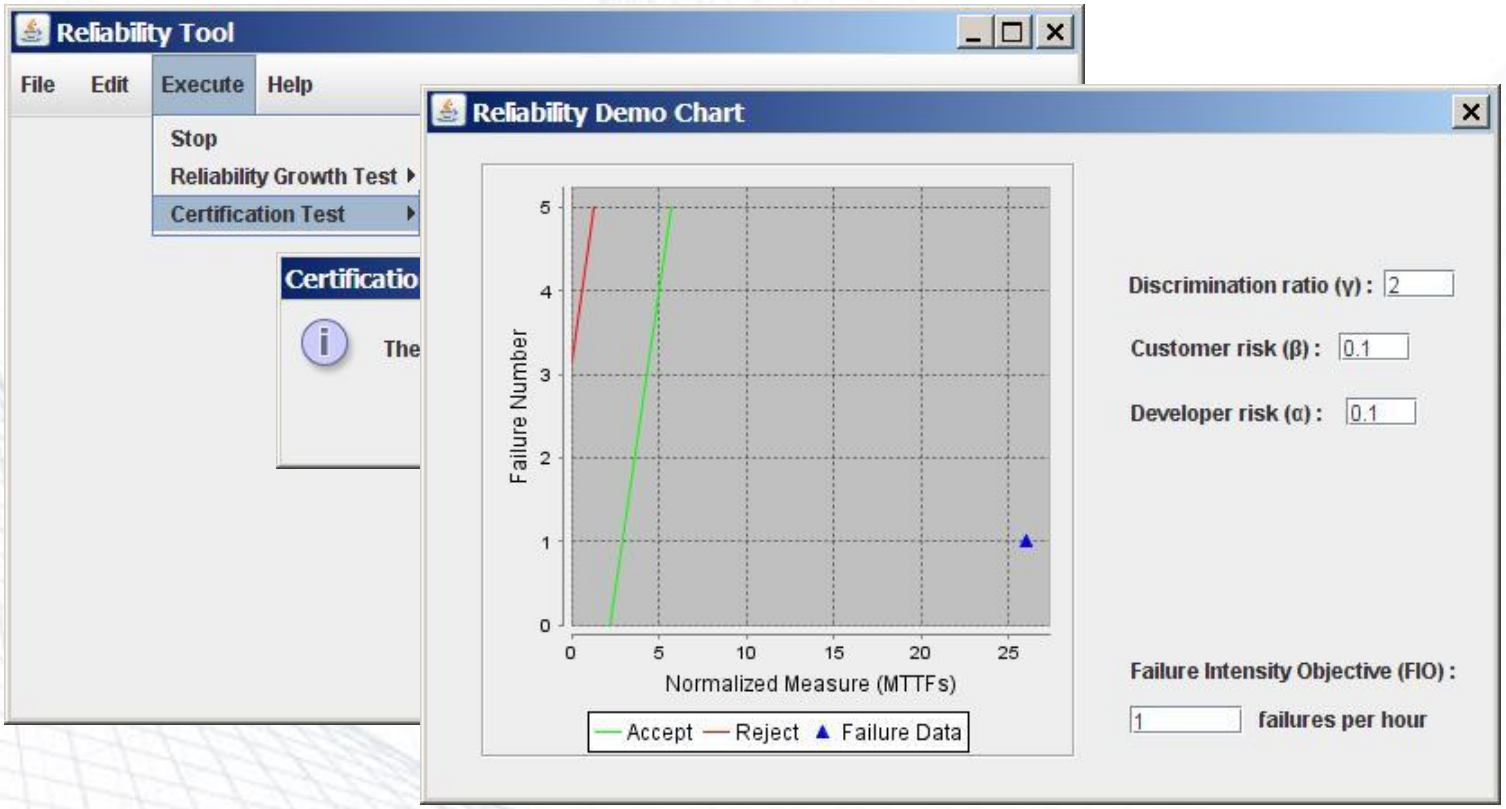
SRTATA /3

- Reliability Growth Analysis: Time-between-failure



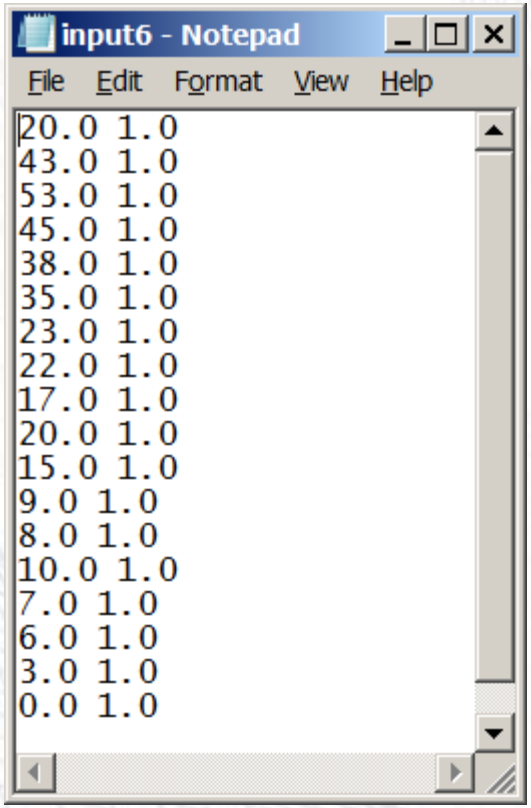
SRTATA /4

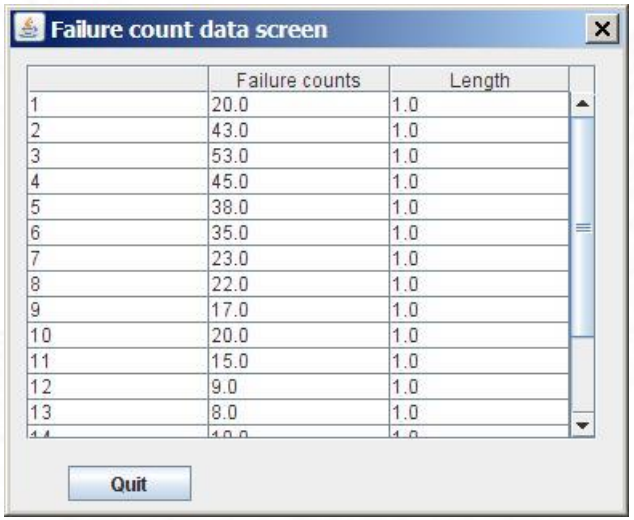
- Reliability Acceptance Analysis



SRTATA /5

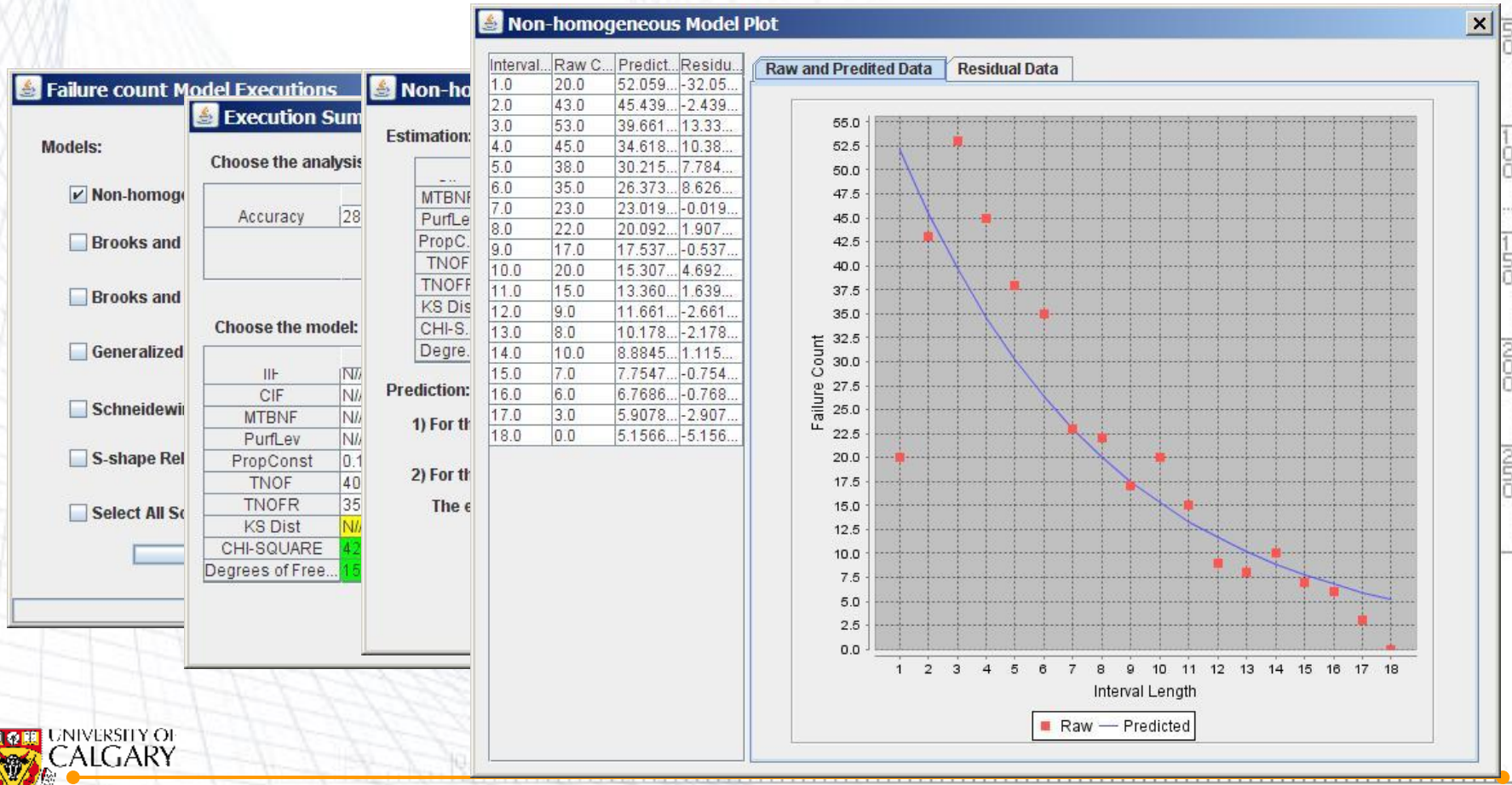
- Input and display: failure count data





SRTATA /6

- Reliability Growth Analysis: failure count



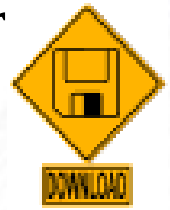
More Info

- SRE tools repository:
  - Center for Reliability Engineering at the Univer Maryland

    http://www.enre.umd.edu/tool.htm
  - Open Channel Foundation

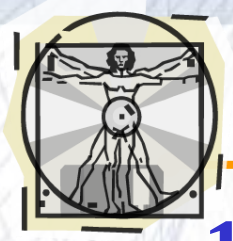
    http://www.openchannelsoftware.org/discipline/Reliability_Analysis
  - ReliaSoft (commercial software)

    http://www.reliasoft.com/products.htm



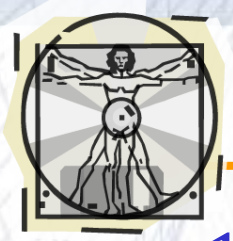
Section 2

System Reliability Assessment



Types of Assessment

1. **Decisions related to certification test**
   - Accept/reject an acquired component
   - Accept/reject a system, OS, hardware
   - Quality assessment of a system, software, etc.
2. **Decisions related to reliability growth test**
   - Tracking bugs in pre-release
   - Guiding software testing process
   - Releasing the product
3. **Decision related to adequacy of tests**

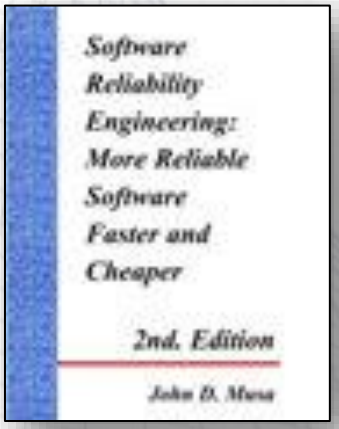


Types of Techniques

1. Reliability Demonstration Chart ← based on inter failure times only and target failure rate (or MTTF)
2. Reliability growth analysis ← based on inter failure times and/or failure count and target failure rate (or MTTF)
3. Zero failure testing ← based on failure density

UNIVERSITY OF CALGARY

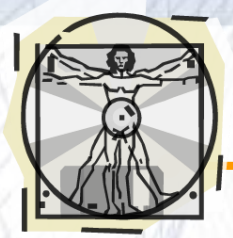
Section 3



SRE Book ch.6

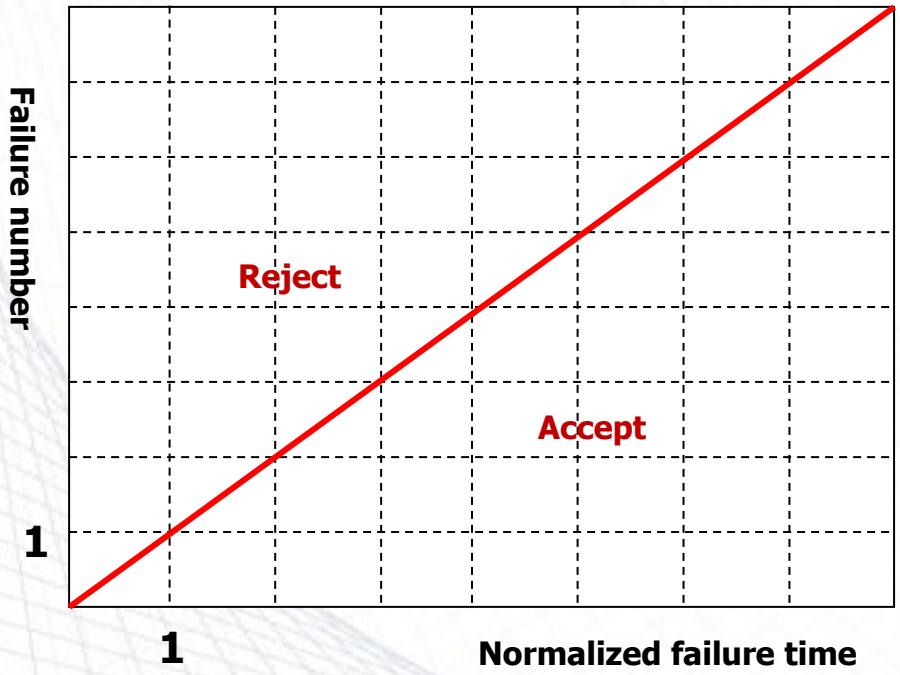
Assessment Criteria:

Certification Testing Using RDC



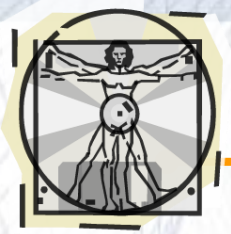
Using No Risk

- If a system passes its target MTTF without failure, it is acceptable



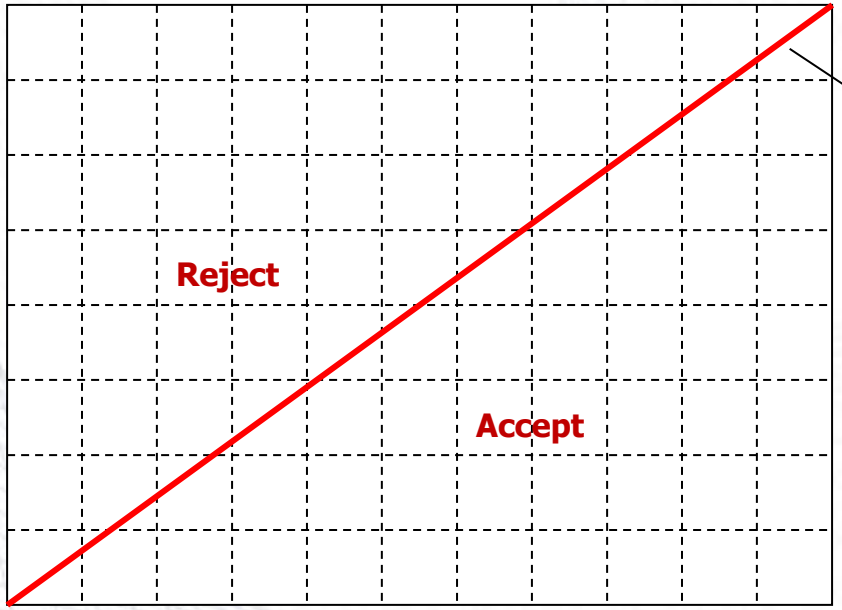
For a multi-failure system

- If all failures experiences are in the accept region the system is good to go
- Usually difficult to satisfy this condition because failures happen randomly



Introducing Product Risk

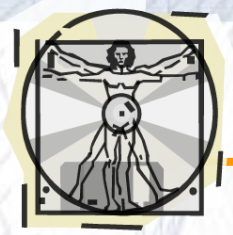
- Risk related to the measured entity ← measurement risk; main risk component: γ



$$\frac{-\ln\gamma}{1-\gamma} \qquad 1<\gamma$$

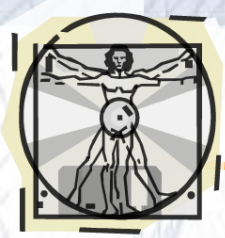
Decreasing γ brings the line closer to vertical: less risk makes the product more acceptable

Increasing γ brings the line closer to horizontal: more risk makes the product more rejectable



Transaction Risks

- Risk related to the measured entity ← aka. product risk; measurement risk; main risk component: γ
- In a multiple stakeholder transaction involving supplier and purchaser:
  - Risk in assessing the entity which is actually wrong to be right (← purchaser's risk: β)
  - Risk in assessing the entity which is actually right to be wrong (← supplier's risk: α)
- RDC uses these 3 risks together



Risk Parameters Involved

- **Discrimination ratio ($1<\gamma$):** Acceptable error in estimating failure intensity
- **Customer risk ($0<\beta<1$) :** Probability that the developer is willing to accept of falsely saying the target failure rate is met (i.e., ***acceptance***) when it is not
- **Developer risk ($0<\alpha<1$) :** Probability that the developer is willing to accept of falsely saying the target failure rate is **not** met (i.e., ***rejection***) when it is

Reliability Demo Chart /1

- An efficient way of checking whether the target failure rate (λ_F) is met or not
- Based on collecting failure data at time points
- Vertical axis: failure number (n)
- Horizontal axis: normalized failure data (Tn), i.e.,

  failure time $\times \lambda_F$ or

  failure time / MTTF

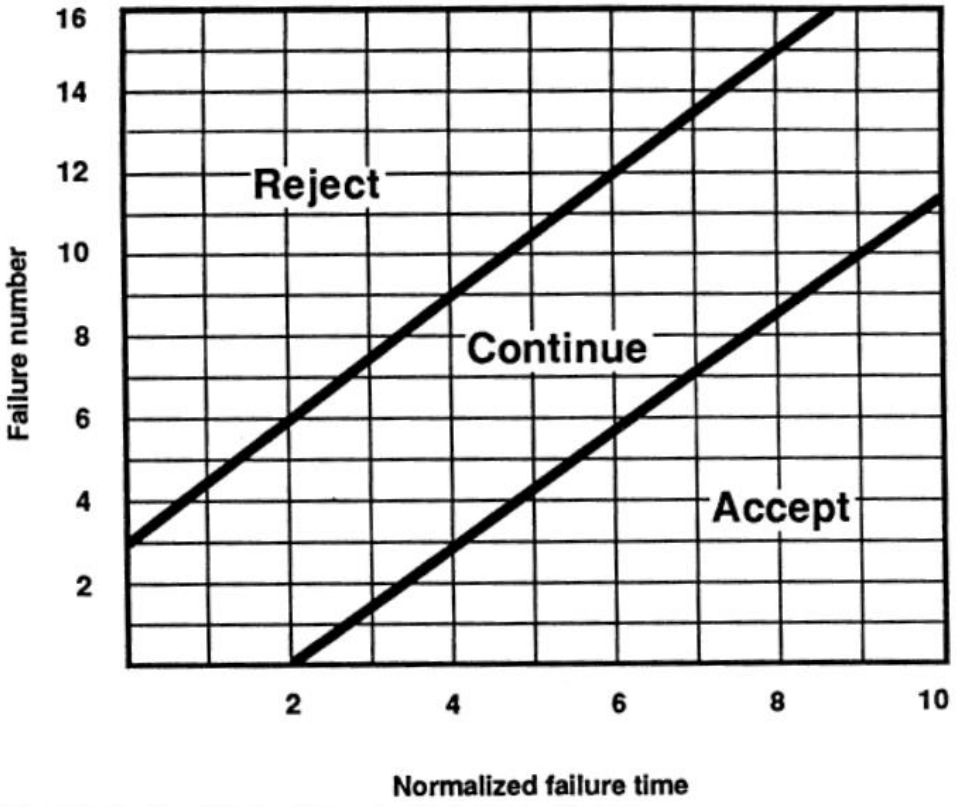
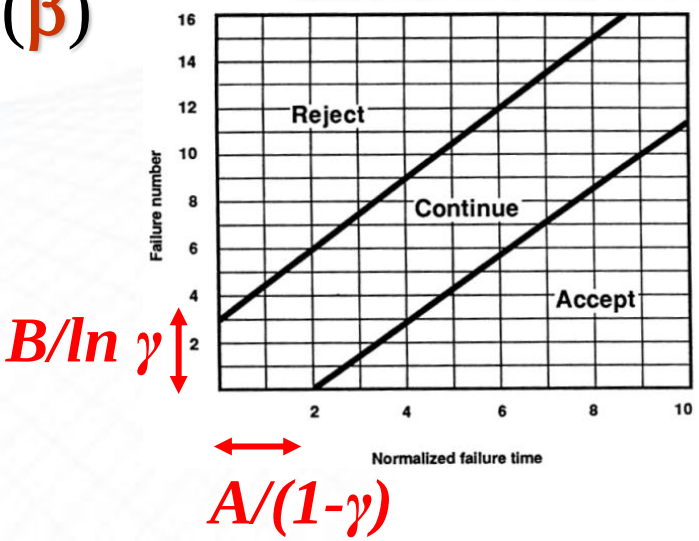


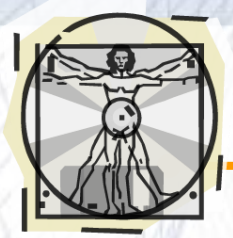
Figure from Musa's Book

Reliability Demo Chart /3

$$A = \ln \frac{\beta}{1-\alpha} \qquad B = \ln \frac{1-\beta}{\alpha}$$

- ***A*** changes rapidly with *customer risk* (β) but very slightly with *developer risk* (α)
- ***B*** changes rapidly with *developer risk* (α) but very slightly with *customer risk* (β)





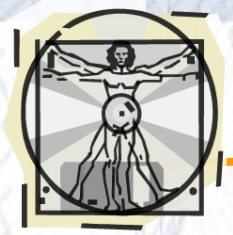
Reliability Demo Chart /4

- Boundary between ***accept*** and ***continue*** regions

$$T_n = \frac{A}{1-\gamma} - \frac{\ln \gamma}{1-\gamma} n$$ (γ is the discrimination ratio)

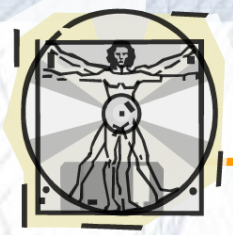
- Boundary between ***reject*** and ***continue*** regions

$$T_n = \frac{B}{1-\gamma} - \frac{\ln \gamma}{1-\gamma} n$$ (γ is the discrimination ratio)



Reliability Demo Chart /7

- When risk levels (α and β) decrease, the system will require more test before reaching the accept or reject regions, *i.e.*, the continue region becomes wider
- When discrimination ratio (γ) decreases, the system will require more test before reaching the accept or reject regions, *i.e.*, the continue region becomes wider and the slope of the boundary line tends towards vertical



RDC: When and How to Use?

- When to use RDC?
  - When failure data is limited to a few failures, time of failures are known, and one wants to find out what is the trend for reliability of the system
- How to use RDC?
  - Collect failure data (failure number and its time)
  - Set the target MTTF and anticipated confidence levels
  - Put the (normalized) failure data on the graph and analyze the trend

RDC: Example /1

- Consumer risk
  $\beta = 5\%$
- Supplier risk
  $\alpha = 5\%$
- Discrimination ratio $\gamma = 2$

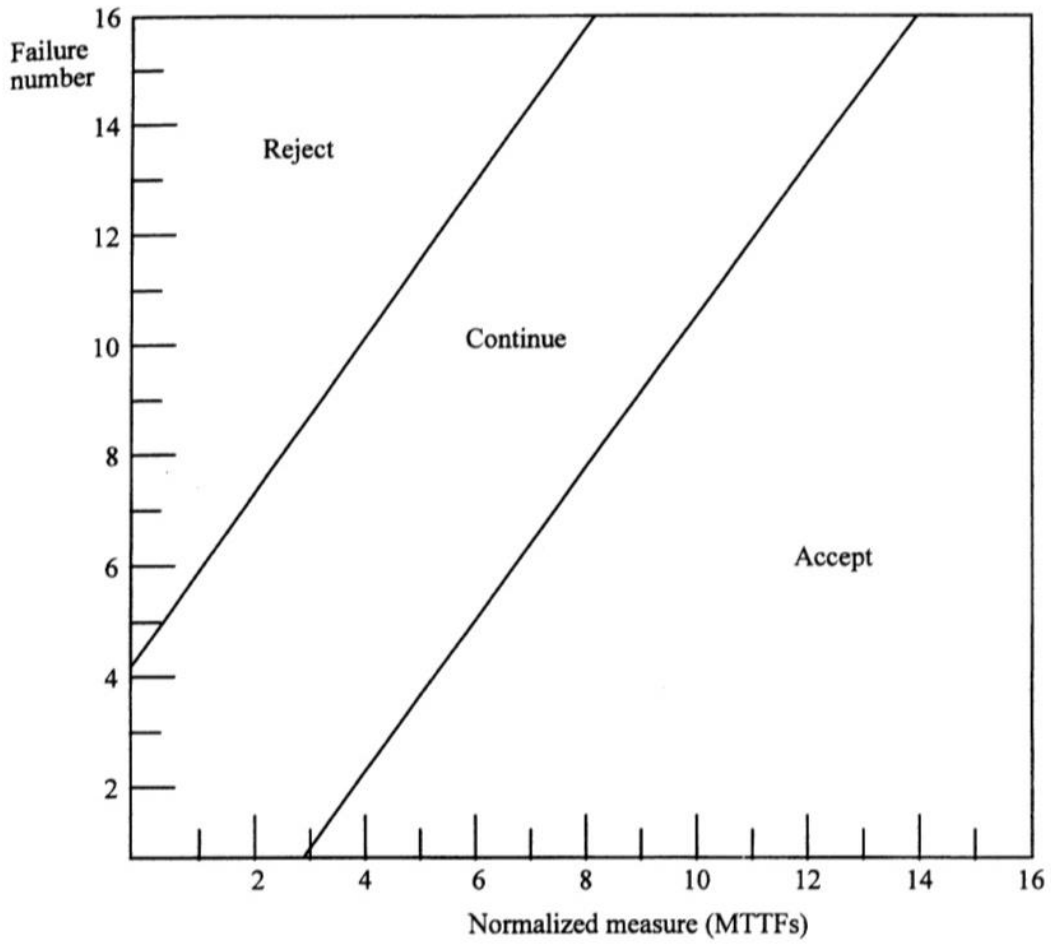


Figure from Musa's Book

RDC: Example /2

- Consumer risk
  $\beta = 1\%$
- Supplier risk
  $\alpha = 1\%$
- Discrimination ratio $\gamma = 2$

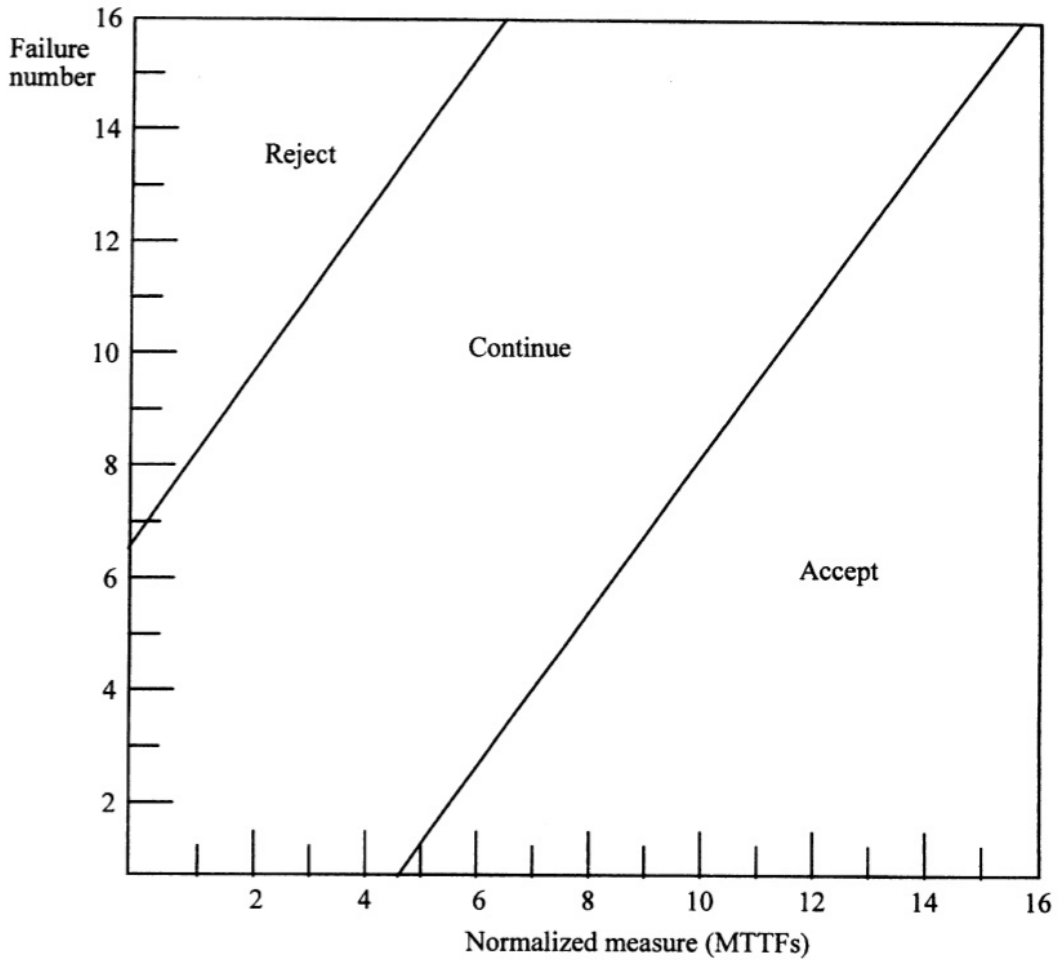
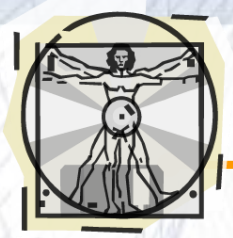


Figure from Musa's Book



RDC: Example /3

- Consumer risk
  $\beta = 0.1\%$
- Supplier risk
  $\alpha = 0.1\%$
- Discrimination ratio $\gamma = 2$

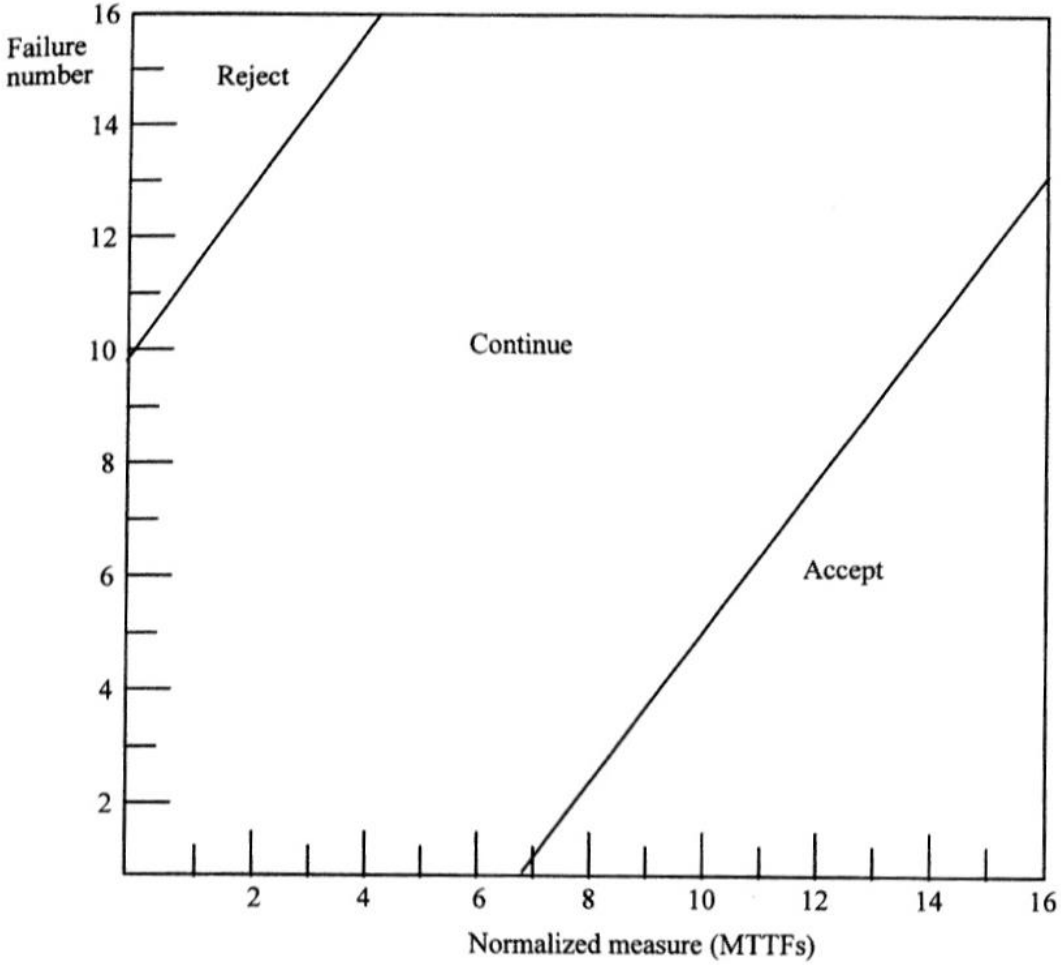


Figure from Musa's Book

RDC: Example /4

- Consumer risk
  $\beta = 10\%$
- Supplier risk
  $\alpha = 10\%$
- Discrimination ratio $\gamma = 1.2$

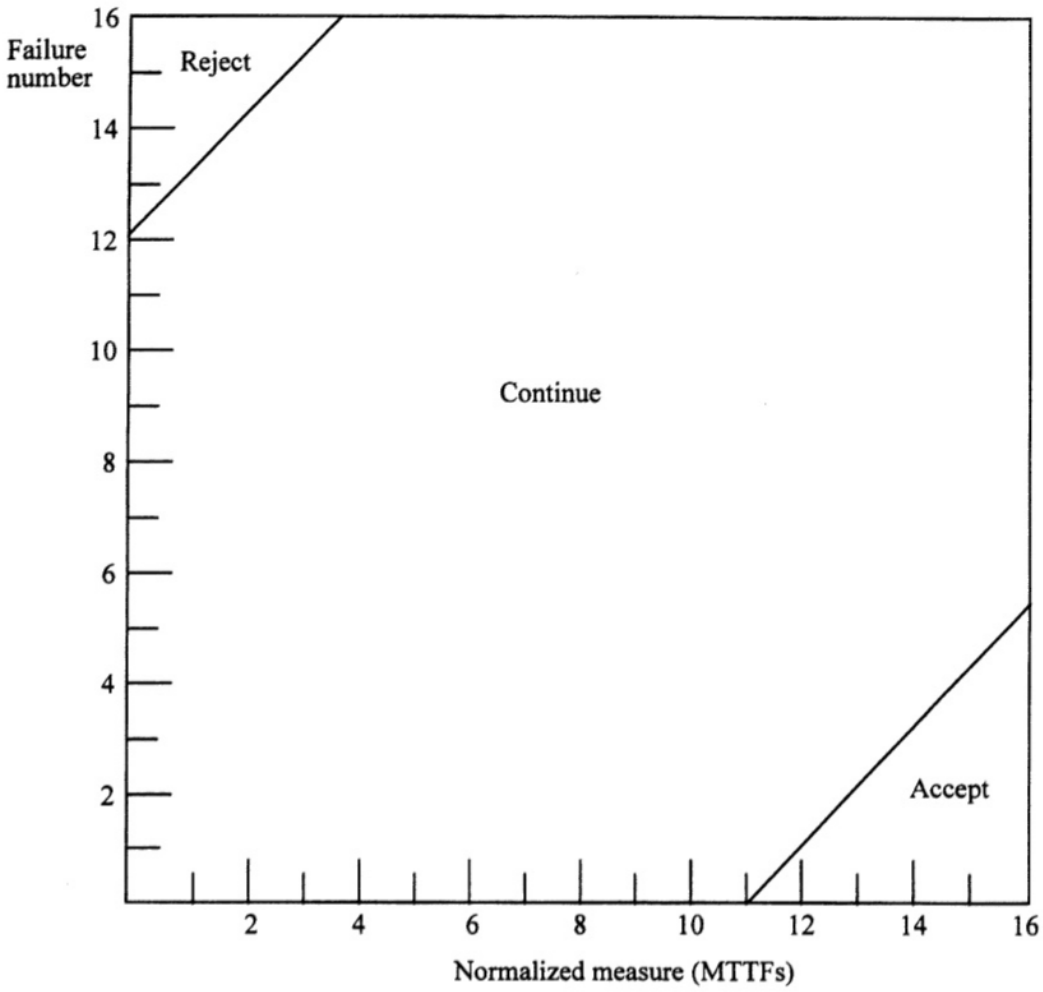


Figure from Musa's Book

RDC Usage: Example 1

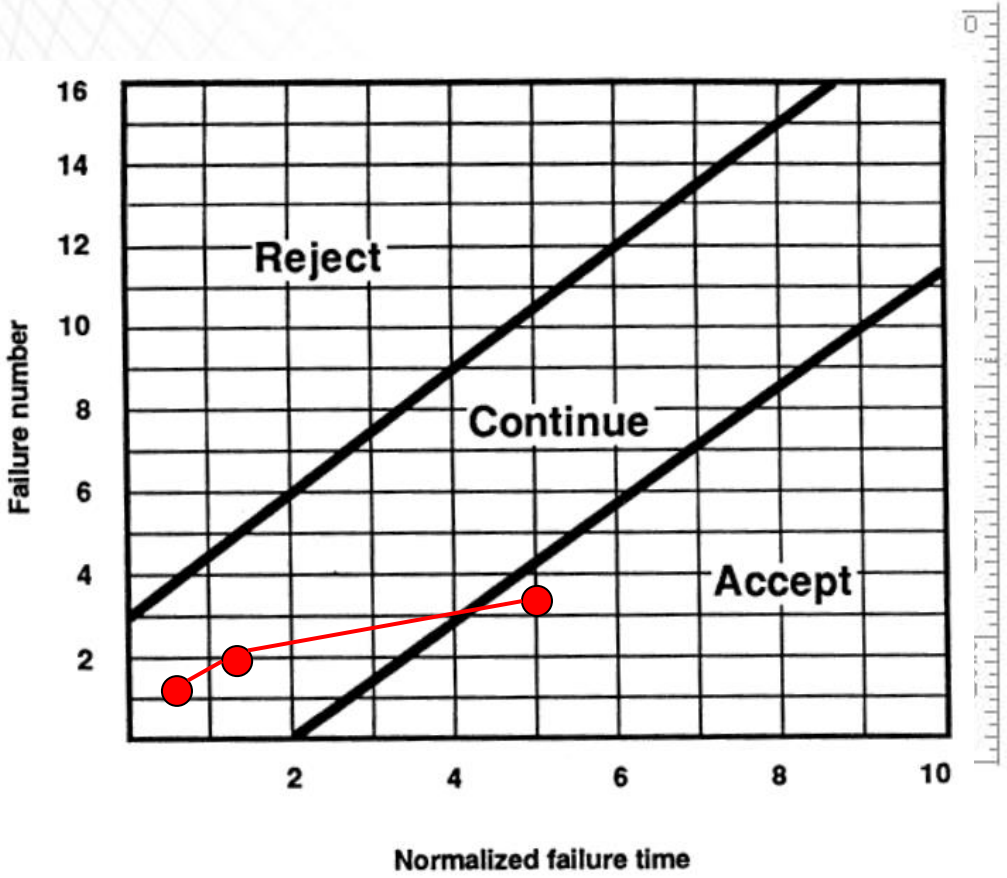
Failure number	*Measure (million transactions)*
1	0.1875
2	0.3125
3	1.25

$\lambda_F = 4$ failures/million transactions

$\alpha = \%10$

$\beta = \%10$

$\gamma = 2$



RDC Usage: Example 2

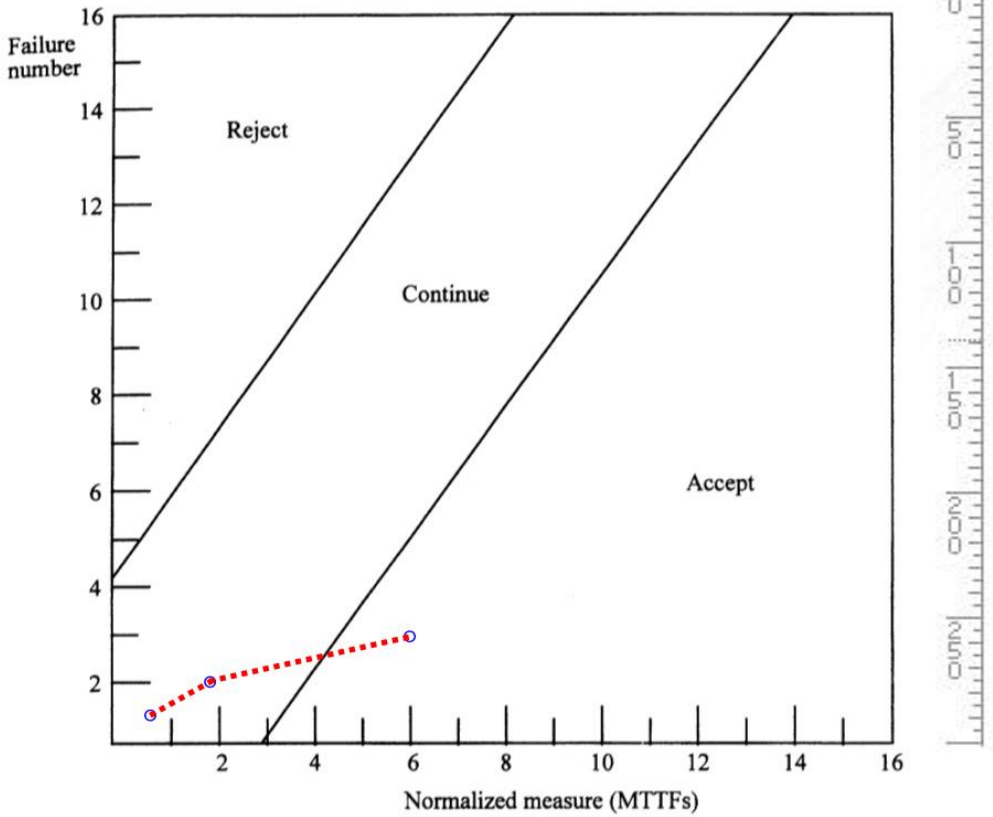
Failure number	*Measure (CPU hour)*
1	8
2	19
3	60

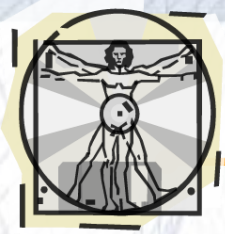
$\lambda_F = 0.1$ failures/CPU hour

$\alpha = 0.05$

$\beta = 0.05$

$\gamma = 2$



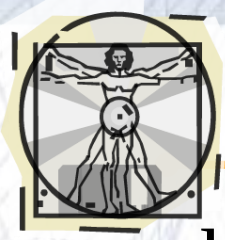


RDC Usage: Example 3

- We are going to buy a new colour copier for our department. We have borrowed the copier for the test run and we are going to conduct certification test on it. Maker's data shows that we need to change the toner every 10,000 pages. We would like to have the system running with only one failure during the lifetime of a toner (i.e. one paper jam during the toner's lifetime).

a) What shall be our failure intensity objective for the system?

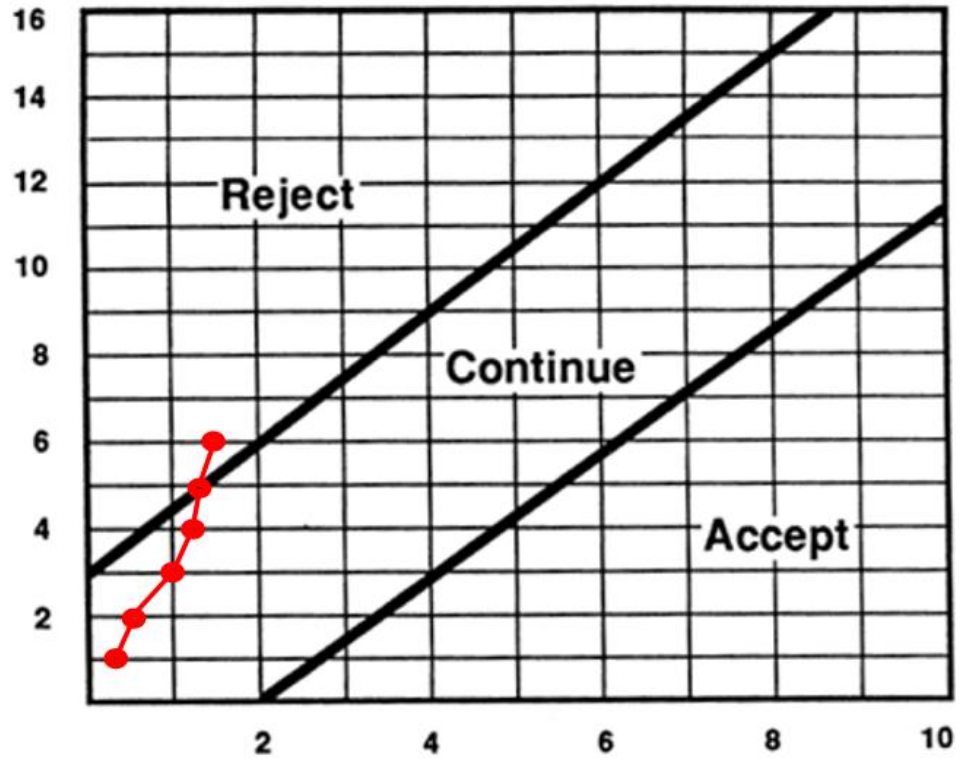
λ_F = 1/10,000 pages



Example 3 (contd.)

b)We observed that paper jams occurred at 4,000 pages, 6,000 pages, 10,000 pages, 11,000 pages, 12,000 pages and 15,000 pages of output. Does it meet our objective?

Example 3 (contd.)

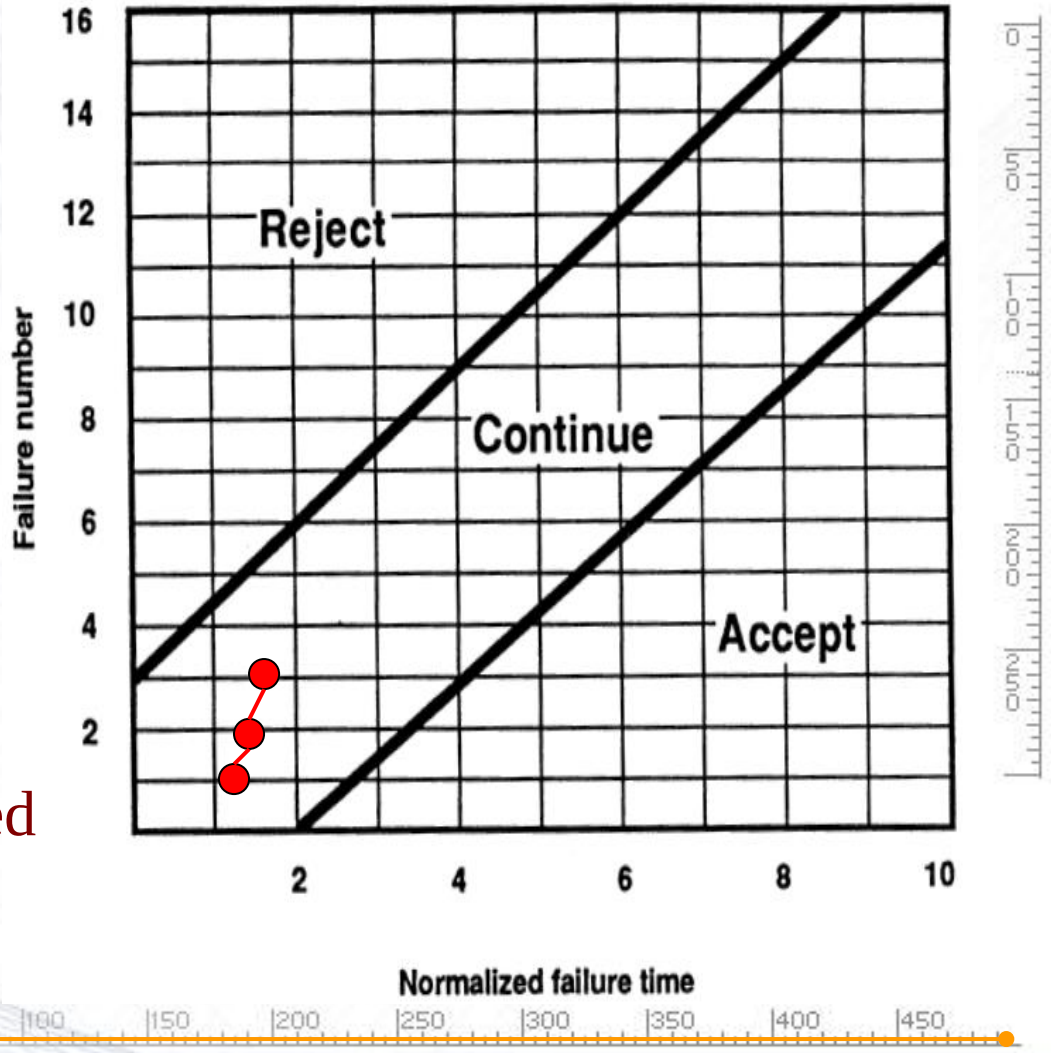


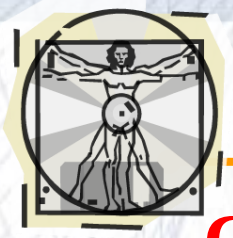
- Because of failing the certification test we will reject the copier.

Example 3 (contd.)

c) What if the paper jams occurred 11,000 pages, 12,000 pages and 15,000 pages of output? Does it meet our objective now?

- Not sure. We may need more testing.





RDC Usage: Example 4

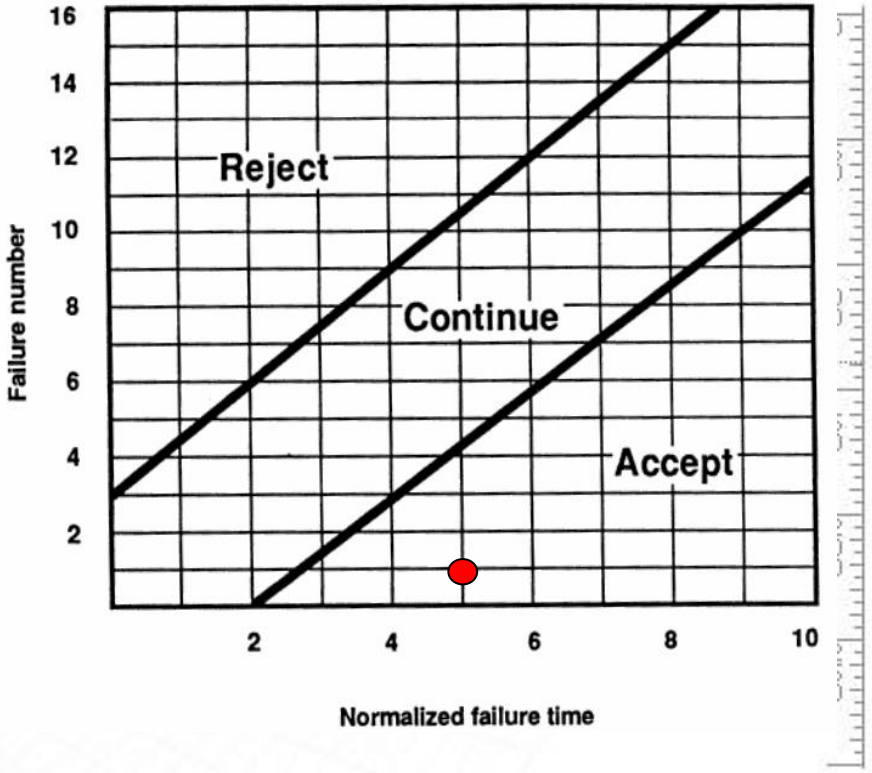
Continuous Integration

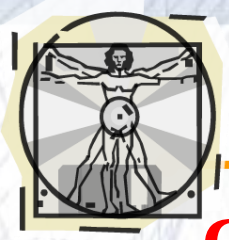
- We have developed a program for a Web server with the failure intensity of 1 failure/100,000 transactions
- The program runs for 50 hours, handling 10,000 transactions per hour on average with no failures occurring
- Is this software good to be released?

Example 4 (contd.)

Answer

- The total number of transactions is $50 \times 10{,}000 = 500{,}000$
- Therefore, normalized failure time is 5
- Using RDC, this point is well in the accept region, therefore the answer is YES and there is 10% risk of wrongly accepting or rejecting the software





Example 4 (contd.)

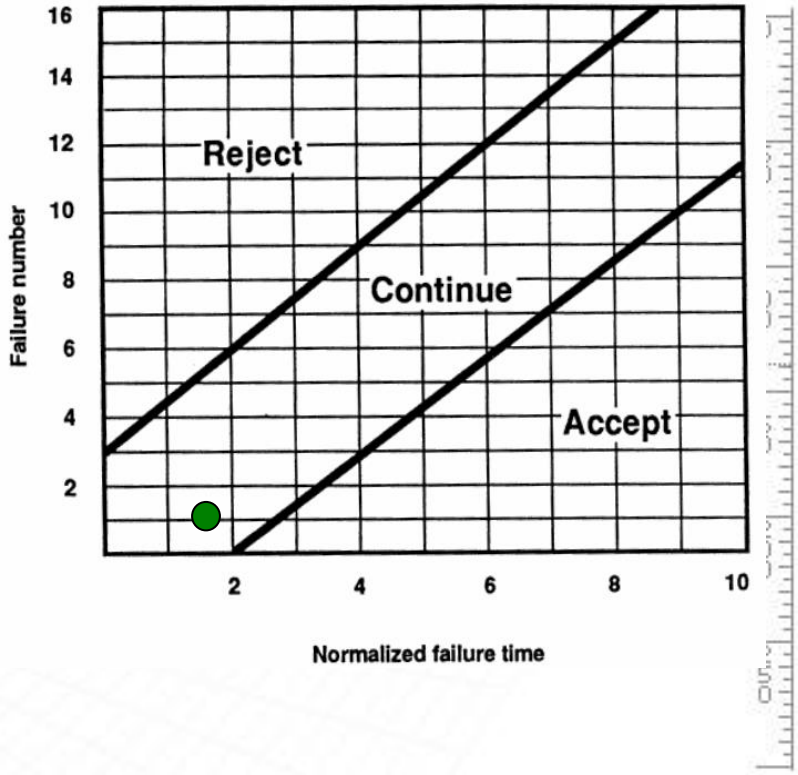
Continuous integration

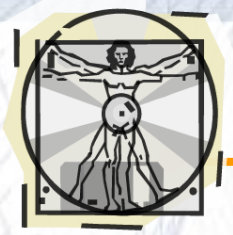
- Suppose that a new component is added to the system
- Failure intensity of the new component is 0.5 failures/100,000 transactions
- The system experiences a failure after 10 hours (i.e. processing 100,000 transactions)
- Is this software good to be released?

Example 4 (contd.)

Answer

- The new failure intensity for system $\lambda = \lambda_1 + \lambda_2 = 1.5$ per 100,000 transactions
- Total number of transactions until failure is $10 \times 10{,}000 = 100{,}000$, therefore, normalized failure is 1.5
- Using RDC, this point is in the continue region, therefore the answer is NO and we have to continue testing



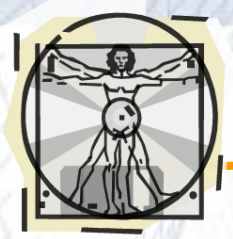


RDC: Benefits and Limitations

- RDC analysis is very versatile, time and cost efficient way of analyzing the reliability of a system
- A disadvantage of the RDC is that it cannot be used to calculate the exact quantitative value for the reliability (or availability) of the system under study
  - RDC can only indicate that the SUT is acceptable or not
- Experimenting with different values of confidence levels and MTTF (what-if scenarios) is doable
- RDC open source excel project: http://sourceforge.net/projects/rdc/

Section 4

Assessment Criteria:
Reliability Growth

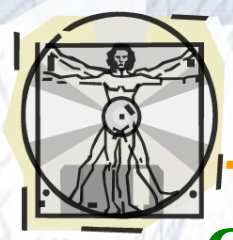


Assessment Criteria

- Indicators from at least the following dimensions should be considered together to get an adequate picture of the quality of the product
  - System stability, reliability, and availability
  - failure volume (bugs found and fixed)
  - Outstanding critical problems reported
  - Feedback from early customer programs
  - Other quality attributes that are of specific importance to a particular product and its customer requirements and market acceptance (e.g., ease of use, performance, security, portability, etc.)

Our focus

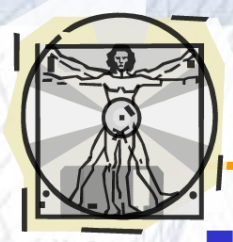
From Dr. Kan's Book



Review: Test Completion Criteria

Q. *When is testing enough?*

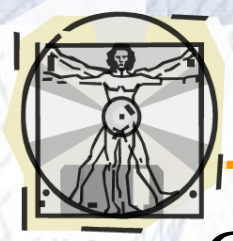
- Test phase time or resources are exhausted
- All black-box test cases are run
- White-box test coverage targets are met
- Mutation tests and/or planned GUI tests passed
- Rate of failure discovery goes below a target value
- Target percentage of all failures in the system are found
- Measured ***reliability*** of the system achieves its target value



Approach: Models

- Use SRE tools such as CASRE or SRTAT to plot the λ/λ_F ratio vs. time to see the trend of changing λ
- SRE tools use **basic exponential** and **logarithmic Poisson** and a few other models
- The Basic exponential model assumes finite failures in infinite time; the Logarithmic Poisson model assumes infinite failures
- In estimating the λ/λ_F ratio, the Basic exponential model tends to be "optimistic" (low); the Logarithmic Poisson model tends to be "pessimistic" (high)

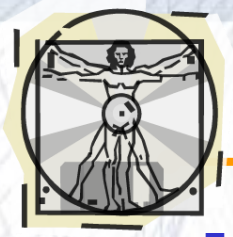
λ_F : Target failure rate for the developed system



Release Criteria /3

Consider releasing the product when

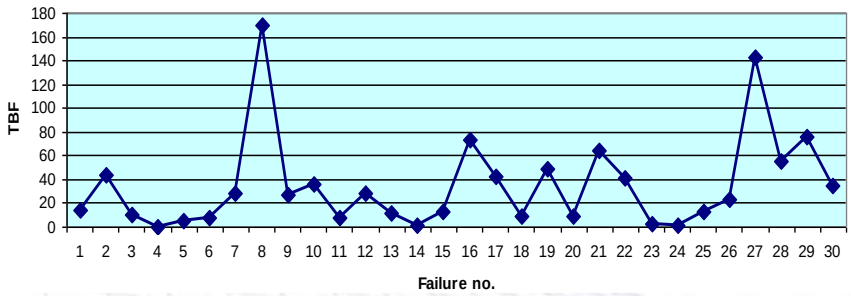
1. Tests terminated satisfactorily for the product itself
2. Test terminated satisfactorily for all the product variations and the normalized λ/λ_F ratio for these variations doesn't appreciably exceed 0.5
3. The product and its variations pass all acceptance test rehearsals planned for them
4. Related systems and/or components, libraries, etc., pass all acceptance tests

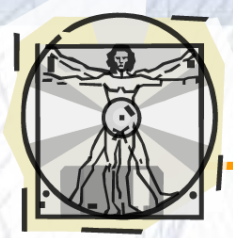


Example 1

- We want to select an operating system as a build platform for a new software product. The Windows operating system was evaluated for this purpose. The system log during the testing period showed 30 errors recorded in the event log file as shown.

No.	*Type*	*TBF (hours)*	*Error Source*
1	Error	14.75	DCOM
2	Error	43.99	Service Control Manager
3	Error	9.89	DCOM
4	Error	0.07	DCOM
5	Error	5.70	DCOM
6	Error	7.89	DCOM
7	Error	28.79	DCOM
8	Error	170.15	DCOM
9	Error	26.83	DCOM
10	Error	36.15	System Error
11	Error	7.99	DCOM
12	Error	28.09	DCOM
13	Error	11.80	DHCP
14	Error	1.78	System Error
15	Error	12.50	DCOM
16	Error	73.08	DCOM
17	Error	42.60	Service Control Manager
18	Error	9.18	DCOM
19	Error	49.43	DCOM
20	Error	9.19	DCOM
21	Error	64.25	System Error
22	Error	40.90	System Error
23	Error	3.07	Service Control Manager
24	Error	0.75	Service Control Manager
25	Error	13.36	System Error
26	Error	23.02	Service Control Manager
27	Error	143.31	Service Control Manager
28	Error	55.46	System Error
29	Error	75.57	DCOM
30	Error	34.31	DHCP





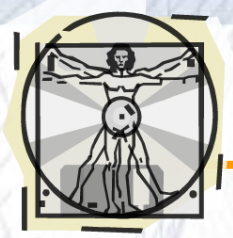
Example 1 (cont'd)

- Calculate the system MTTF, failure intensity and reliability for 10 hours of operation, assuming that MTTF=MTBF and using exponential reliability model

$$MTBF = \frac{\sum_{i=1}^{30} MTBF_i}{i} = \frac{1043.85}{30} = 34.795 \text{ hour}$$

$$\lambda = \frac{1}{MTTF} = \frac{1}{34.795} = 0.0287 \text{ failure/hour}$$

$$R = e^{-10/34.795} = 0.75$$



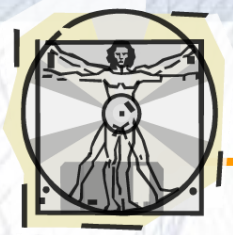
Example 1 (cont'd)

- The system must be rebooted after each "System Error" type error. If the recovery time to reboot the system after each "System Error" is 15 minutes, calculate the availability of this system.

$$Availability = \frac{uptime}{uptime + downtime} = \frac{1043.85}{1043.85 + (0.25 \times 6)} = 0.9985$$

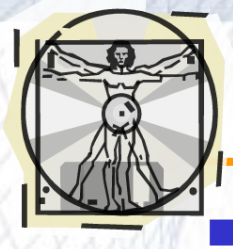
- If the target failure rate for deploying this system is 1 failure per 100 hours can we elect using this system as our build platform?

$\lambda_F = 0.01$ and λ_F is smaller than λ=0.0287
so the answer is NO



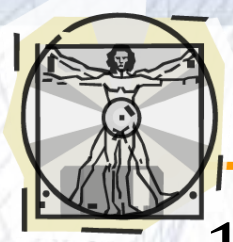
Case Study: Using CASRE

- Software Reliability Estimation tool running on 32-bit Windows OS
- CASRE extends the SMERFS package by adding a menu based GUI to it
- Uses ASCII text input data files
- Displays results in tabular and/or graphical form (plots)
- Can use many different models



Input Data Specification

- ASCII based text file with a .dat extension
- Two file formats
  - Time Between Failures (error #, time since last failure, failure severity class)
  - Failure Counts (interval #, # errors in interval, interval length, failure severity class)
- The format of the file must be strictly adhered to
- No direct manipulation of the data file is allowed but CASRE has menu links to common text editors
- For accuracy of results, the expected number of failures should be greater than 40-50 failures



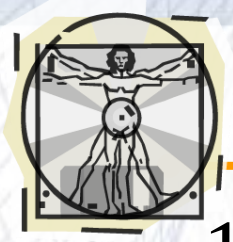
Using CASRE /1

1. Prepare input data
   - Input data can be either failure count or failure per interval data

Sample failure count data

<failure number> <number of natural or time units since previous failure> <severity class>

1	30	1
2	55	1
3	70	1
4	60	1
5	90	1
6	110	1
7	100	1
8	150	1
9	120	1
10	215	1



Using CASRE /2

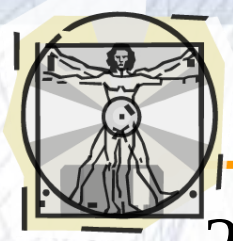
1. Prepare input data
   - Input data can be either failure count or failure per interval data

Sample failure per interval data

<interval number> <failure in interval>
<duration of interval> 0 0 0 <severity class>

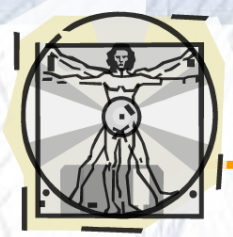
1	5	2.5	0	0	0	1
2	1	1	0	0	0	1
3	4	3	0	0	0	1
4	1	2	0	0	0	1
5	0	1.5	0	0	0	1
6	1	3	0	0	0	1
7	2	4	0	0	0	1
8	1	2.5	0	0	0	1
9	2	3	0	0	0	1
10	2	5	0	0	0	1

Example from Musa's Book



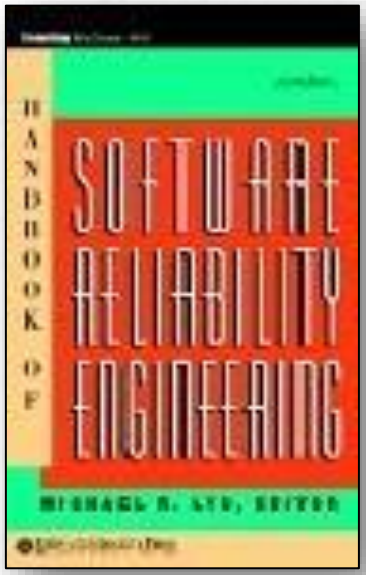
Using CASRE /3

2. Read input file
3. Check if data shows reliability growth (trend test)
4. Select data range
5. Select parameter estimation method
6. Select and run model(s)
7. View and interpret model results
   - Goodness of fit test
   - Model ranking
   - Prediction based on plots



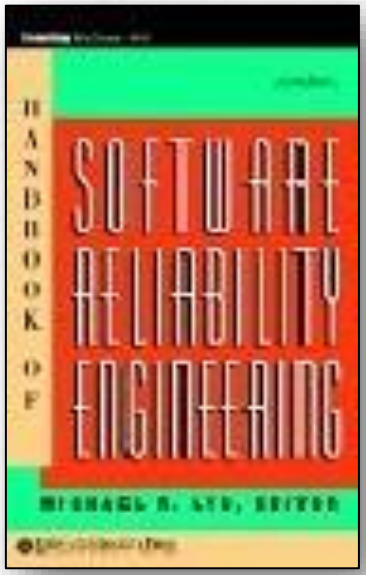
CASRE Reliability Models /1

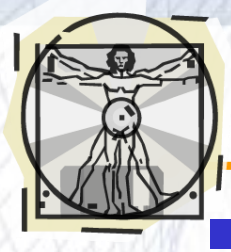
- Time between failure models
  - Geometric
  - Jelinski-Moranda
  - Littlewood-Verrall
  - Musa-Basic
  - Musa-Okumoto
  - NHPP



CASRE Reliability Models /2

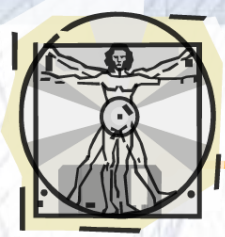
- Failure Count models
  - Generalized Poisson
  - NHPP
  - Schneidewind
  - Shick-Wolverton
  - Yamada S-shaped





Trend Check

- Reliability models should only be used on data where the overall reliability is increasing as testing continues
- Reliability is increasing if the mean time between failures (MTBF) increases as the total number of failures increases during testing
- CASRE version 3 has an automatic trend test option, that will inform the user if the data is applicable to the reliability models

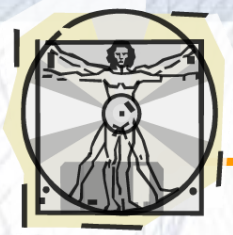


Trend Related Questions …

- Is the system reliability increasing, decreasing or stable?
- Which reliability growth model fits best the gathered data?
- Can the same model be used in all cases of reliability growth, decrease and stable?

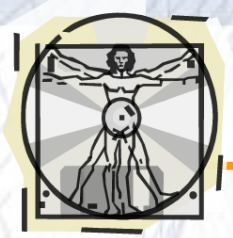
And Trend Related Answers …

- Reliability trends can be analyzed by "trend tests"
- Trend tests can be used to help determine whether the system undergoes reliability growth, decrease or stable reliability
- Trend analysis also helps select appropriate reliability model for each phase



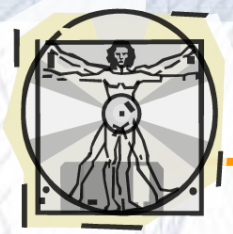
Failure Data for Trend Tests

- The trend tests work with the failure data
- The trend can be analyzed using
  - Inter-failure times data (i.e. time of failure or time between failures known) or
  - Failure count data (i.e. failure per interval or cumulative failure number known)



Inter-failure Times Data /1

- Two trend tests are commonly carried:
  - Arithmetical mean test
  - Laplace tests

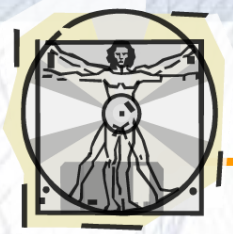


Inter-failure Times Data /2

- The arithmetical mean of the inter-failure times consists of calculating arithmetical mean $\tau(i)$ of the observed inter-failure times θj

$$\tau(i) = \frac{1}{i}\sum_{j=1}^{i}\theta_j$$

- An increasing series of $\tau(i)$ indicates reliability growth and a decreasing series suggests reliability decrease

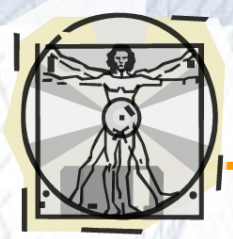


Inter-failure Times Data /3

- For *N(T)* as the cumulative number of failures over the time period *[0, T]*, the Laplace factor *u(T)* is derived:

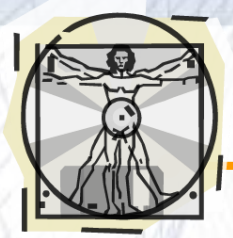
$$u(i) = \frac{\frac{1}{i-1}\sum_{n=1}^{i-1}\sum_{j=1}^{n}\theta_j - \frac{\sum_{j=1}^{i}\theta_j}{2}}{\sum_{j=1}^{i}\theta_j\sqrt{\frac{1}{12(i-1)}}}$$

- For the case that *T* is equal to the time of occurrence of failure *i*



Inter-failure Times Data /4

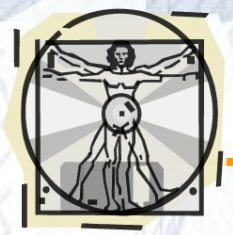
- Negative values of the Laplace factor *u(i)* indicate a decreasing failure intensity, i.e., reliability growth
- Positive values of the Laplace factor *u(i)* indicate an increasing failure intensity, i.e., reliability decrease
- Values between –2 and +2 indicate stable reliability



Failure Count Data /1

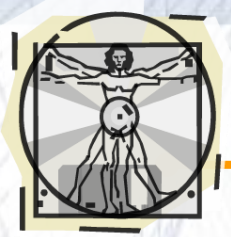
- For the time period *[0, T]*, divided into *k* units of equal length and for *n(i)* be the number of failures observed during the time interval *i*, the Laplace factor *u(k)* is derived by:

$$u(k) = \frac{\sum_{i=1}^{k}(i-1)n(i) - \left(\frac{k-1}{2}\right)\sum_{i=1}^{k} n(i)}{\sqrt{\frac{k^2-1}{12}\sum_{i=1}^{k} n(i)}}$$



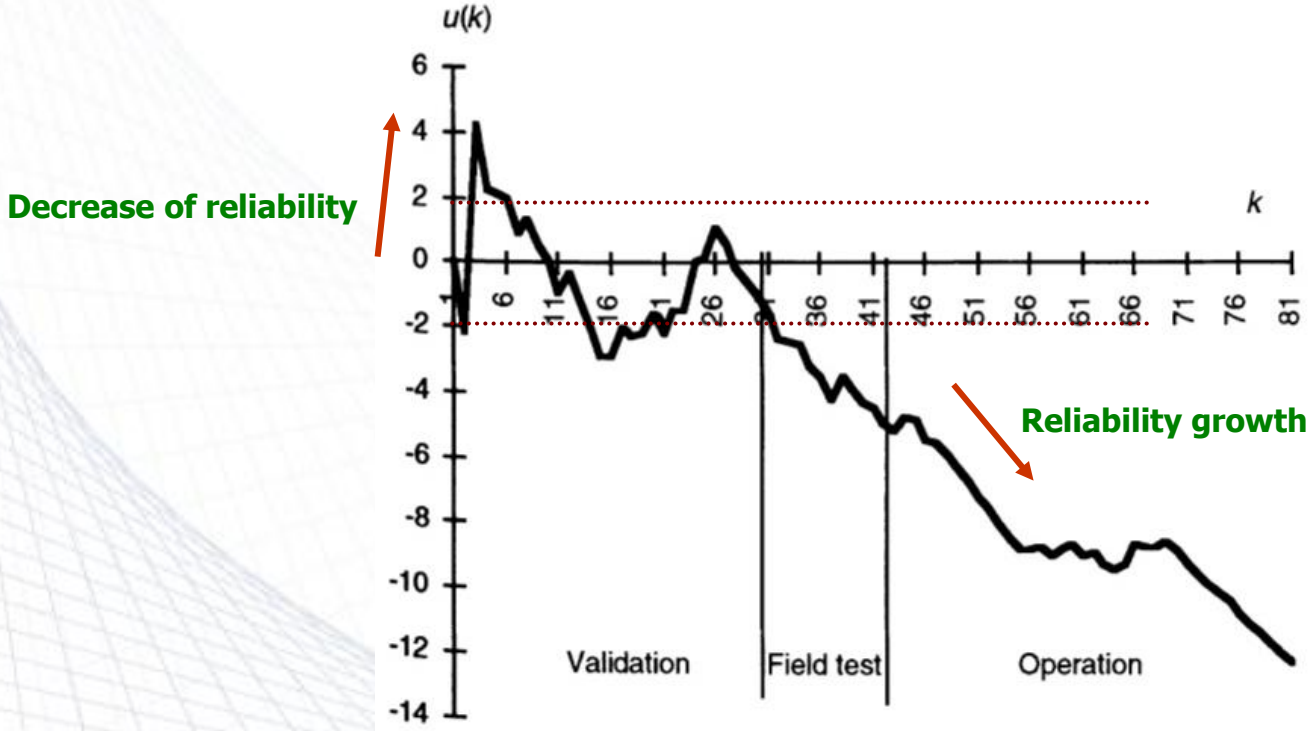
Failure Count Data /2

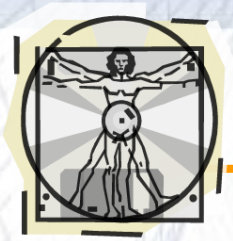
- Negative values of the Laplace factor *u(k)* indicate a decreasing failure intensity, i.e., reliability growth
- Positive values of the Laplace factor *u(k)* indicate an increasing failure intensity, i.e., reliability decrease



Typical Plots /3

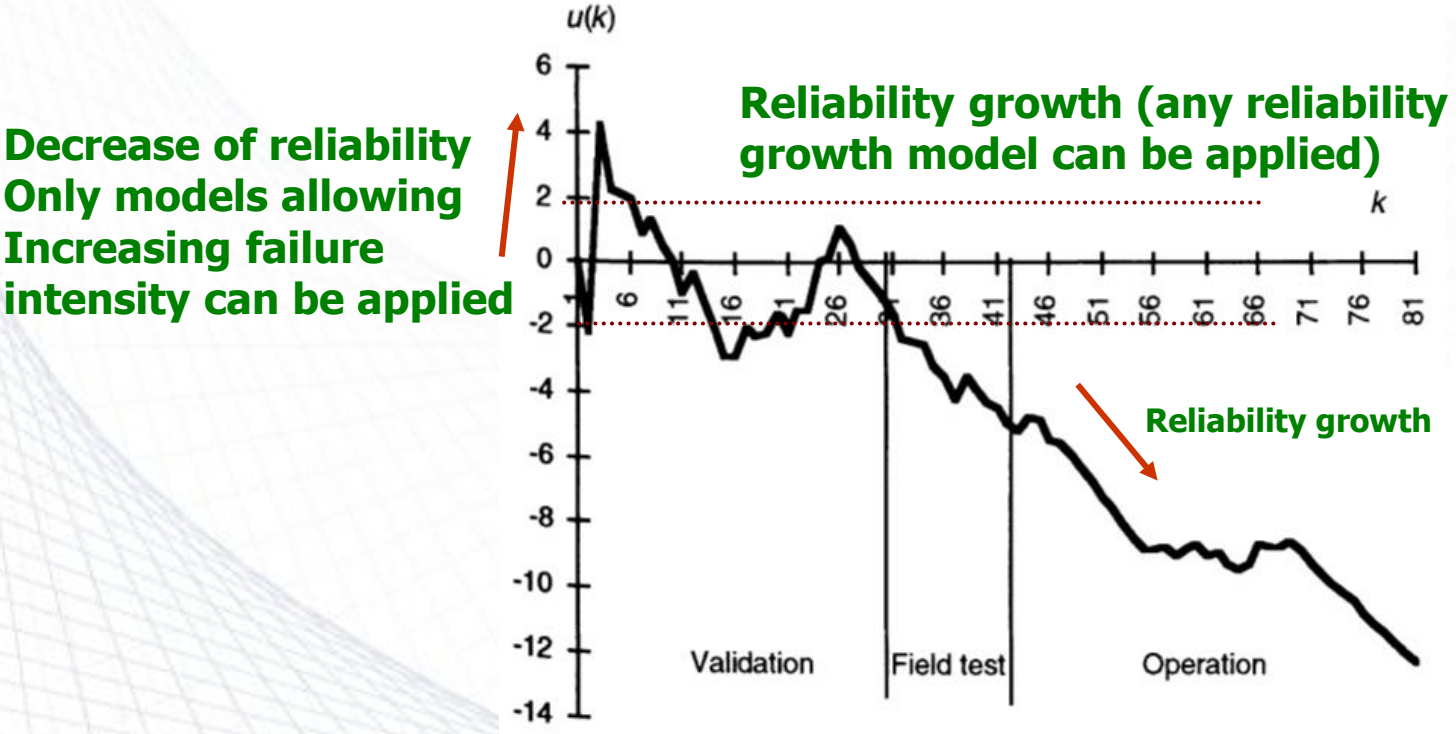
- Typical plot for Laplace factor during various project phases

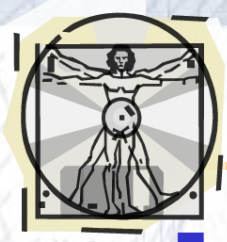




Selecting Models

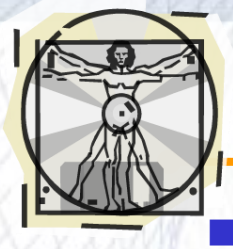
- Typical plot for Laplace factor during various project phases





Case Study

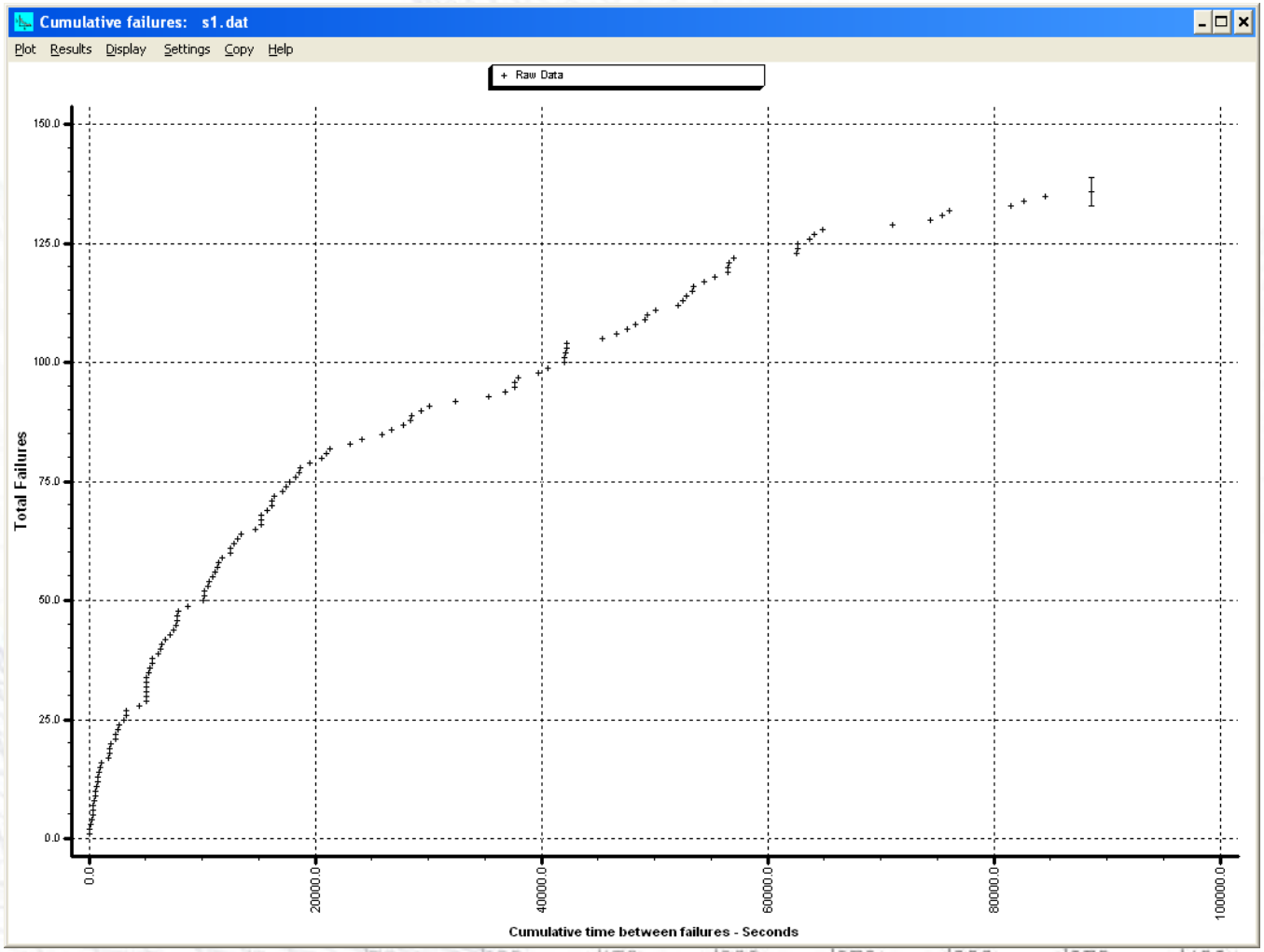
- Project X is a web based application for accessing a database using a browser
- This version of the software is a minor release with changes to the GUI display and data access engine
- Two programmers were assigned to the project. One programmer worked on the GUI, and the other on the data access engine
- The project took approximately 4 weeks to complete



Case Study (contd.)

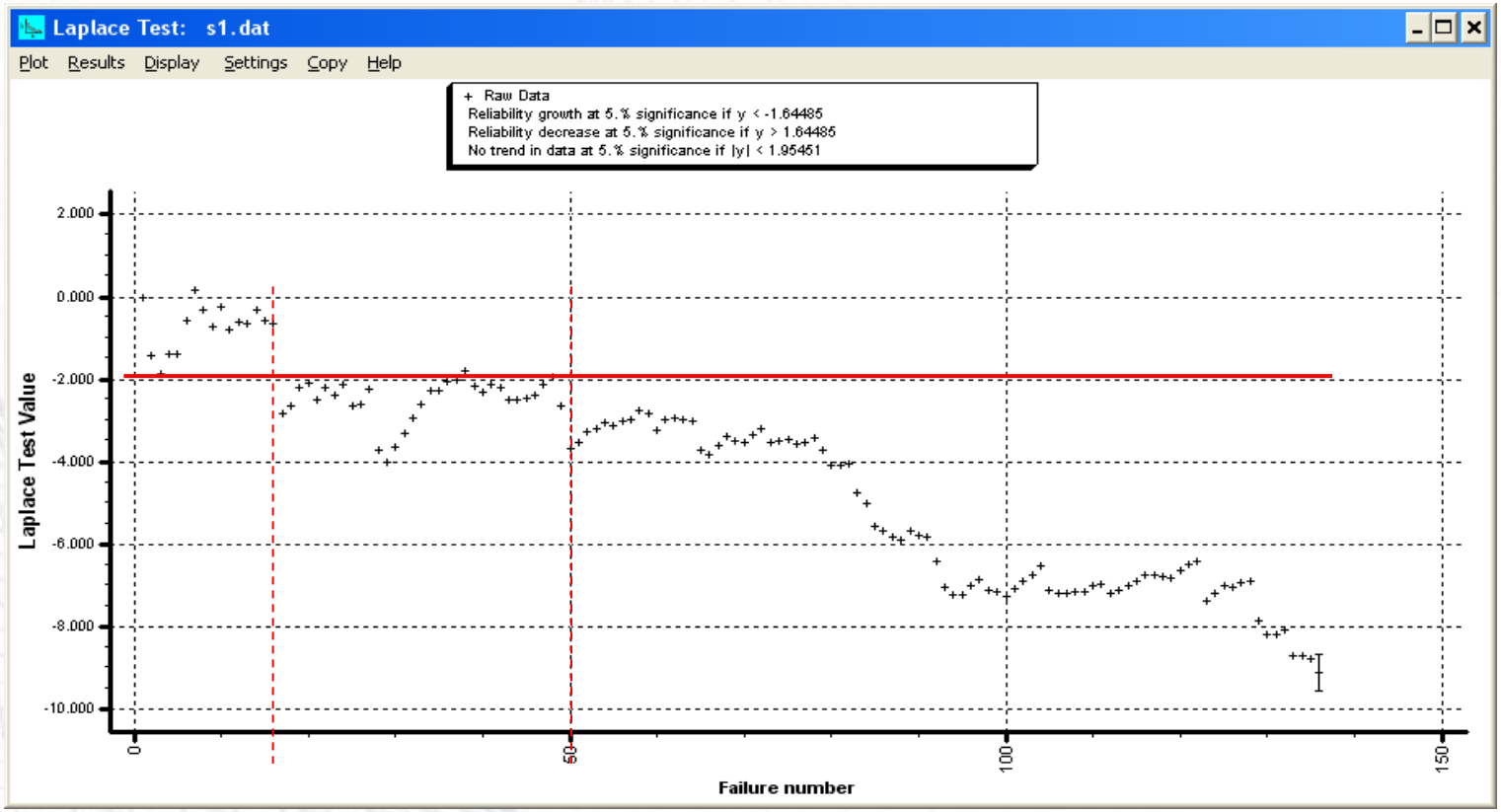
- A single tester was assigned to the project
- The test phase was completed in approximately 25 hours (3 working days or 90,000 seconds)
- 136 failures were discovered during the testing
- Using the dates and times recorded for the failures discovered during testing, a “time between failures” input file was generated for CASRE
- The severity of all the failures was set to
  - 1 - Low Severity

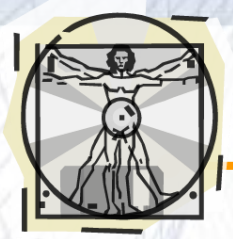
Cumulative Failures Plot



Trend Analysis

- Laplace test shows reliability growth.





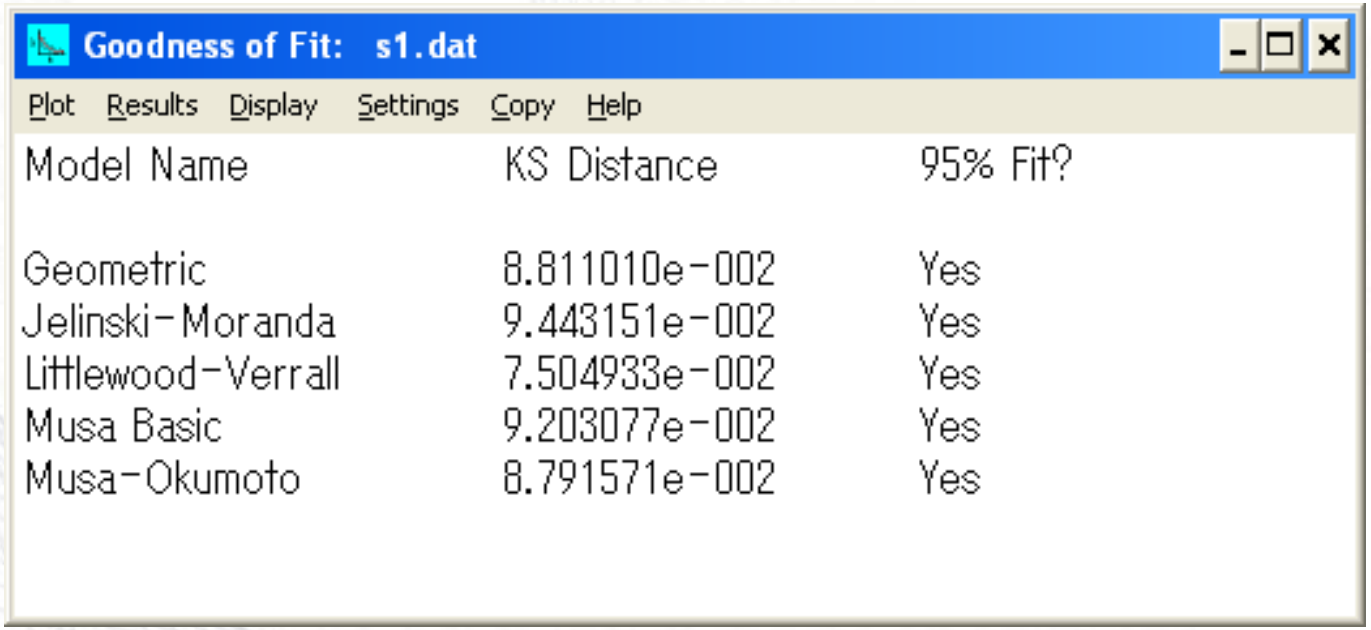
Project Results

- In order to determine which models would provide the best fit for the project data, the following models were run
  - Geometric
  - Jelinski - Moranda
  - Littlewood - Verrall
  - Musa Basic
  - Musa - Okumoto

Goodness of Fit Test

On Graphic display window select:

Display → Goodness of fit



CASRE Model Ranking

On Graphic display window select:

Display → Model rankings → Rank summary or Rank details

Model Ranking Summary: s1.dat

Plot Results Display Settings Copy Help

Model Name	Rank	Reliability
Geometric	1	1.463635e-001
Littlewood-Verrall	1	9.667713e-002
Musa-Okumoto	1	1.578047e-001
Musa Basic	4	3.278631e-001
Jelinski-Moranda	5	3.702861e-001

Model Ranking Detail: s1.dat

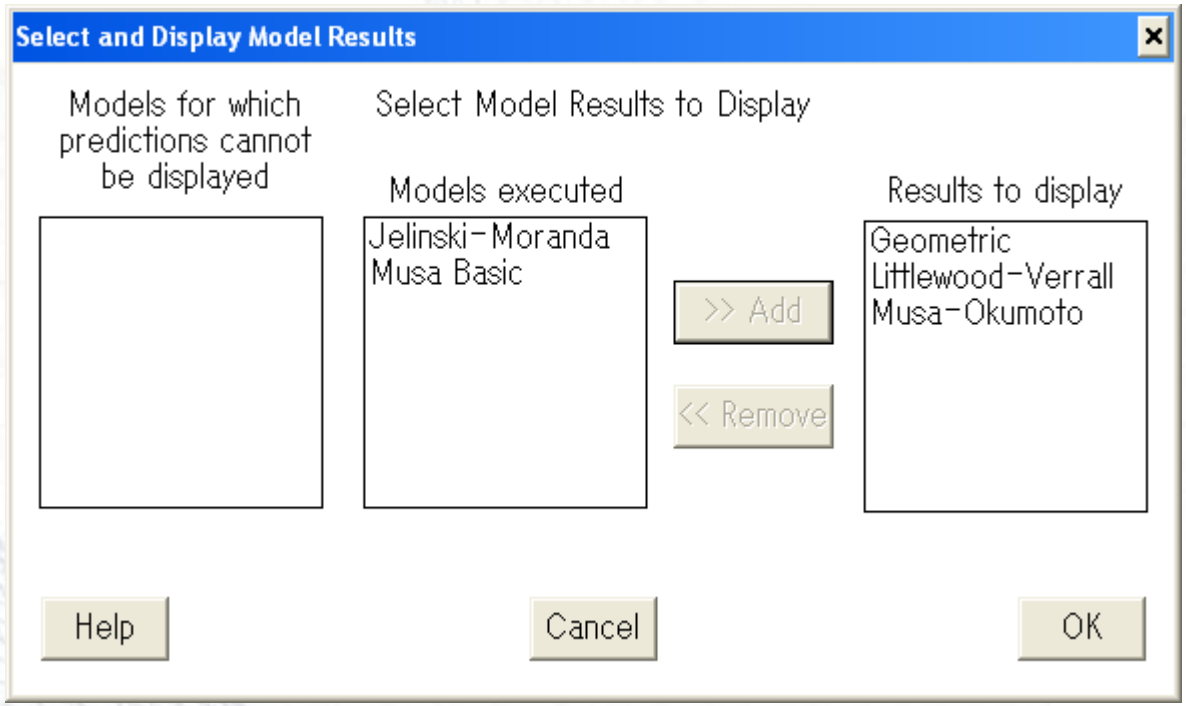
Plot Results Display Settings Copy Help

Model Name	-ln PL	Model Bias	Bias Trend	Model Noise	KS Distance	Rank
Geometric	5.310828e+002 (1)	7.806553e-002 (1)	1.084881e-001 (3)	2.026753e+000 (2)	8.811010e-002 (3)	1
Littlewood-Verrall	5.314613e+002 (3)	1.023081e-001 (3)	9.225496e-002 (1)	1.632047e+000 (1)	7.504933e-002 (1)	1
Musa-Okumoto	5.311823e+002 (2)	8.197740e-002 (2)	1.056971e-001 (2)	2.307074e+000 (3)	8.791571e-002 (2)	1
Musa Basic	5.376974e+002 (4)	1.935277e-001 (4)	1.556185e-001 (5)	5.351962e+000 (4)	9.203077e-002 (4)	4
Jelinski-Moranda	5.387762e+002 (5)	2.201526e-001 (5)	1.542180e-001 (4)	5.582339e+000 (5)	9.443151e-002 (5)	5

Display Resuls

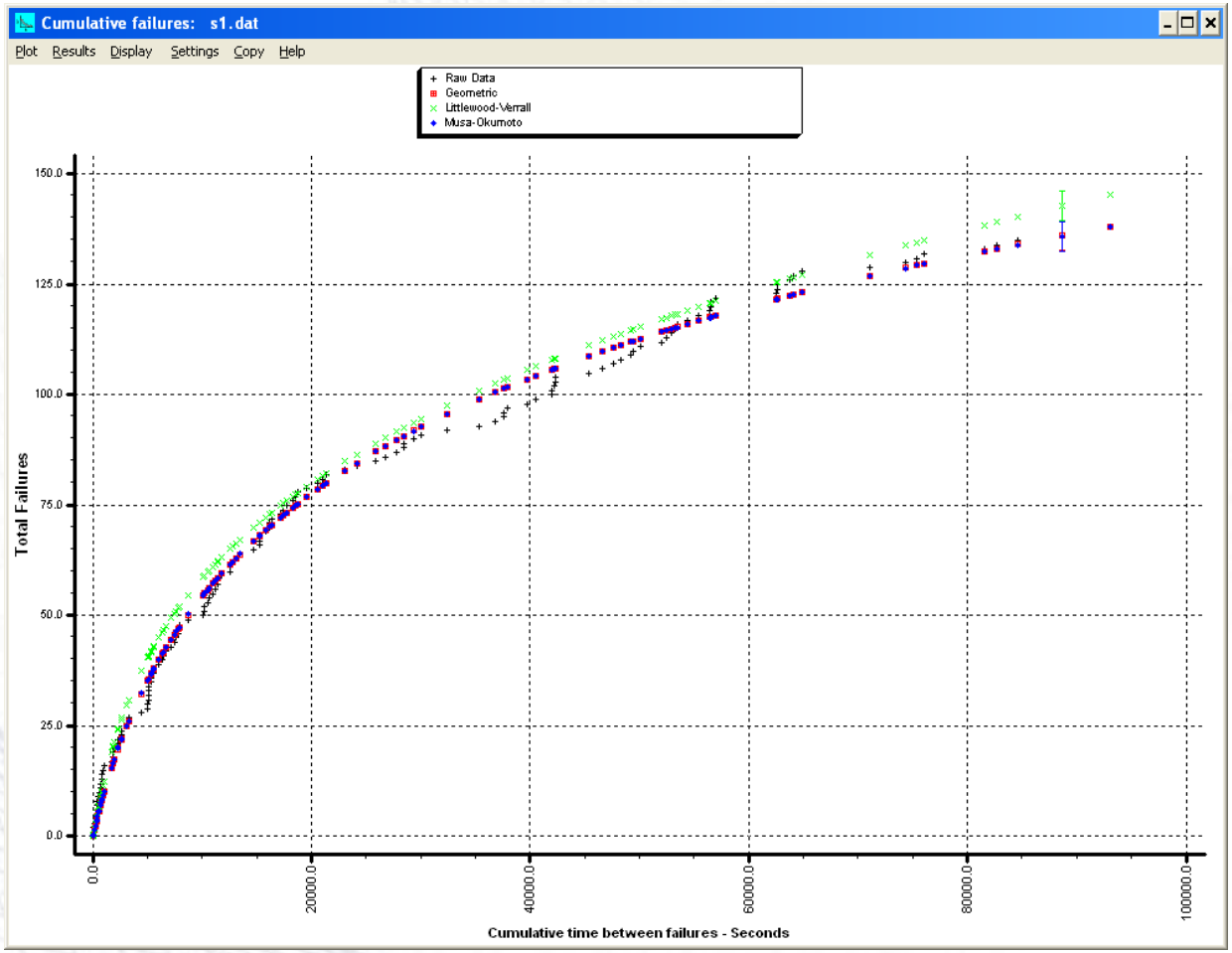
On Graphic display window select:

Results → Select model results

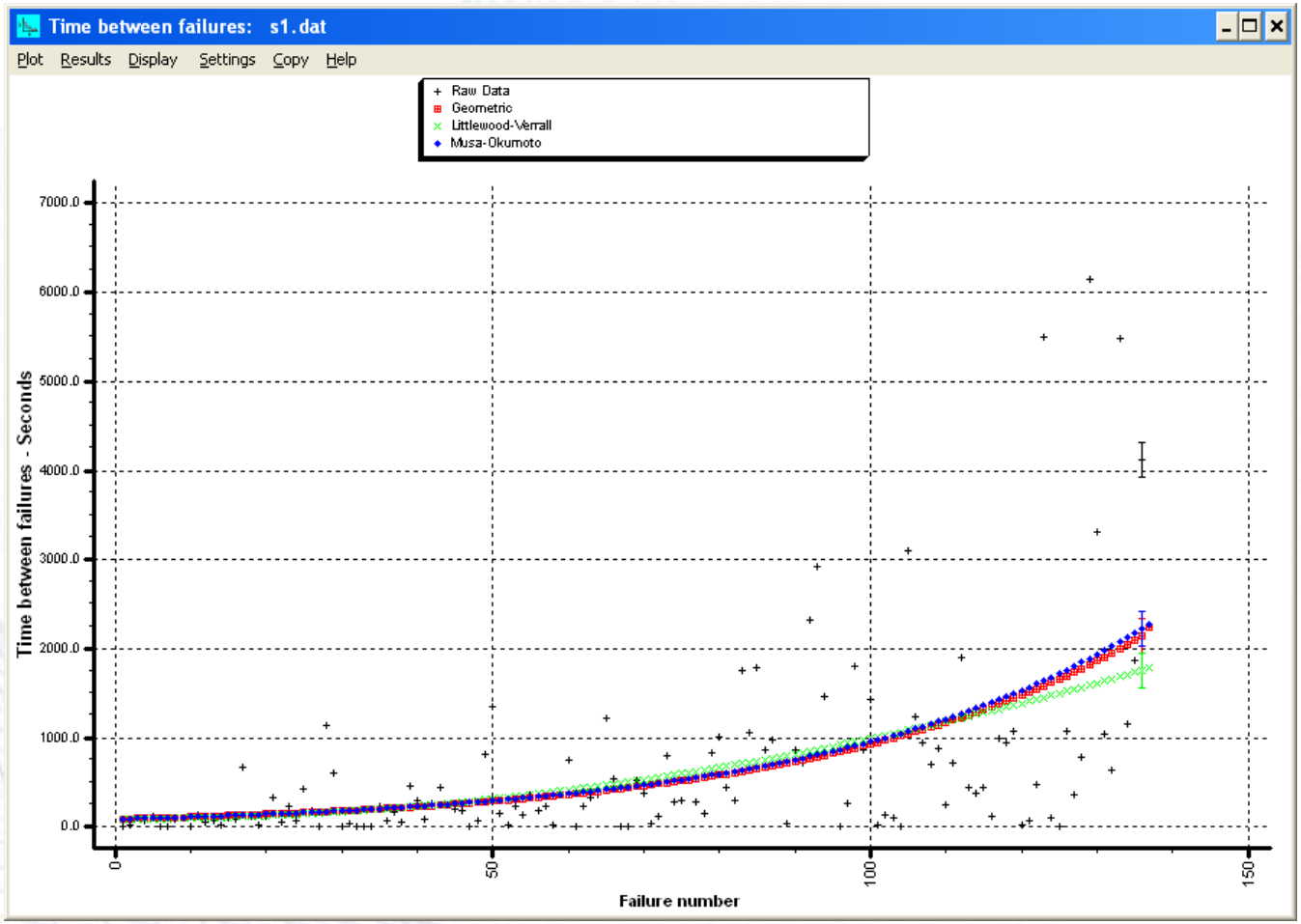


Only 3 graphs can be displayed at a time

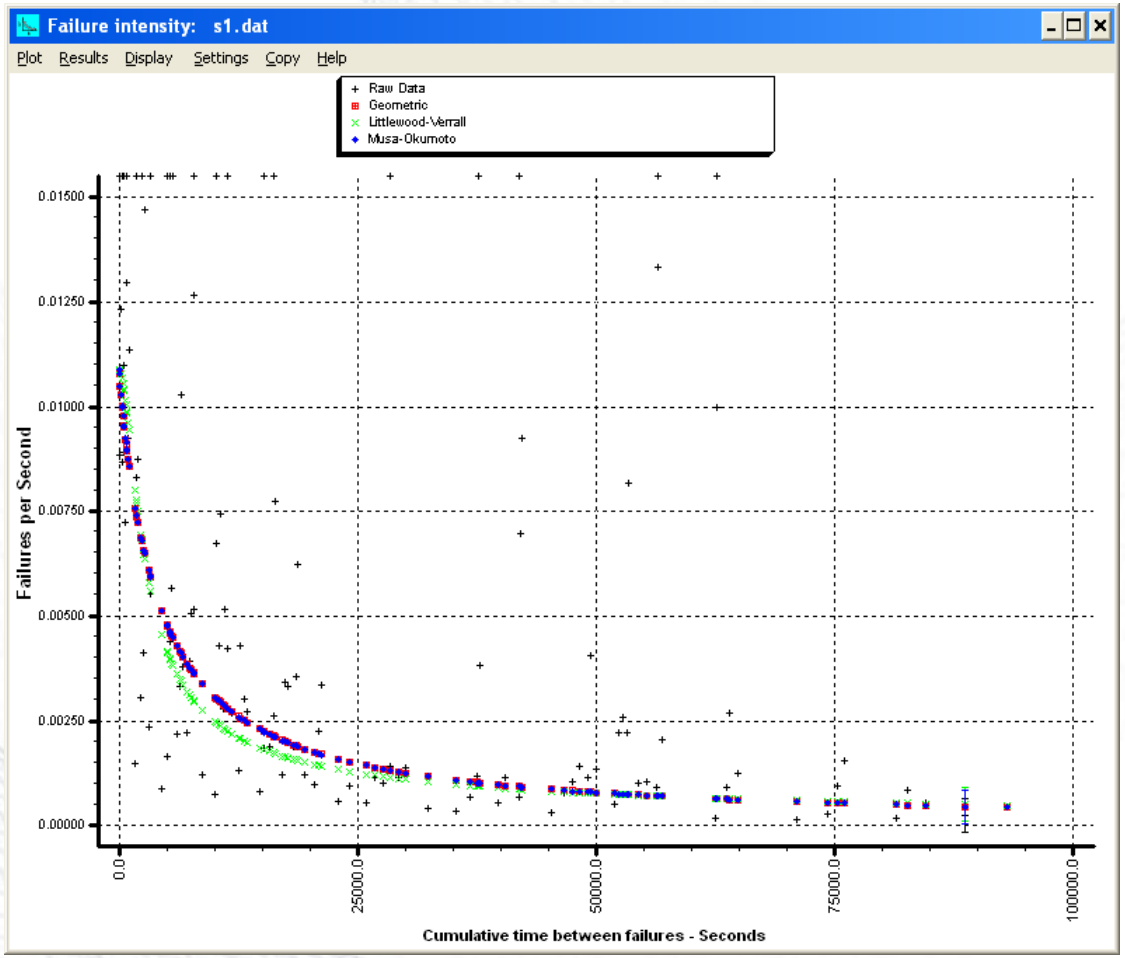
Display: Cumulative Failures



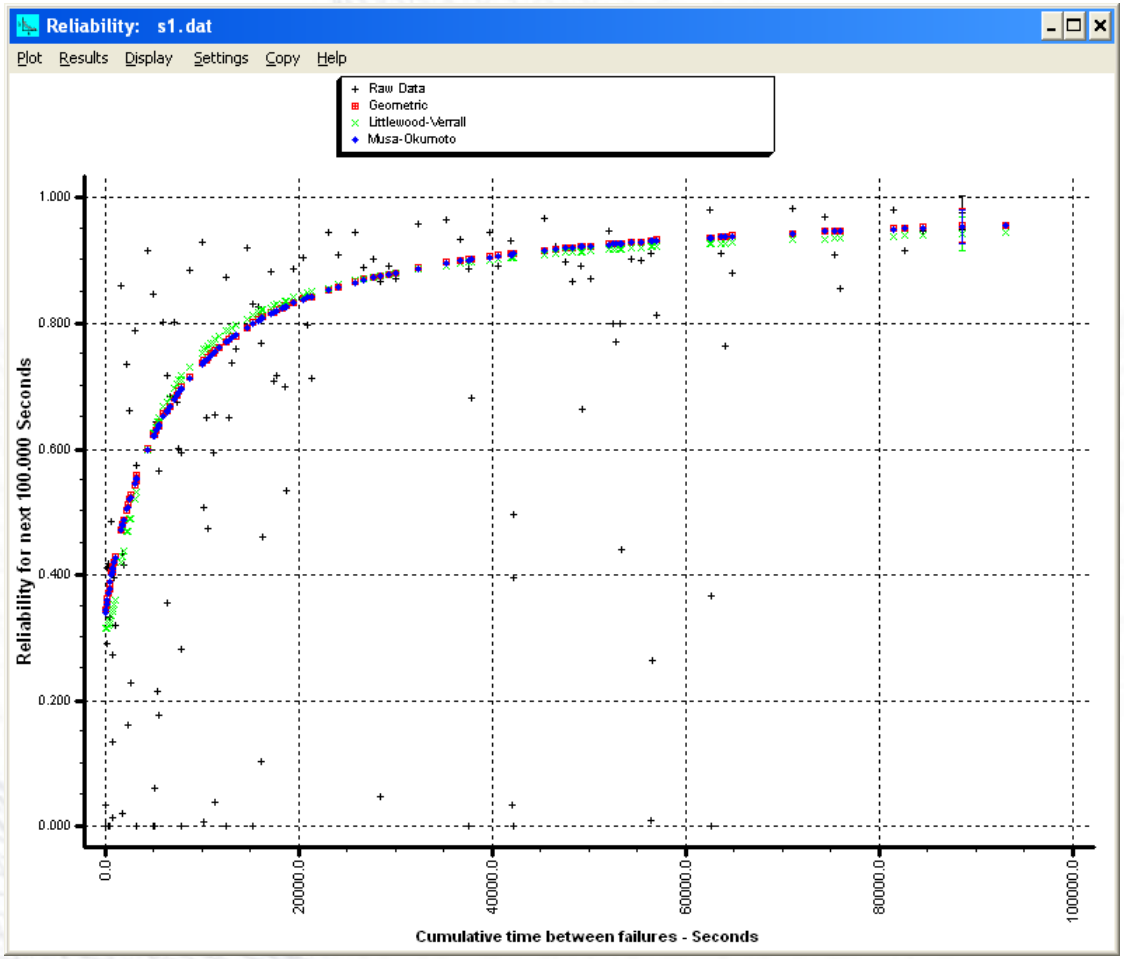
Display: Time Between Failures

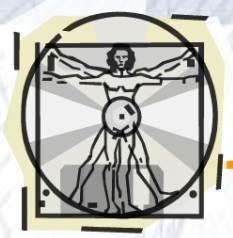


Display: Failure Intensity



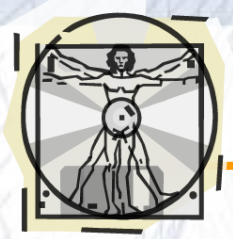
Display: Reliability





Interpreting Results /1

- Accuracy of estimation of the failure intensity λ depends on the number of failures experienced (i.e., the sample size)
- Good results in estimating failure intensity are generally experienced for programs with 5,000 or more developed source lines
- Satisfactory results are obtained for programs with 1,000 or more developed source lines



Interpreting Results /2

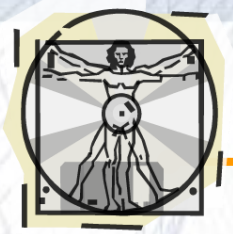
- When the failure rate λ at the end of testing period is very large and the trend indicates little chance of achieving the λ_F by the scheduled release date, what can be done?
  - Adding additional test and debugging resources
  - Adjusting the balance among the objectives for failure rate, development time, and development cost
  - Deferring some of the features until a future release

UNIVERSITY OF CALGARY

Section 5

Assessment Criteria:

Zero Failure Testing



Zero Failure Testing /1

- How many hours the system must be tested in order to meet a quality goal? ← "failure density" here
- Developed by Brettschnieder at Motorola

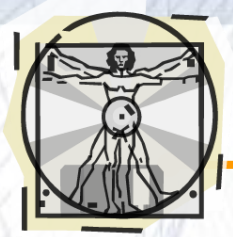
$$T_{zero} = \frac{\ln\left(\frac{n_F}{0.5+n_F}\right)}{\ln\left(\frac{0.5+n_F}{n_T+n_F}\right)} \times (T_{test} - T_{last})$$

Total test time

Time of last failure

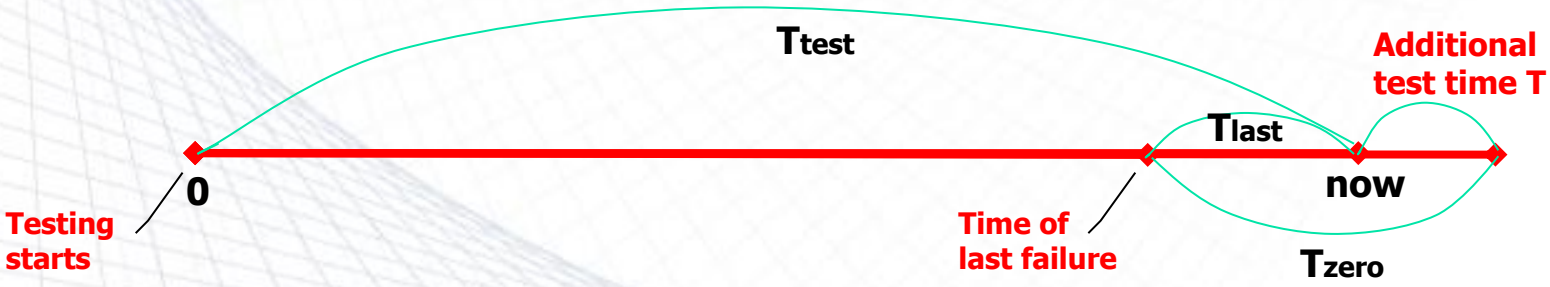
# of failures observed so far

Target # of failures



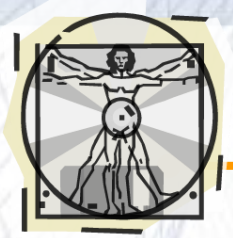
Zero Failure Testing /2

- Target projected number of failures (n_F) is calculated using "target failure density" and software "size" (KLOC)
- Number of repairable failures observed so far (n_T)
- Test execution hours since the last failure found ($T = T_{test} - T_{last}$)



Example

- Suppose you are testing a 33,000 LOC program, and to date 15 repairable failures have been detected over an execution time of 500 hours
- No failures have been observed in the last 50 hours of testing
- Management want to know how much more testing is needed to ensure that the customers observe no more than a projected average of 0.03 failures per KLOC



Example (cont'd)

- Calculate additional testing time

$$n_F = \frac{(0.03 \times 33,000)}{1,000} = 1$$

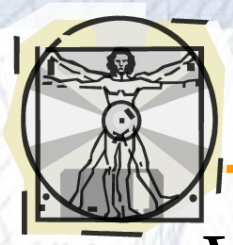
$$n_T = 15$$

$$T_{zero} = \frac{\ln\left(\frac{n_F}{0.5+n_F}\right)}{\ln\left(\frac{0.5+n_F}{n_T+n_F}\right)} \times (T_{test} - T_{last}) = \frac{\ln\left(\frac{1}{0.5+1}\right)}{\ln\left(\frac{0.5+1}{15+1}\right)} \times (500-50) = 77$$

$$\text{Additional test hours} = 77 - 50 = 27$$

Section 6

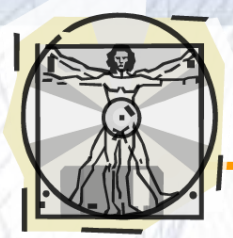
Assessment Criteria: Special Cases



Special Situations

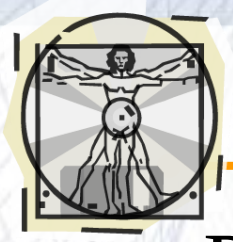
What else may affect λ/λ_F estimation?

- **Program evolution**
  - Dealing with small increments of program size and/or stepwise evolution (e.g. continuous integration)
- **Unreported failures**
  - How unreported failures affect failure intensity estimation?



Evolving Programs /1

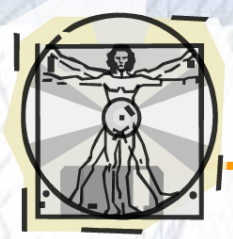
- Reliability models used in estimating failure intensity (λ) assume that the executed program is stable (not changing, except for those changes that result from failure correction)
- Programs can evolve due to:
  - Requirements changes
  - Integration of new parts
  - Necessity to adapt to changing hardware and software environment
  - Necessity for system performance improvement
  - Evolution as part of the development process



Evolving Programs /2

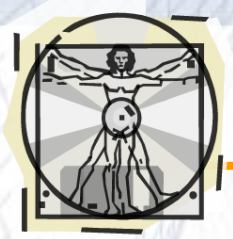
Possible scenarios:

- Ignoring changes for programs evolving slowly and in small size increments
- Ignoring old failure data of the old phase and base the estimates on the new phase
- Stepwise evolution
  1. Applying changes by components
  2. Applying changes by operation groups



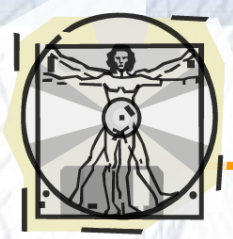
a) Ignoring Changes

- The effects of evolution can be ignored for programs evolving slowly and in small size increments (e.g., increments of less than %5 of total code per week)
- **Advantages:** No extra testing and data collection is required.
- **Disadvantages:** Estimates of model parameters and λ/λ_F ratio will lag the true situation and will have error, but the range of errors may be acceptable



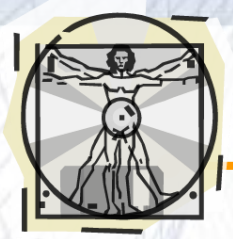
b) Ignoring Old Data

- Set the range of variation for the failure data to include only the recent failures
- The recent failure window must usually include at least 40-50 failures to estimate the parameters properly
- Only applicable when the changes in the program is limited (e.g., less than 20% of the programs change)



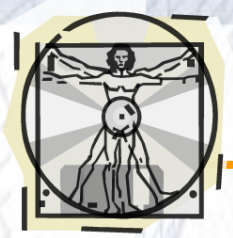
c) Stepwise Evolution /1

- Stepwise evolution approaches generally work best when having a *small number of large changes*, each resulting from the addition of an independent element (subsystems, packages, etc.)
- In ***component by component approach***, add the component failure intensities separately to obtain the system failure intensity at each stage
- In ***operation group approach***, add the weighted failure intensities of operation groups to obtain the system failure intensity at each stage



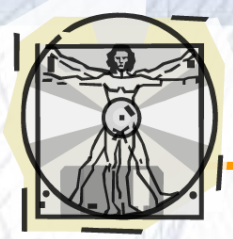
c) Stepwise Evolution /2

- Advantages and disadvantages of stepwise evolution approaches:
- **Advantages:** System operational profile can be applied directly
- **Disadvantages:**
  - Extra data collection required because of the multiple elements
  - Greater estimation error (or later achievement of a specified degree of accuracy) due to smaller sample sizes



Other Evolution Scenarios

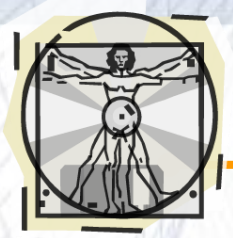
- Additional situations analogous to program evolution:
  - A program is fully integrated, but it is tested in an evolutionary fashion. One component or operation group is turned on, then another, and so on
  - A program is fully integrated and active, but the observation of the components for failures is handled in an evolutionary fashion. One starts with a narrow focus of observation and then widens it component by component or operation group by operation group



Unreported Failures /1

- Failures may not be reported, in cases such as
  - Less severe failures
  - Failures not halting program execution
- More failures may be missed in integration test than in regression test because the exact inputs and outputs during integration test may be unknown
- Missing to report failures means underestimating actual failure intensity (λ)

How unreported failures affect failure intensity estimation?

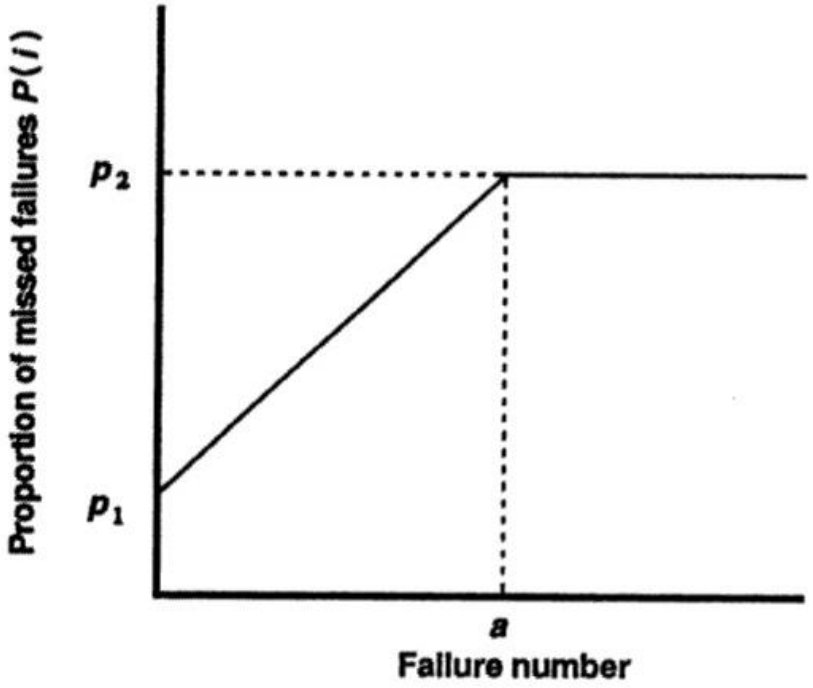


Unreported Failures /2

$$P(i) = P_1 + \frac{i}{a}(p_2 - p_1) \qquad i < a$$

$$P(i) = P_2 \qquad i \geq a$$

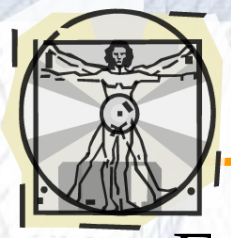
$P(i)$ is the probability that the i th failure goes unnoticed



Assumption:

Initially, the probability of not observing a particular failure is P_1 and as time goes by, this increases linearly with a constant change rate a until reaching to its final value P_2

Figure from Musa's Book



Unreported Failures /3

For the basic exponential model:

- Expected failure intensity at execution time τ

$$\lambda(\tau) = \lambda_0 e^{\left(-\frac{\lambda_0}{\nu_0}\right)\tau}$$

- The plot for $\hat{\lambda}(\tau)/\lambda(\tau)$

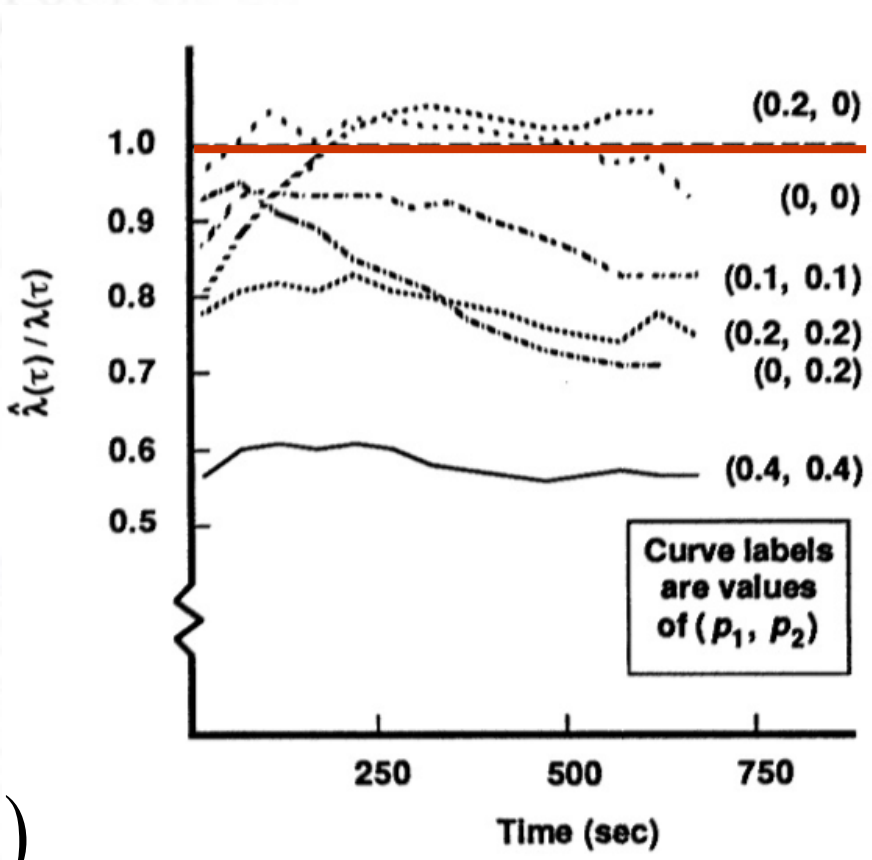
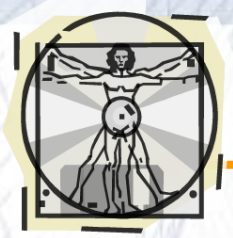


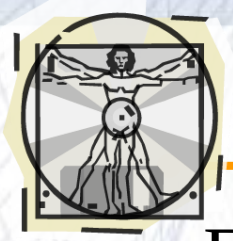
Figure from Musa's Book



Unreported Failures /4

- For the basic exponential model:
- For $p_1=p_2=p$ estimates for failure intensity fluctuates around *1-p*
- If *P(i)* increases with time, the estimate for λ is low and becomes lower as time passes
- If *P(i)* decreases with time, the estimate for λ is initially low but improves as time passes

- Correction is possible



Unreported Failures /5

For the logarithmic Poisson model:

- Expected failure intensity at execution time τ

$$\lambda(\tau) = \frac{\lambda_0}{\lambda_0 \theta \tau + 1}$$

- The plot for $\hat{\lambda}(\tau) / \lambda(\tau)$

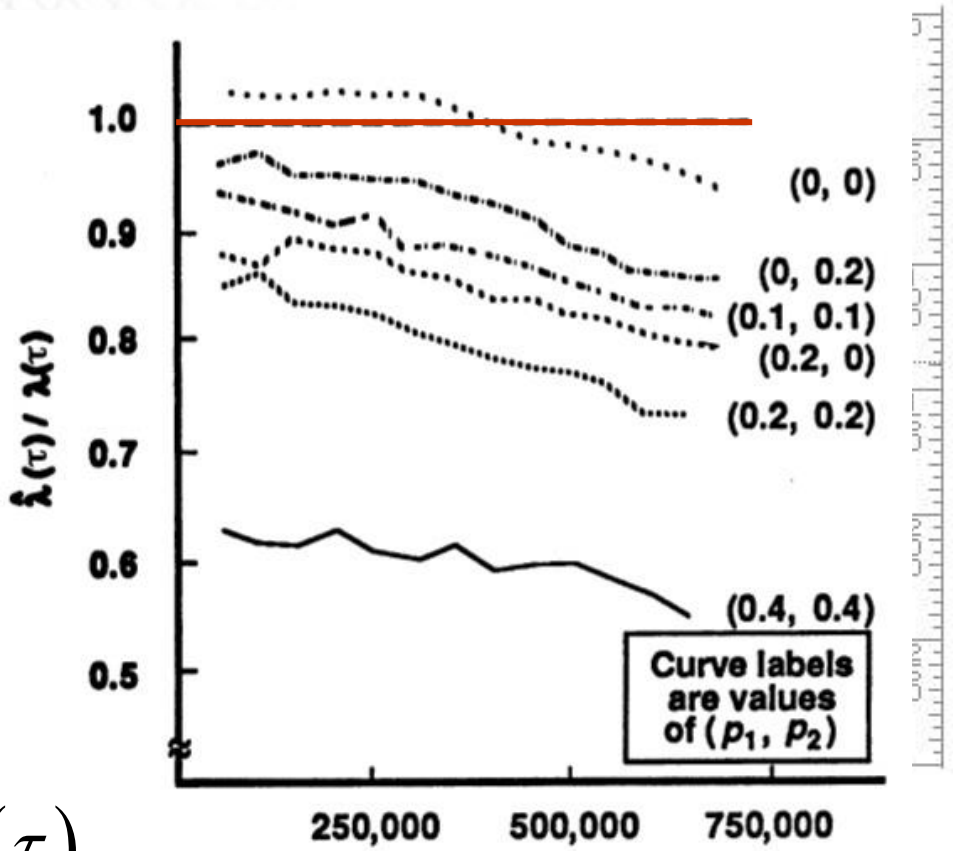
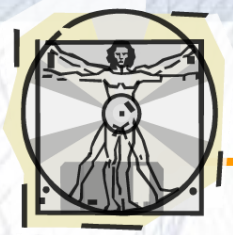
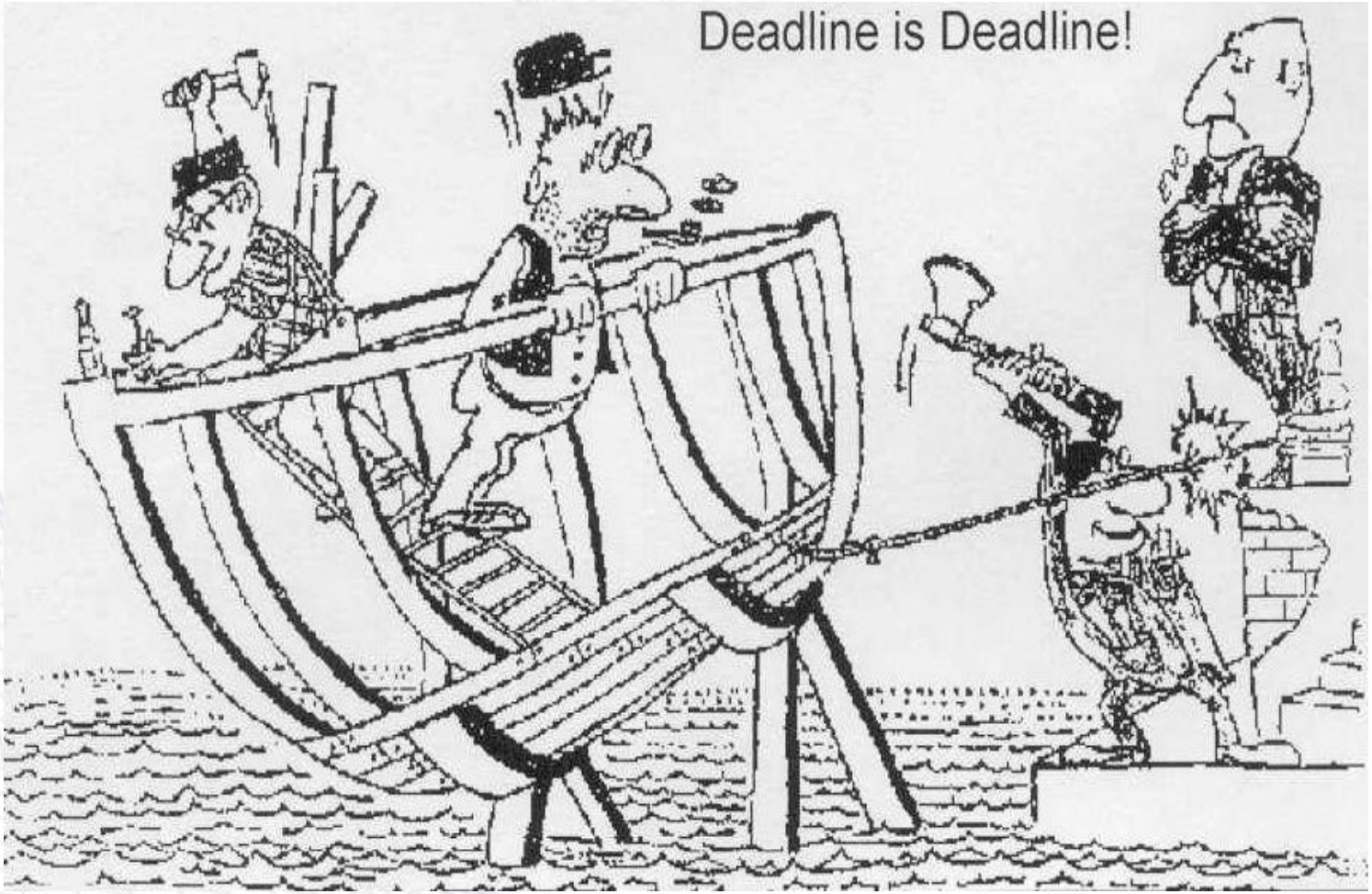


Figure from Musa's Book



Unreported Failures /6

- For the logarithmic Poisson model:
- For $p_1=p_2=p$ estimates for failure intensity fluctuates around *1-p*
- The estimate for λ is low and becomes lower as time passes
- Correction is possible but not as good as the basic model


Deadline is Deadline!