

CS641b

Advanced Topics in Software Quality Assurance

The University of Western Ontario/ConGESE

Dr. Mechelle Gittens

Lecture VI – March 23rd

The University of Western Ontario

*Adapted from Defect Prediction Workshop - CASCON 2005 –
with Dr. Bob Probert, Andriy Miransky and Kyle Robeson*

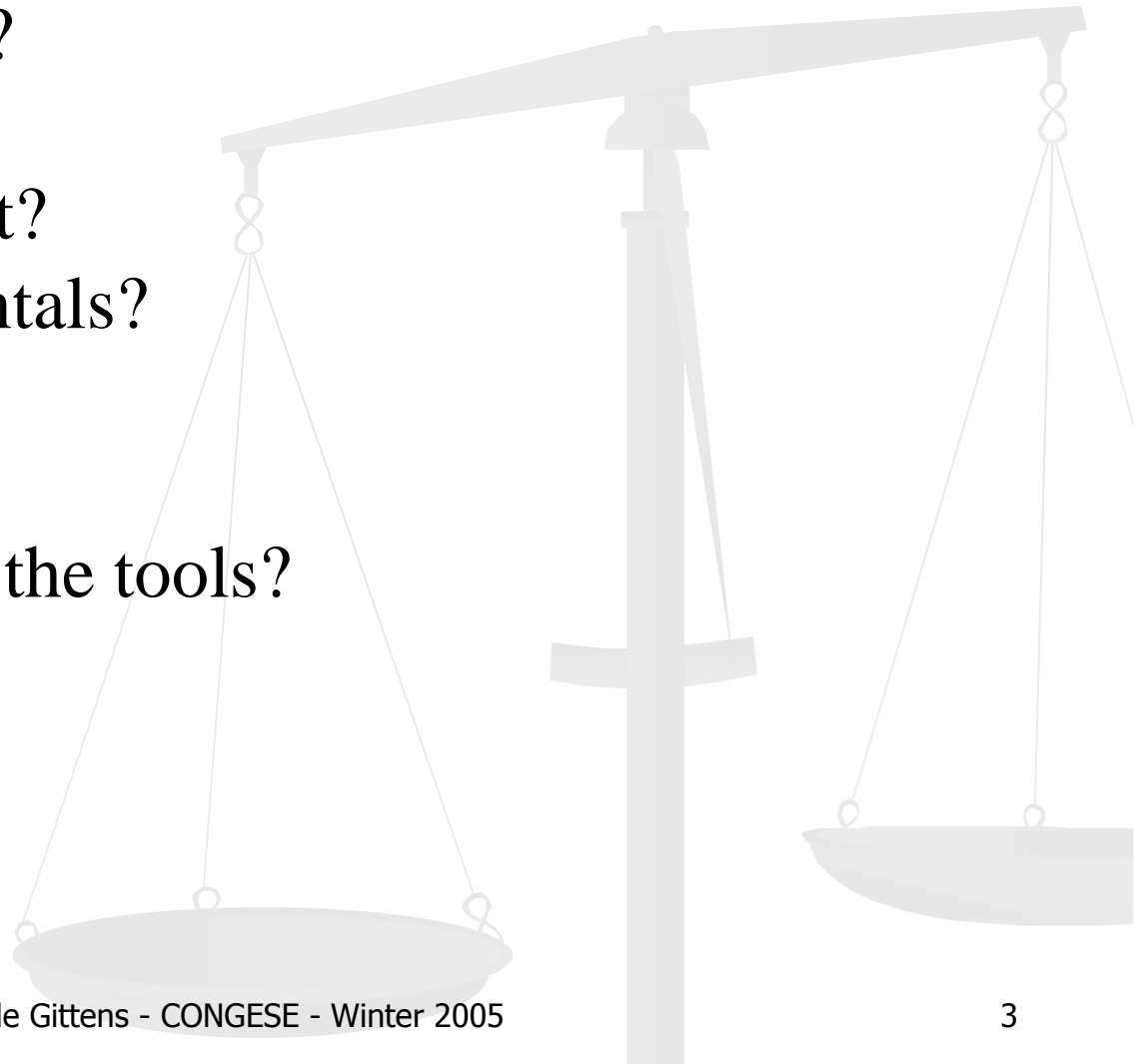
Today's Question

How do you know if your testing method is producing a better product?



Today's Class

- Why defect prediction?
- Can we measure it?
- How can we measure it?
- What are the fundamentals?
- How do we predict?
- What are the models?
- Case Study - What are the tools?
- Web-based testing
- Next Class



Why defect prediction?

- How can you be certain that your intuition is correct?
- What can you measure and track cost-effectively to quantify test progress and compare it to other releases in the product family?

Why defect prediction?

- Make informed choices as development progresses
- Decide when release should occur
- Better manage resources to deal with possible defects
- Create a mitigation plan for costs of field defect occurrences
- Following examples are taken from: *Forecasting Field Defect Rates Using a Combined Time-based and Metrics-based Approach: a Case Study of OpenBSD – Paul Luo li et. al.*

How to predict

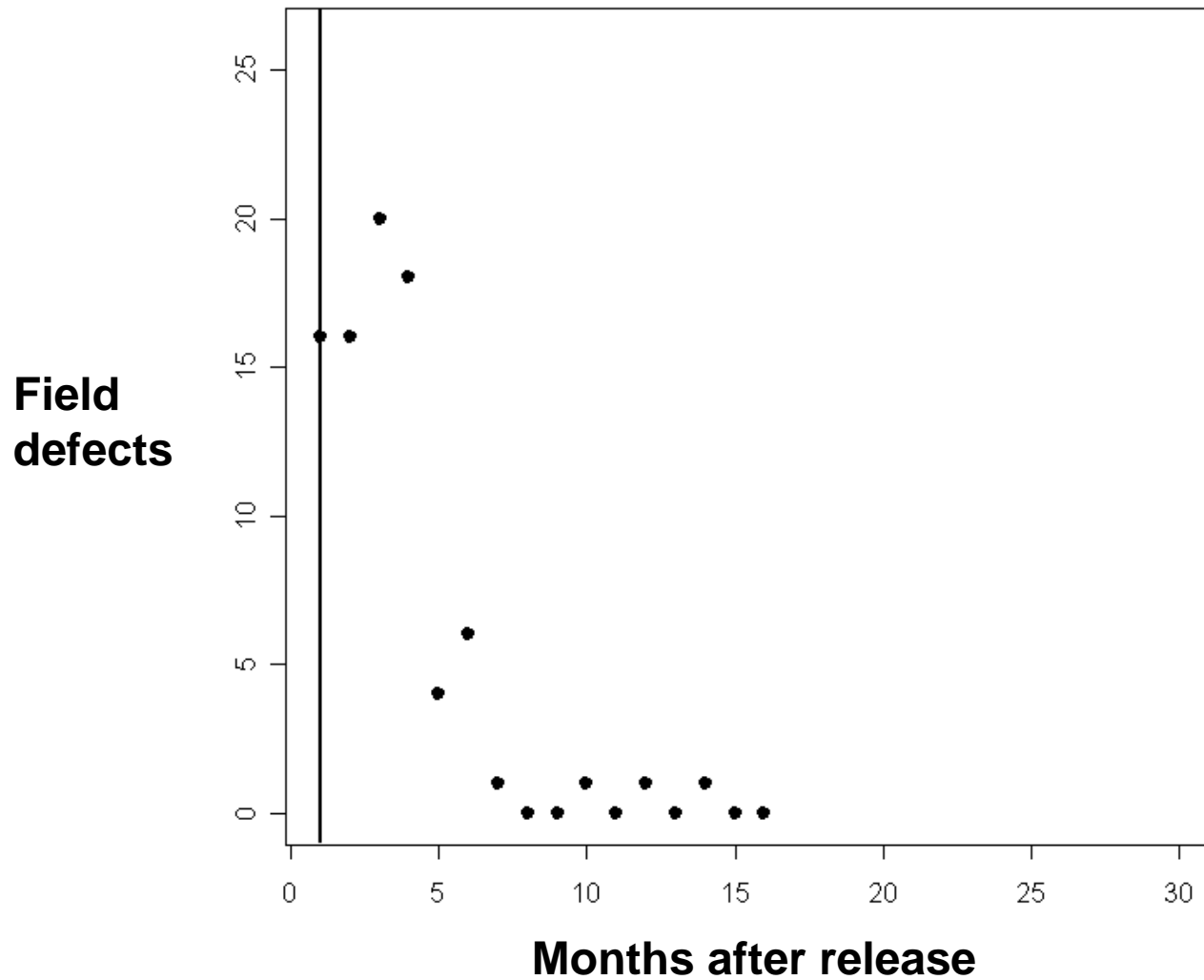
- *Must measure/record to predict*
 - *Record the trend in rate of defects*
 - *Fit to a Software Reliability Model (SRM)*
- *Defect arrival rates:*
 - Weekly defect arrivals
 - Weekly defect arrivals per KCSI
- *Defect backlog:*
 - The number of defects remaining at any given time. i.e., the difference between the defects detected and defects that are closed

How to predict

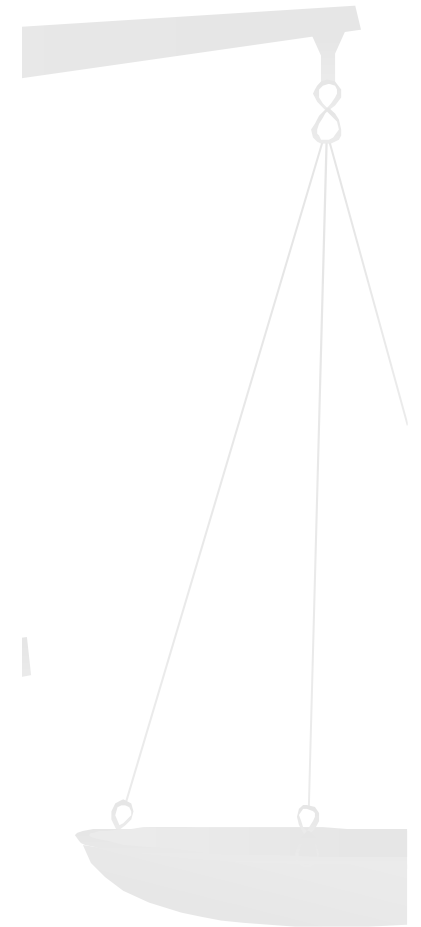
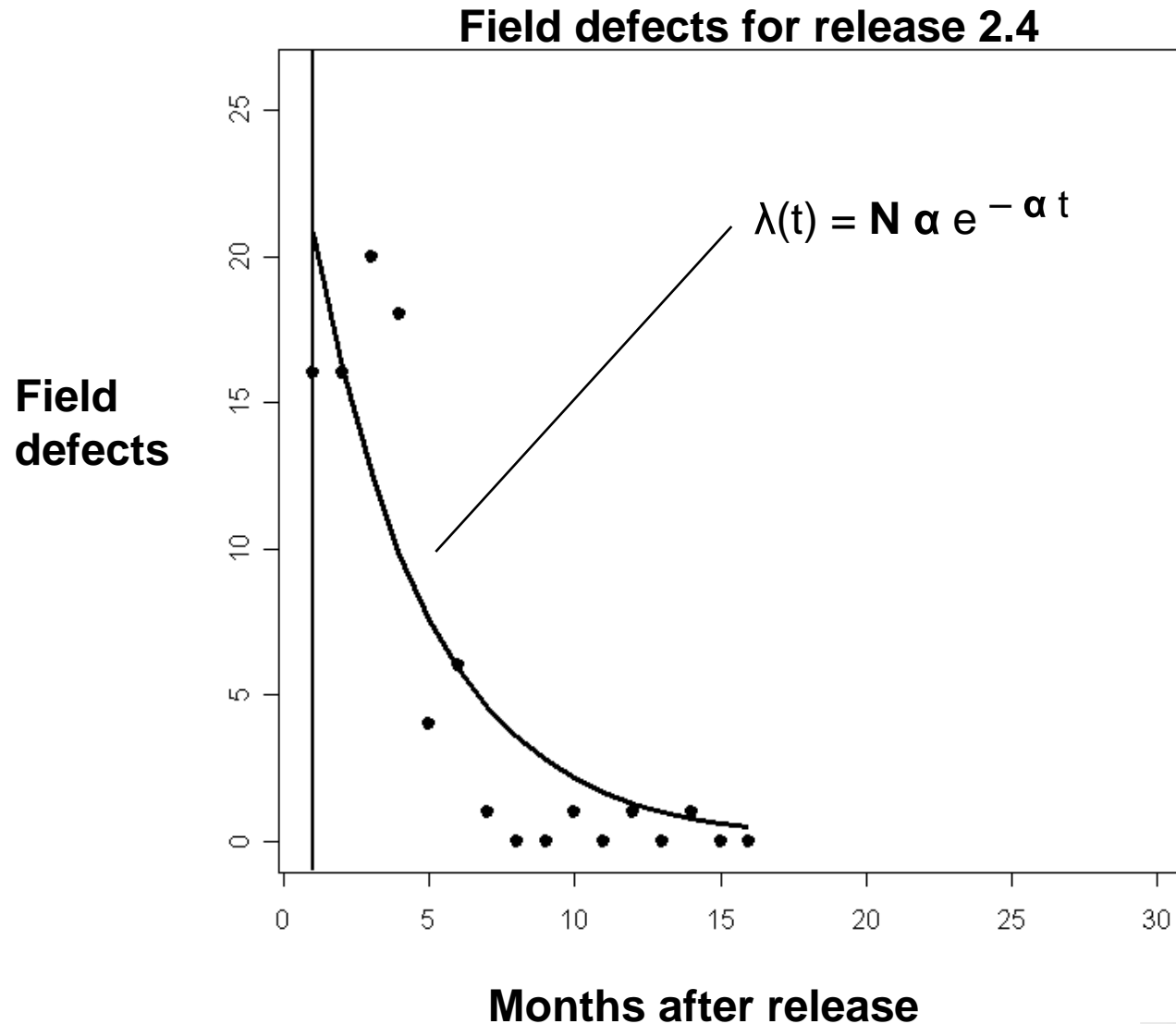
- Identify key factors (variables)
- Curve-fitting: Continuous Models based on statistical distributions (e.g. Rayleigh model, Exponential model)
- Model analysis, regression (e.g. goodness of fit)

Record the rate of field defects

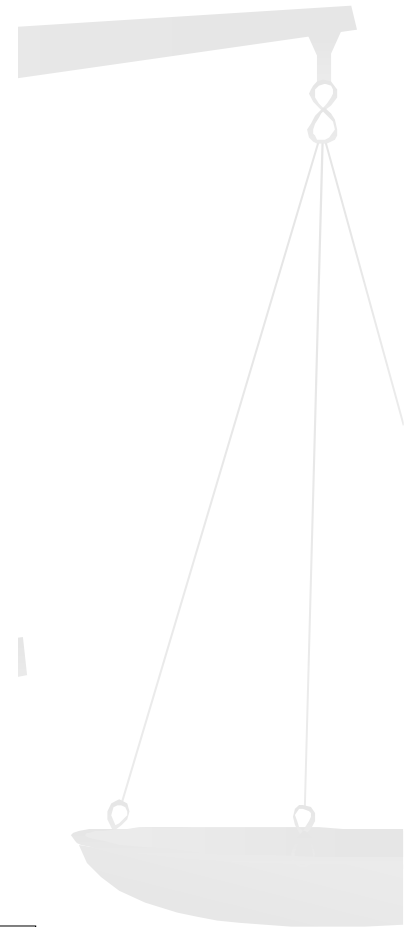
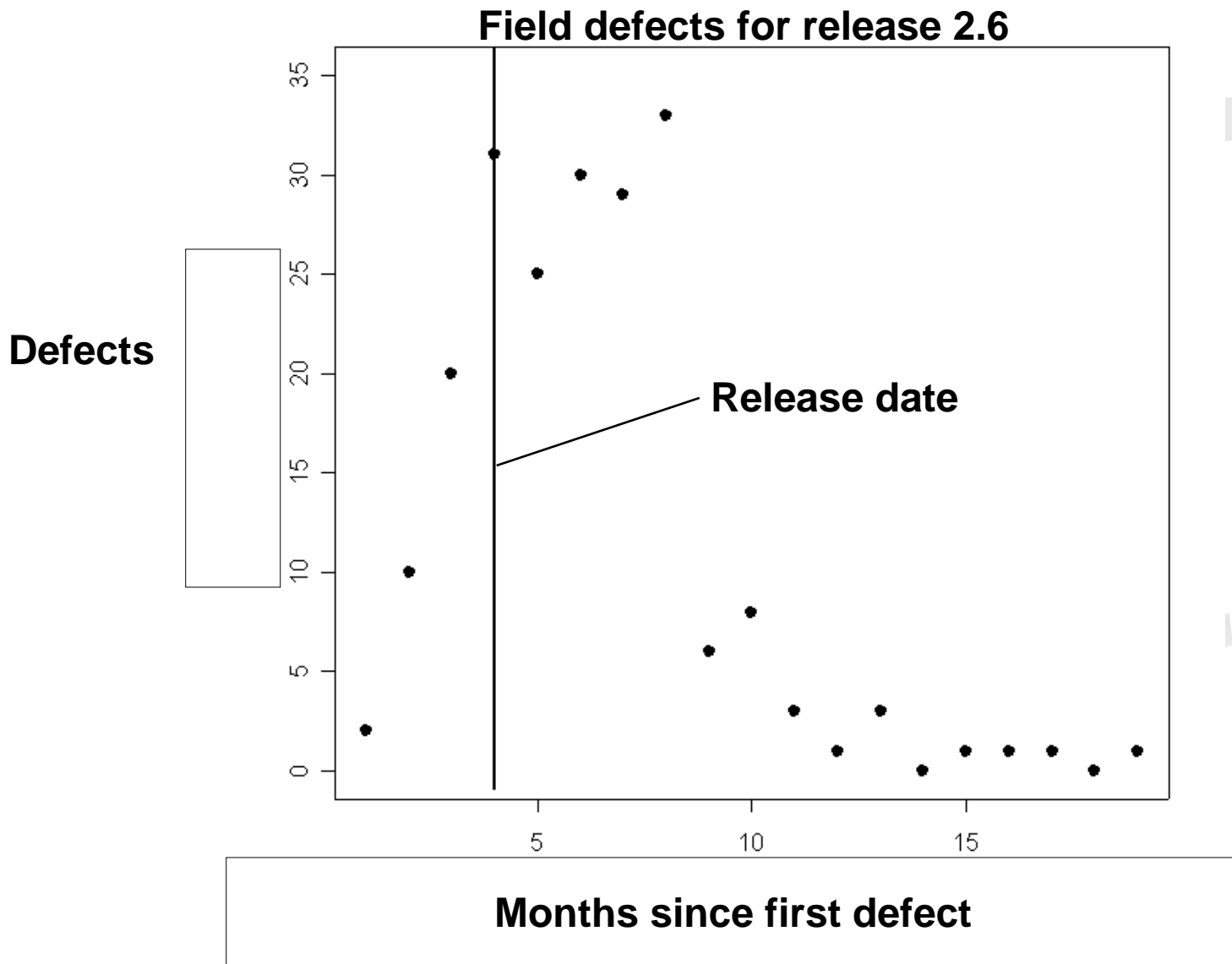
Field defects for release 2.4



Fit to model parameters of an SRM



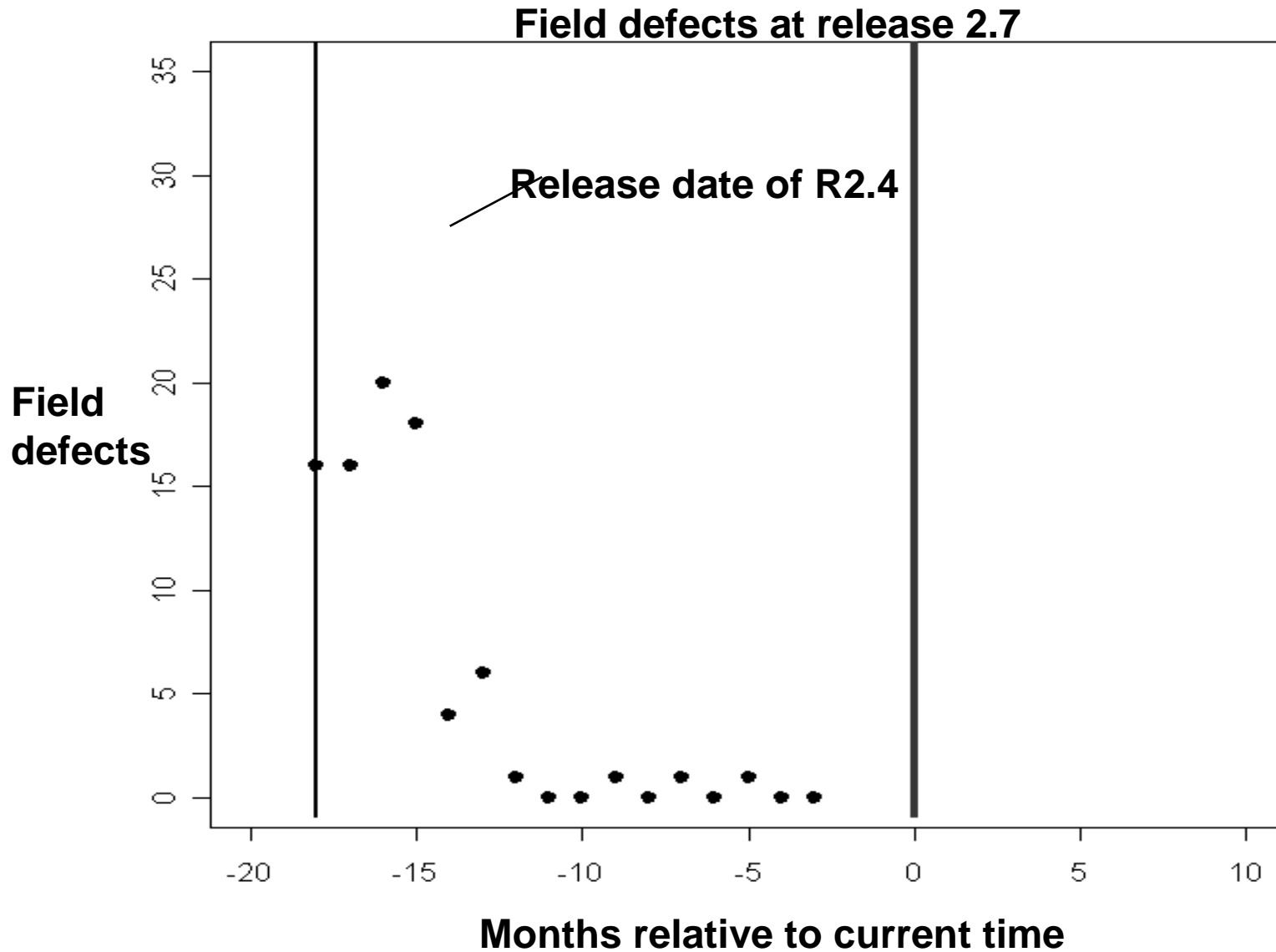
Step 1: Look at all defects



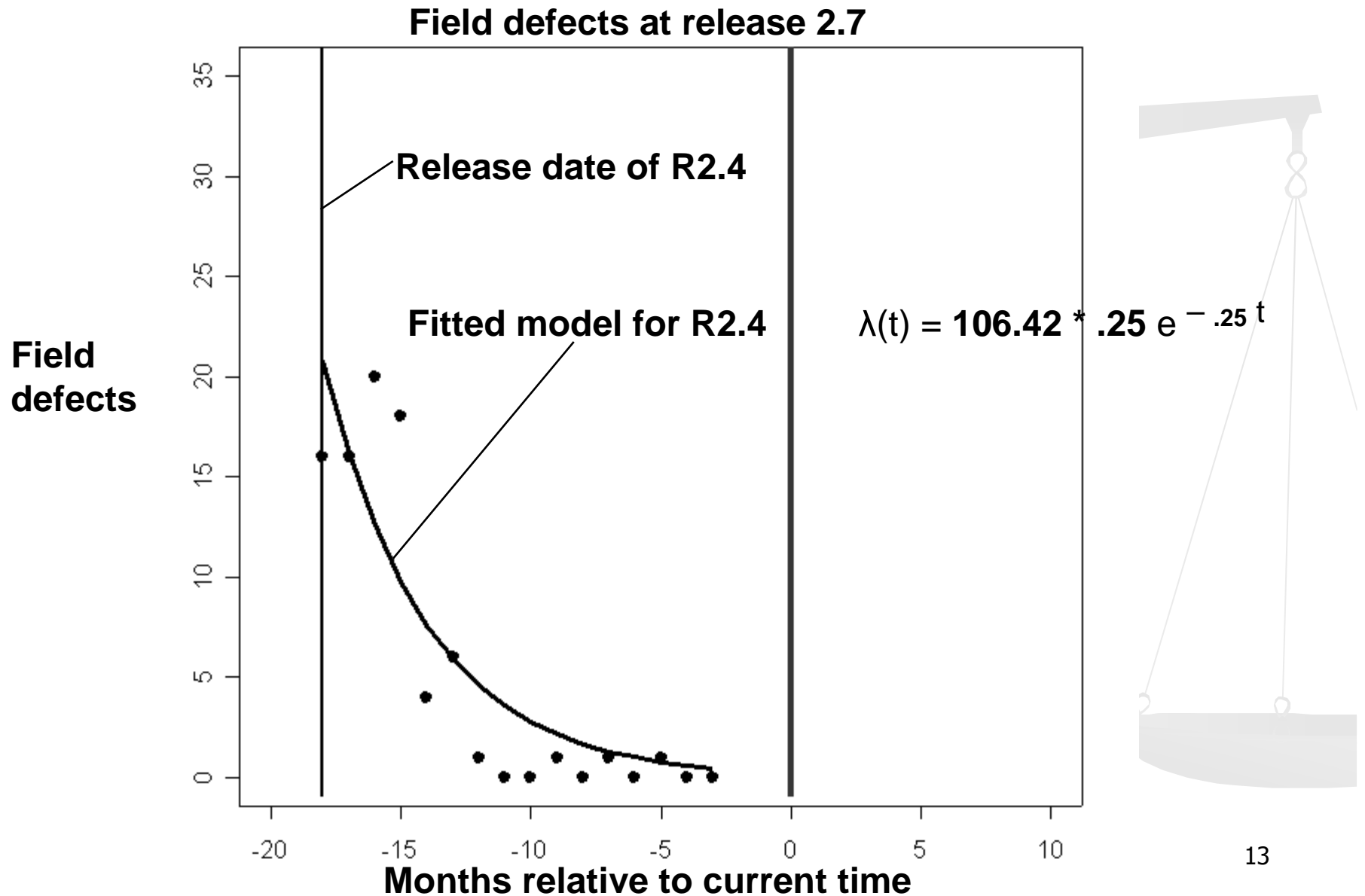
Predict Using “on-the-go” data

- Fit to a model using data from a historical release to predict for future releases
- Estimate the model parameters for active historical releases using a software reliability model and any field defect data available at the time of release

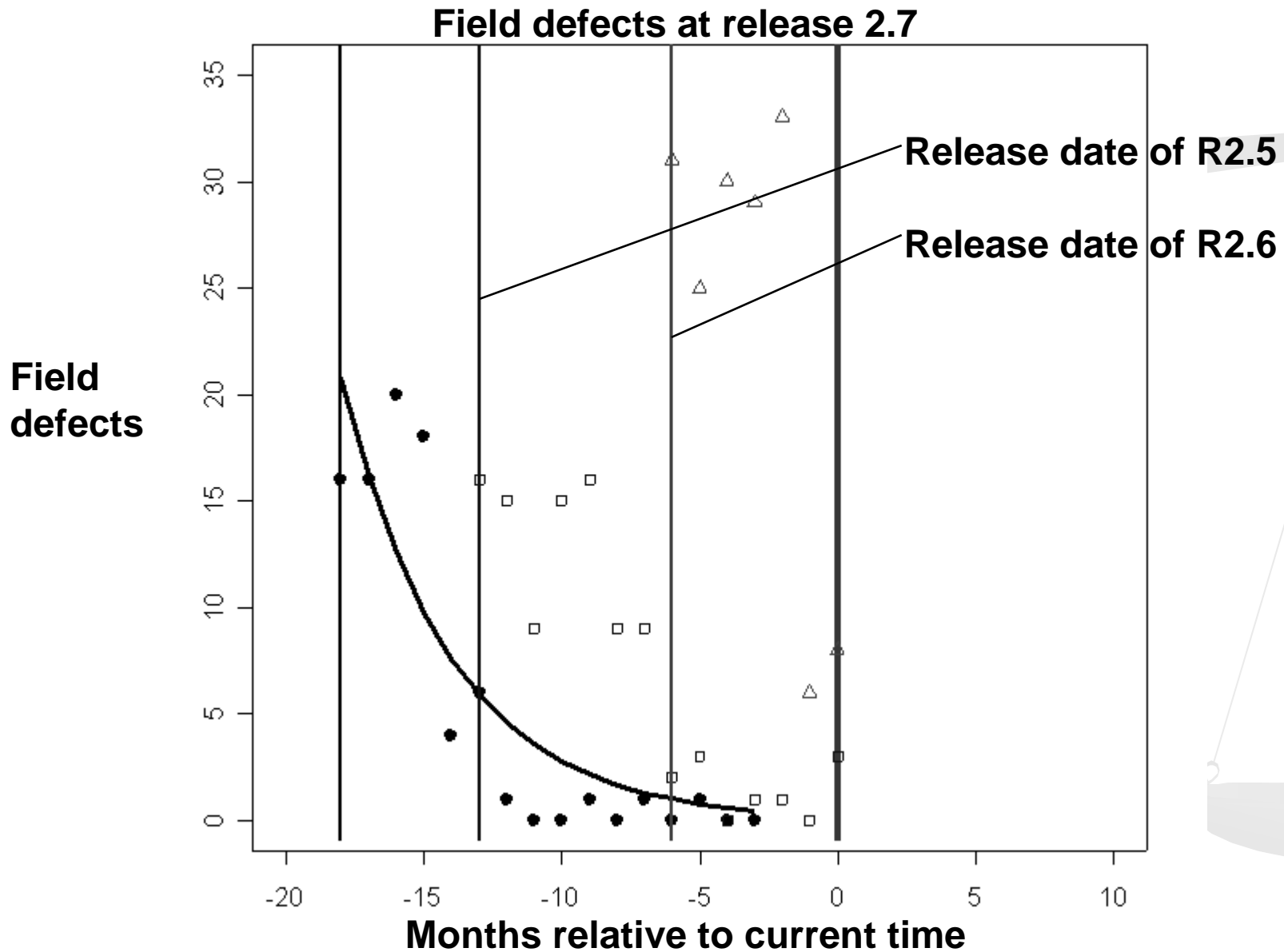
Step 2: Look at historical releases



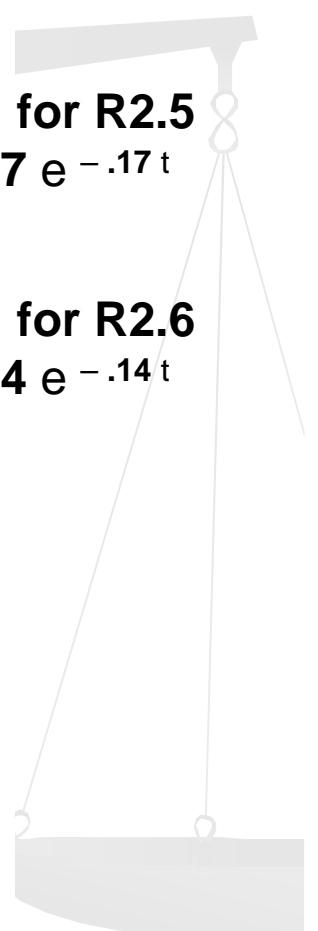
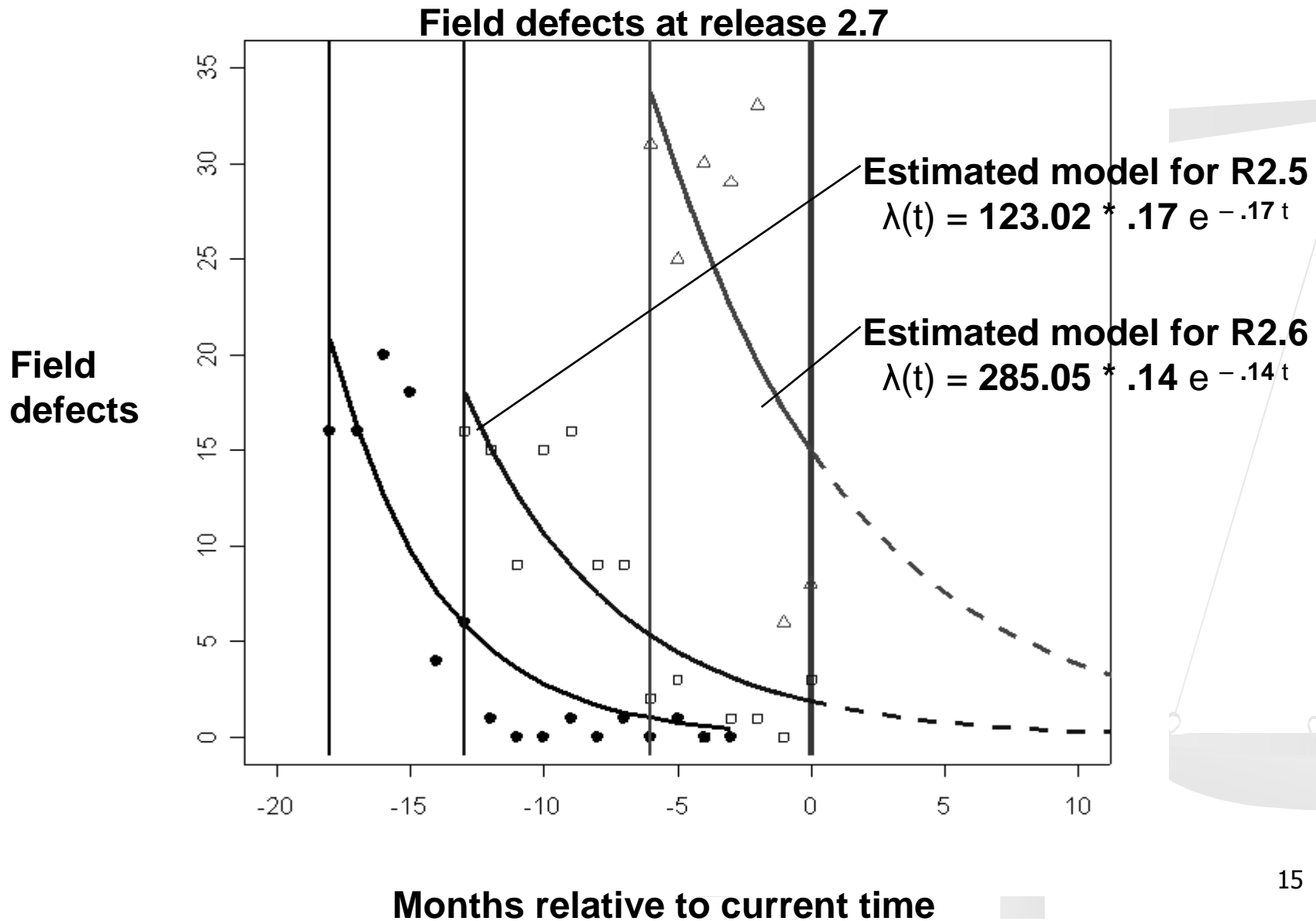
Step 3: Fit established model to data



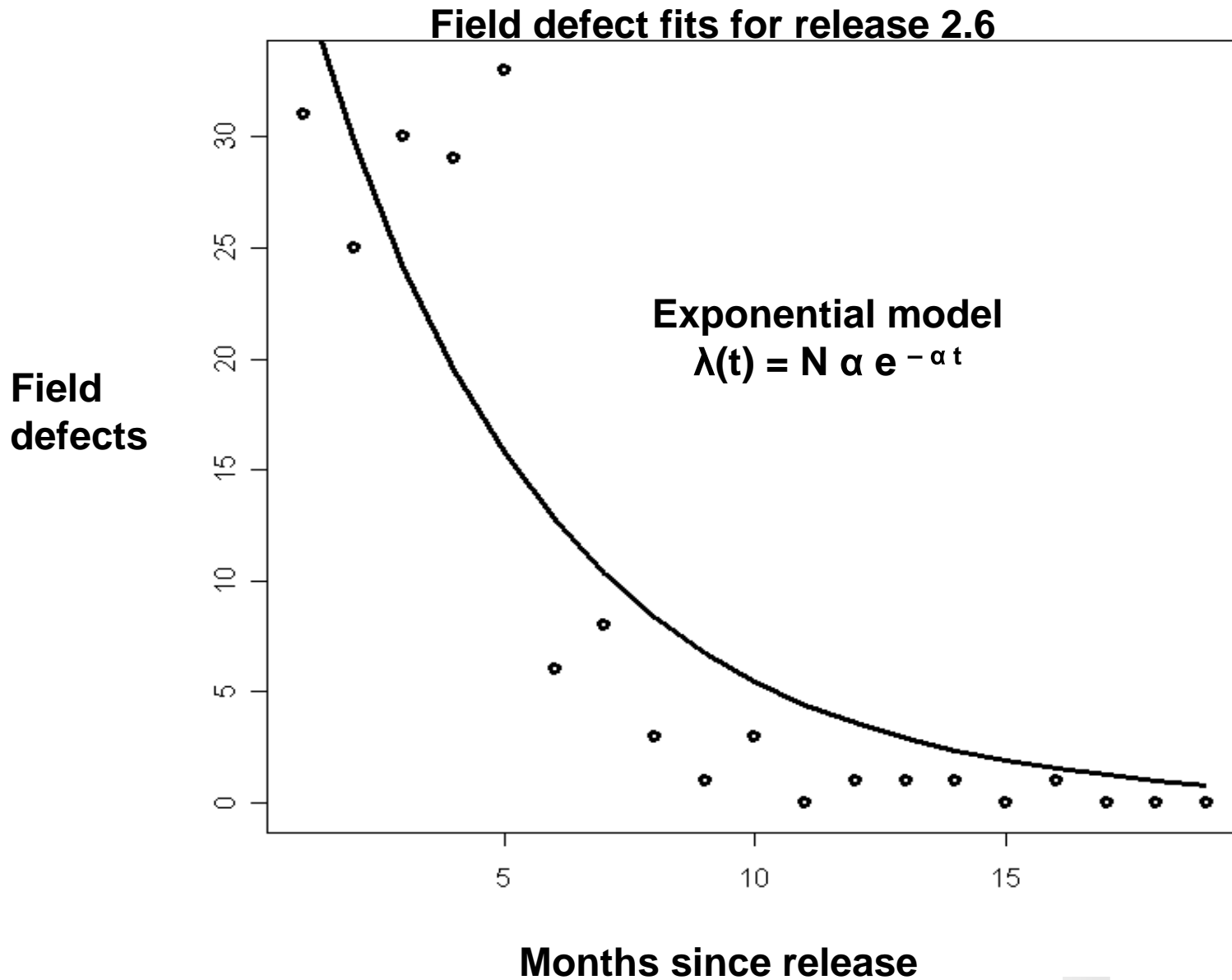
Other historical releases will be active



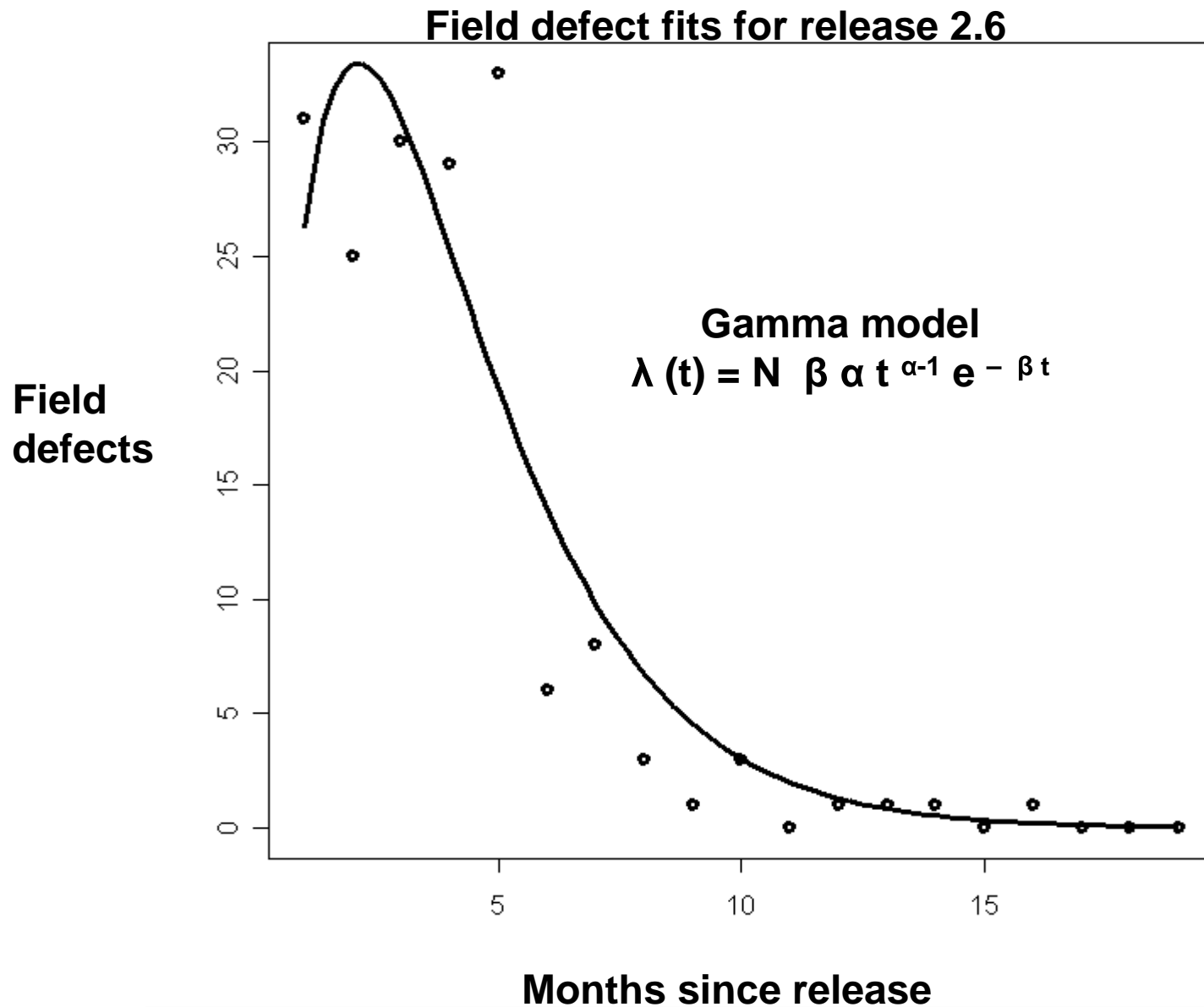
Step 4: Use data available at release



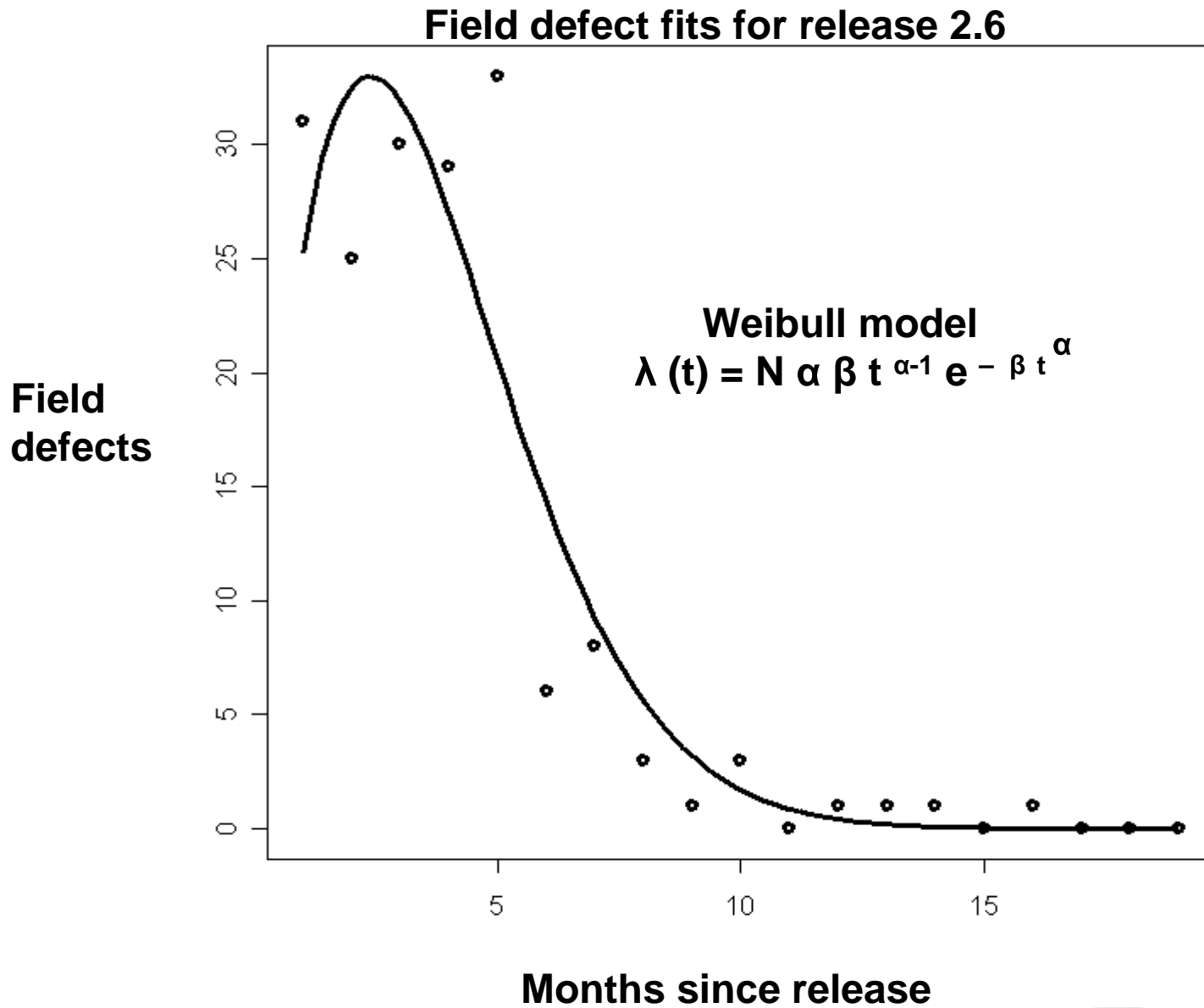
Step 5: Vary model to achieve best fit



Step 5: Vary model to achieve best fit

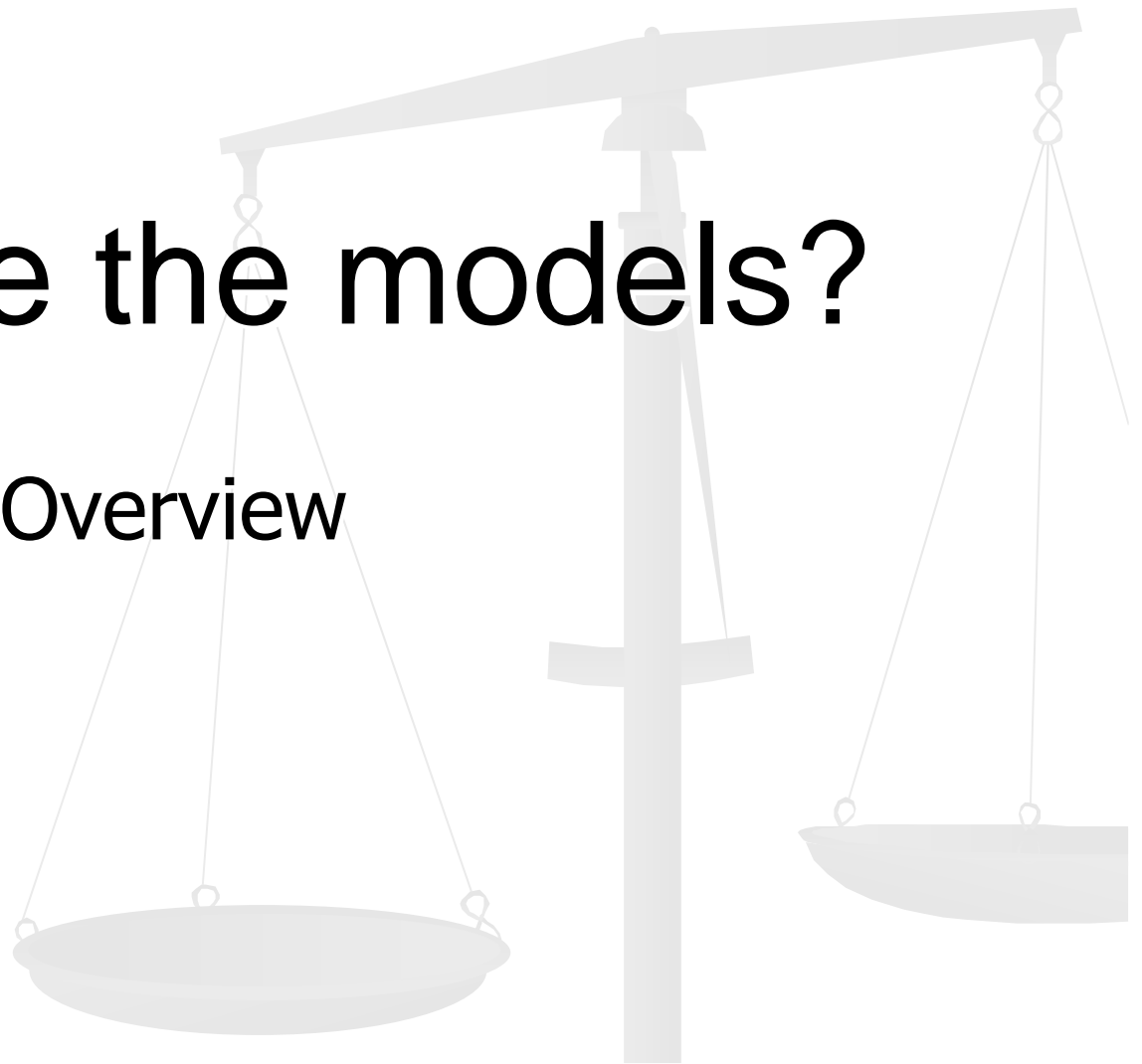


Weibull Model = best fit

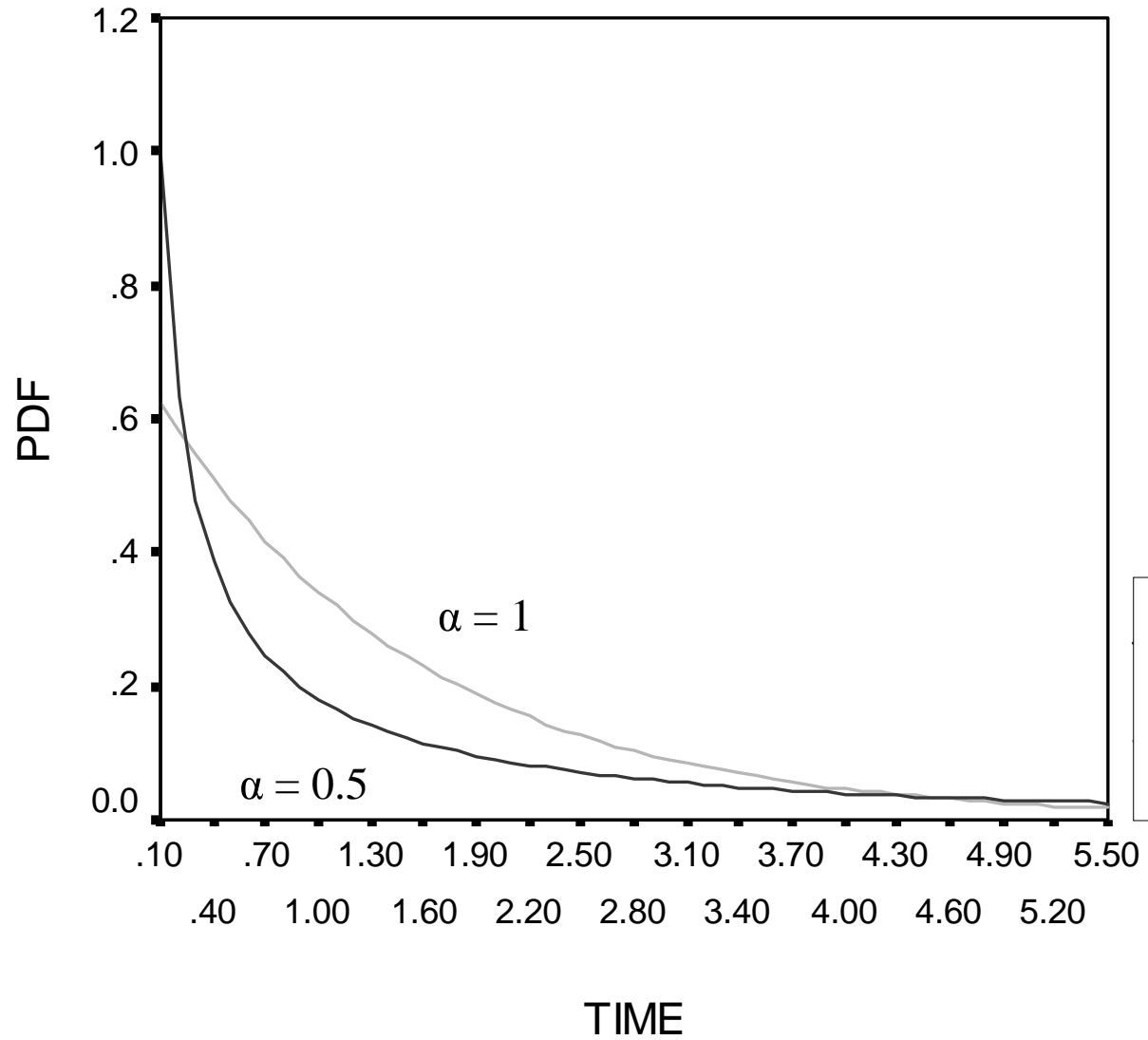


What are the models?

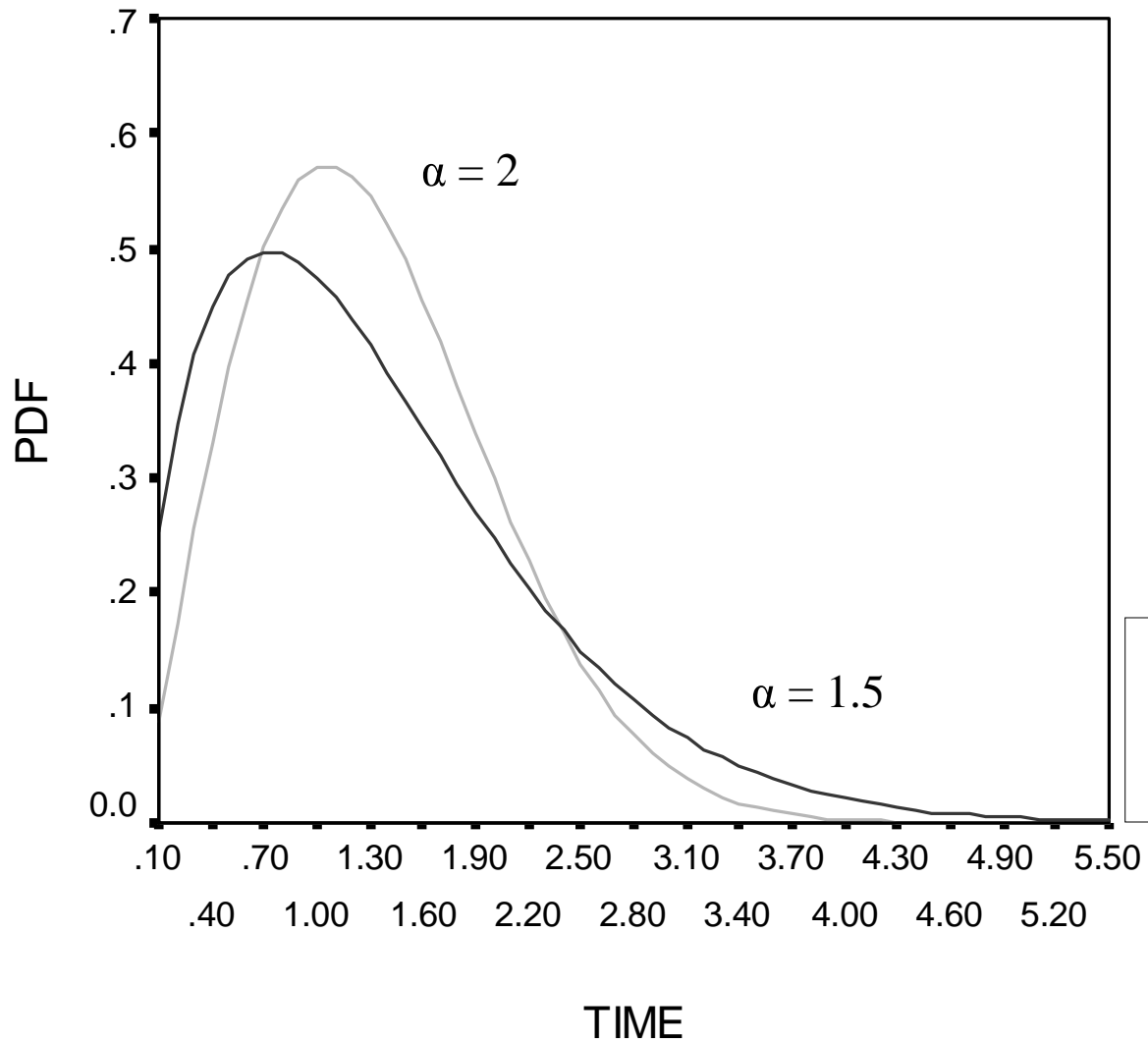
Overview



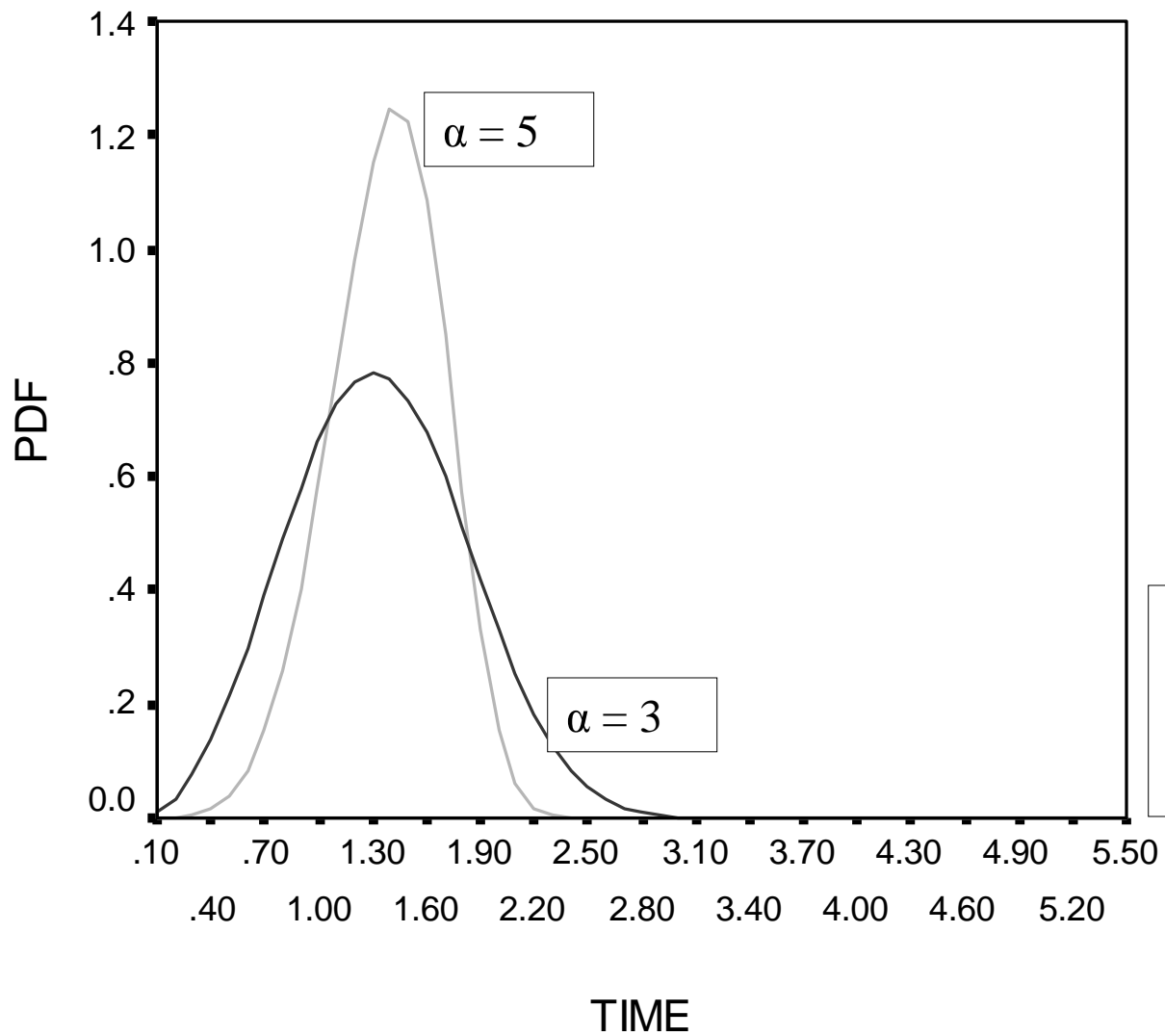
Other examples of the Weibull curve $\alpha = 0.5, 1$



Other examples of the Weibull curve $\alpha = 2, 1.5$

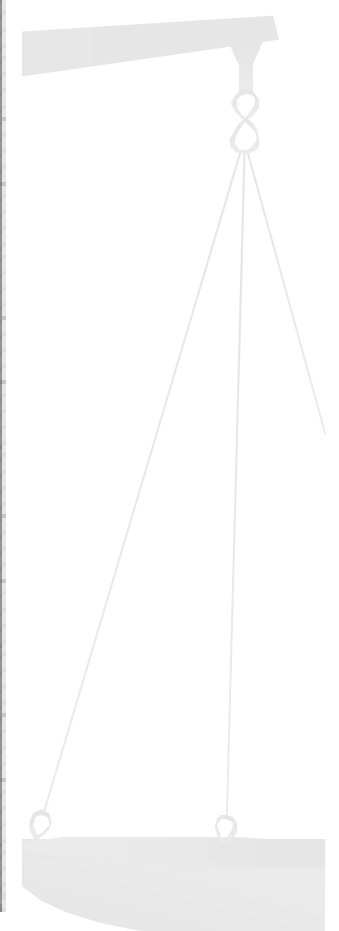
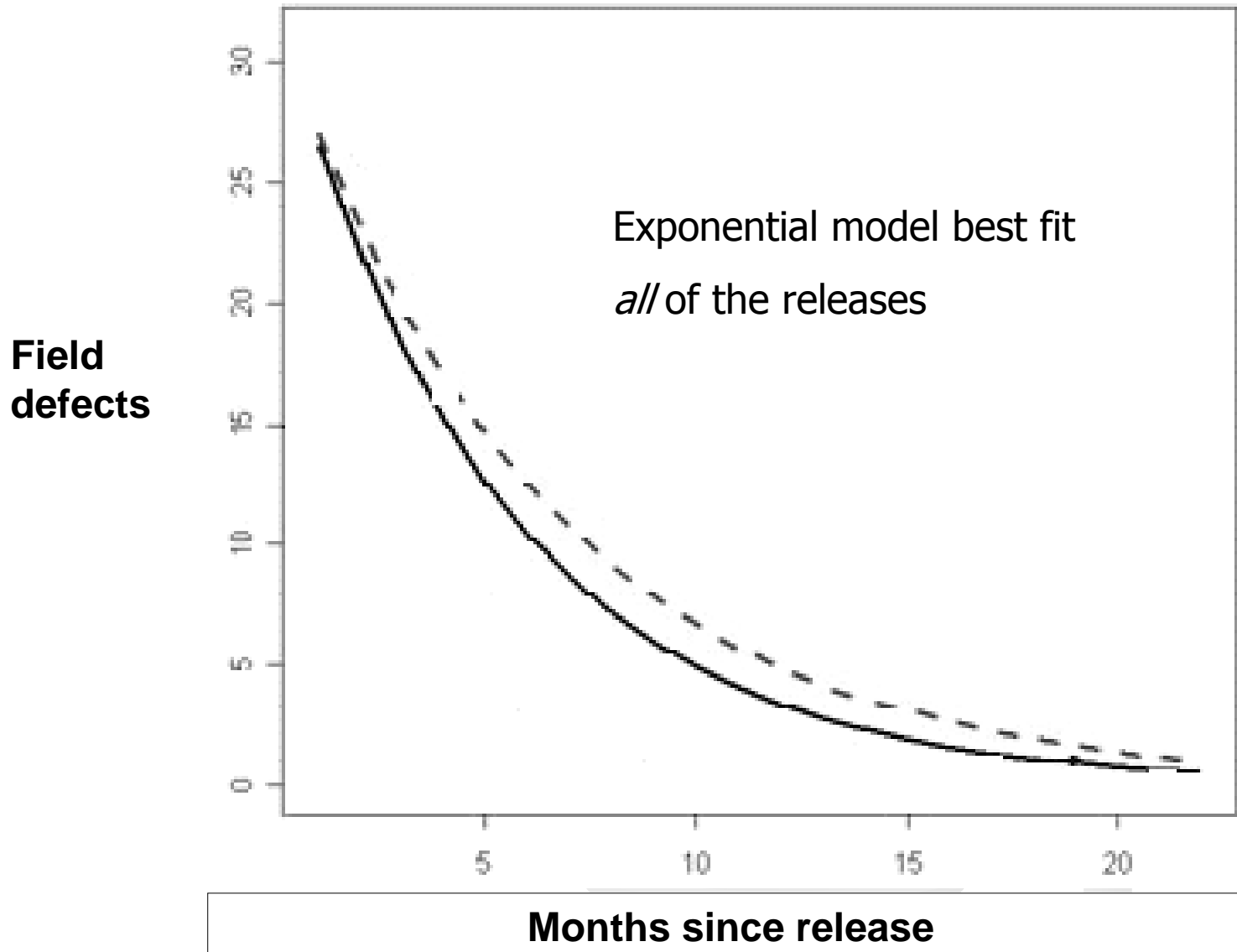


Other examples of the Weibull curve $\alpha = 3, 5$



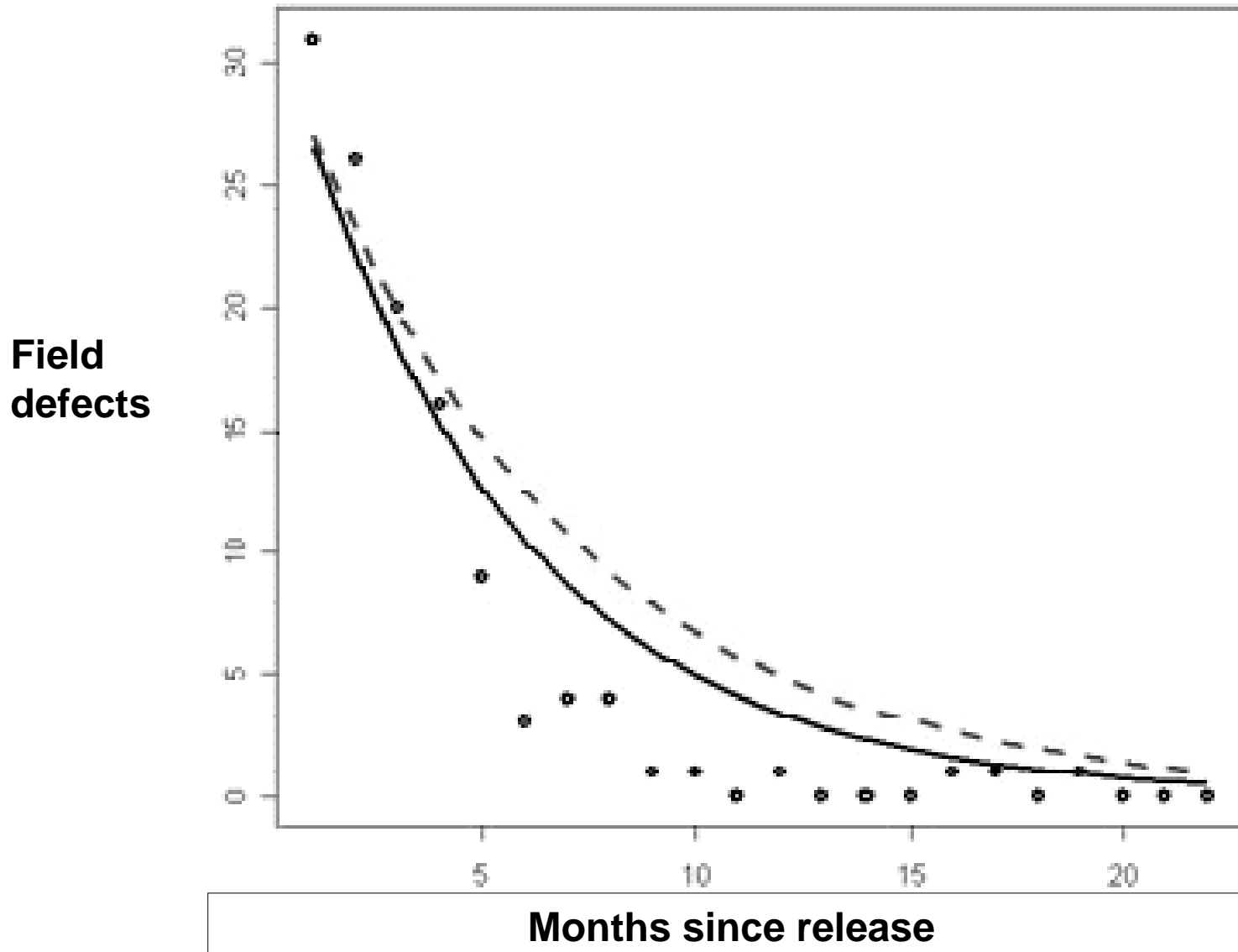
Step 6: Use best fit to forecast for next release

Field defect forecasts for release 2.7



Step 7: Track best fit against actuals

Field defect forecasts for release 2.7

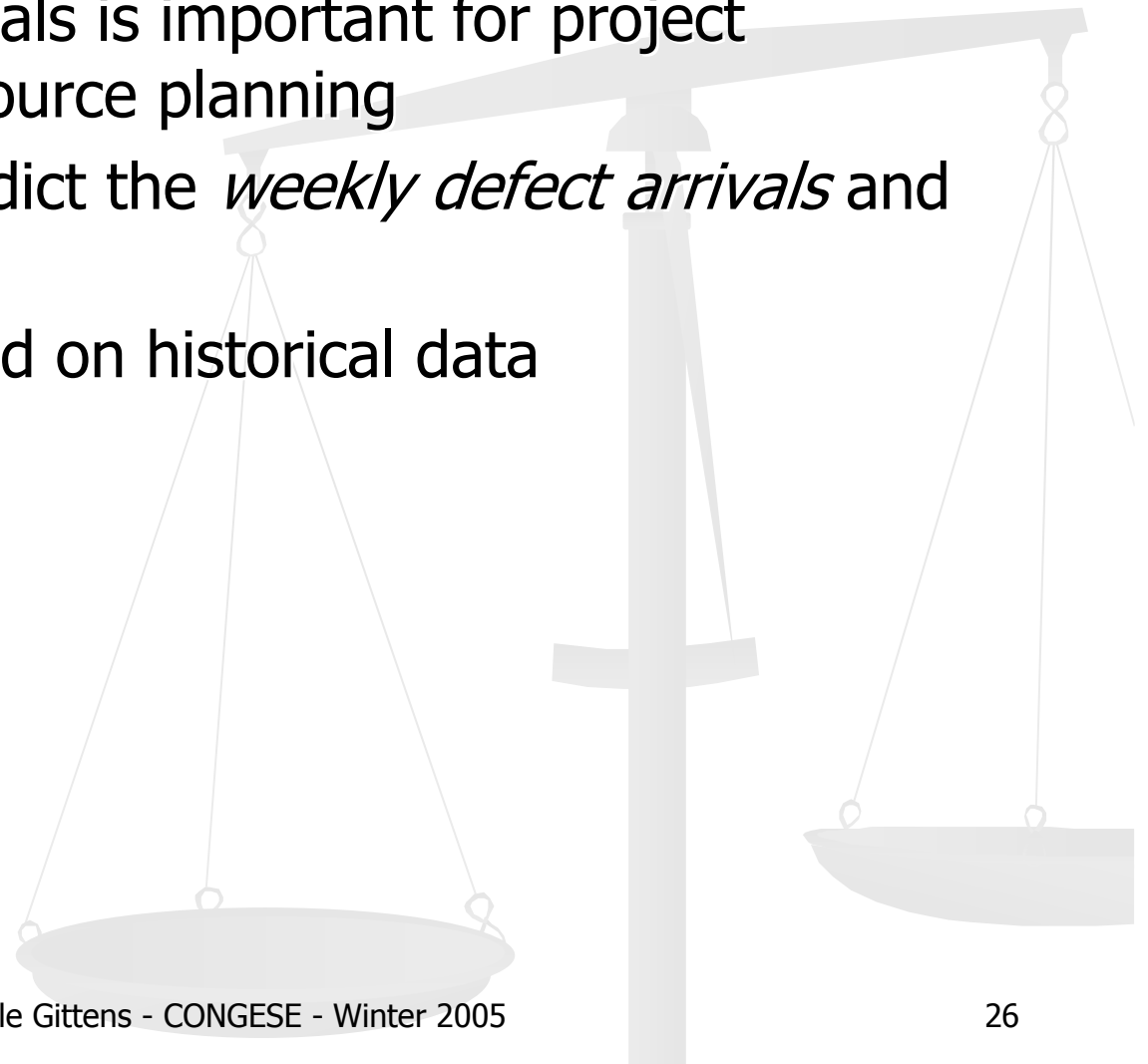


Problems with prediction

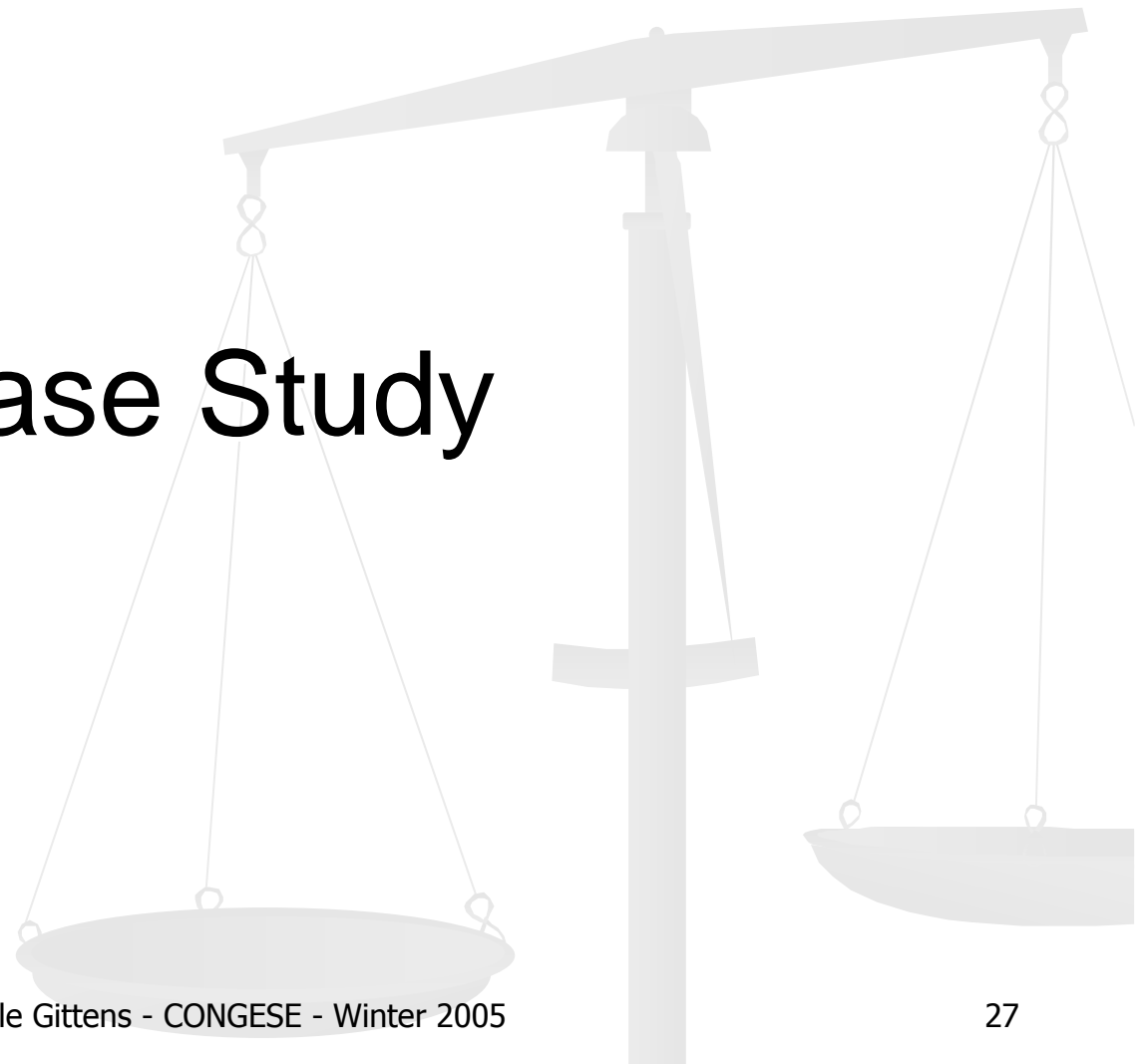
- Data collection – lack of automation, reliability, etc.
- Real-time concerns (want results for THIS release asap)
- Validity of predictions

Problems with prediction

- Predicting defect arrivals is important for project management and resource planning
- The purpose is to predict the *weekly defect arrivals* and *defects in backlog*
- The prediction is based on historical data



Case Study



Fitting a function

- **Fitting the data with a polynomial**

$$y = \beta_0 + \beta_1x + \beta_2x^2 + \dots + \beta_kx^k + \varepsilon, \text{ where}$$

- x is the number of weeks
 - y is the defect arrivals in each week
 - ε is the error term
-
- Sometimes defect information needed for the previous prediction model may not be available.
 - A polynomial is a favorite choice of many because it is easy to manipulate mathematically

Fitting a function

- **Fitting the data with a polynomial**

- The least square method can be used to minimize the error term ε . Popular software such as MS Excel can be used to generate the coefficients of the polynomial

- With $k = 4$, using linear regression method (as in MS EXCEL), the polynomial is obtained as

$$\begin{aligned}y &= \beta_0 + \beta_1x + \beta_2x^2 + \beta_3x^3 + \beta_4x^4 \\ &= 86.51 - 87.8x + 25.75x^2 - 2.03x^3 + 0.048x^4\end{aligned}$$

How good is the fitting?

■ Coefficient of Multiple Determination

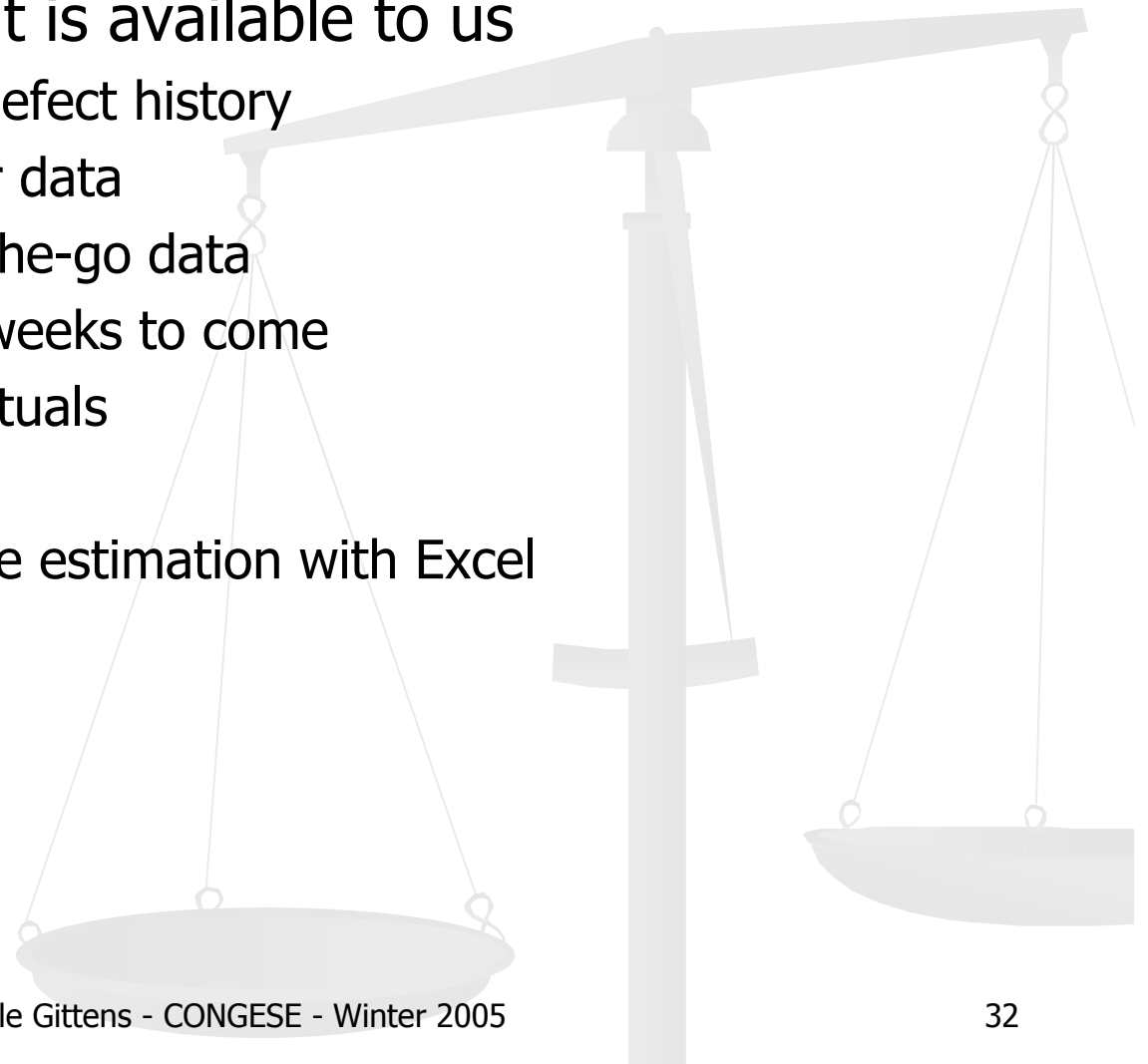
- Defined as $R^2 = SS_R/SS_T$
- In the example, e.g., $R^2 = SS_R/SS_T = 0.9615201$ indicating 96.2% of the variability of y (defect arrival) has been explained by the variables $\{x, x^2, x^3, x^4\}$
- Measures the strength of the fit

Let's do some analysis

- You can do this!
- What is the best predictor of defects?
- History suggests lines of code changed/added
- History suggests time and phases
- There are others
- You need to capture the history of your organization

Let's do some analysis

- We must look at what is available to us
 - Step 1: Gather your defect history
 - Step 2: Clean up your data
 - Step 3: Use your on-the-go data
 - Step 4: Estimate for weeks to come
 - Step 5: Verify with actuals
- Real example – Simple estimation with Excel



References

- Stephan Kan, *"Metrics and Models in Software Quality Engineering"*, 2nd Edition, Addison-Wesley
- M.Caruso and D.W.Desormeau, *"Integrating prior knowledge with a software reliability growth model"*, in Proceedings of the 13th International Conference on Software Engineering, 1991, pp. 238-245
- A. Goel, and K. Okumoto, *"Time-dependent error detection rate model for software reliability and other performance measures"*, IEEE Trans. Reliab., R-28, 1979, pp. 206–211
- *"Metrics and Models in Software Quality Engineering"*, 2nd ed., Addison-Wesley, Boston, 2003
- M. Ohba, *"Software Reliability Analysis Model"*, IBM Journal of Research and Development. Vol. 28, 1984, pp. 428-443
- N.D. Singpurwalla and S.P. Wilson *"Statistical Methods in Software Engineering: Reliability and Risk"* New York, Springer, 1999
- M. Xie and G.Y. Hong, Book Chapter: Software reliability modeling, estimation and analysis, *Handbook of Statistics 20: Advances in Reliability*, ed. N. Balakrishnan and C.R.Rao, Elsevier, pp. 707-731, 2001
- S. Yamada, M. Ohba and S. Osaki, *"S-shaped reliability growth modeling for Software Error Detection"*, IEEE Trans. Reliab., R-32, 1983, pp.475-478

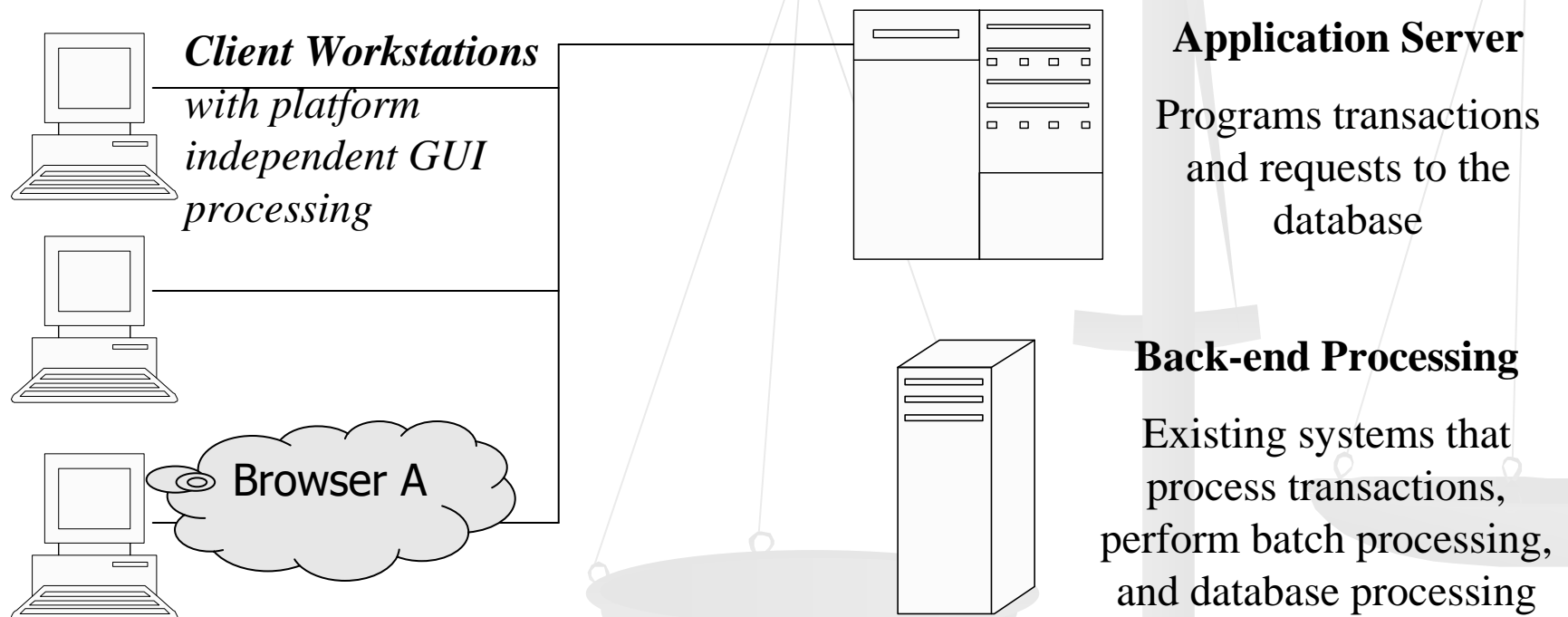


Testing Web-based Systems

adapted from Software Testing by William Perry

Testing Web-based Systems

- These are systems using intranets, the internet, extranets
- Web-based architectures are an extension of client-server architectures
- Client-server architecture



Testing Web-based Systems

- Browsers: on client workstations
- Client workstations: networked to web server (LAN, WAN, dial-up)
- Web server: accepts and processes requests from workstations
- Application server: performs data queries ecommerce transactions etc.
- Back-end processor: in the background to perform batch processing, high volume transactions, interface with systems not in the web app
 - For example web banking app with update internal bank database

Testing Web-based Systems

■ Concerns

- **Browser compatibility:** These tests validate consistent application performance on a variety of browser types and configurations
- **Functional correctness:** These tests validate that the application functions correctly – validating links, calculations, displays of information and navigation
- **Integration:** These tests validate the integration between browsers and servers, applications and data, hardware and software

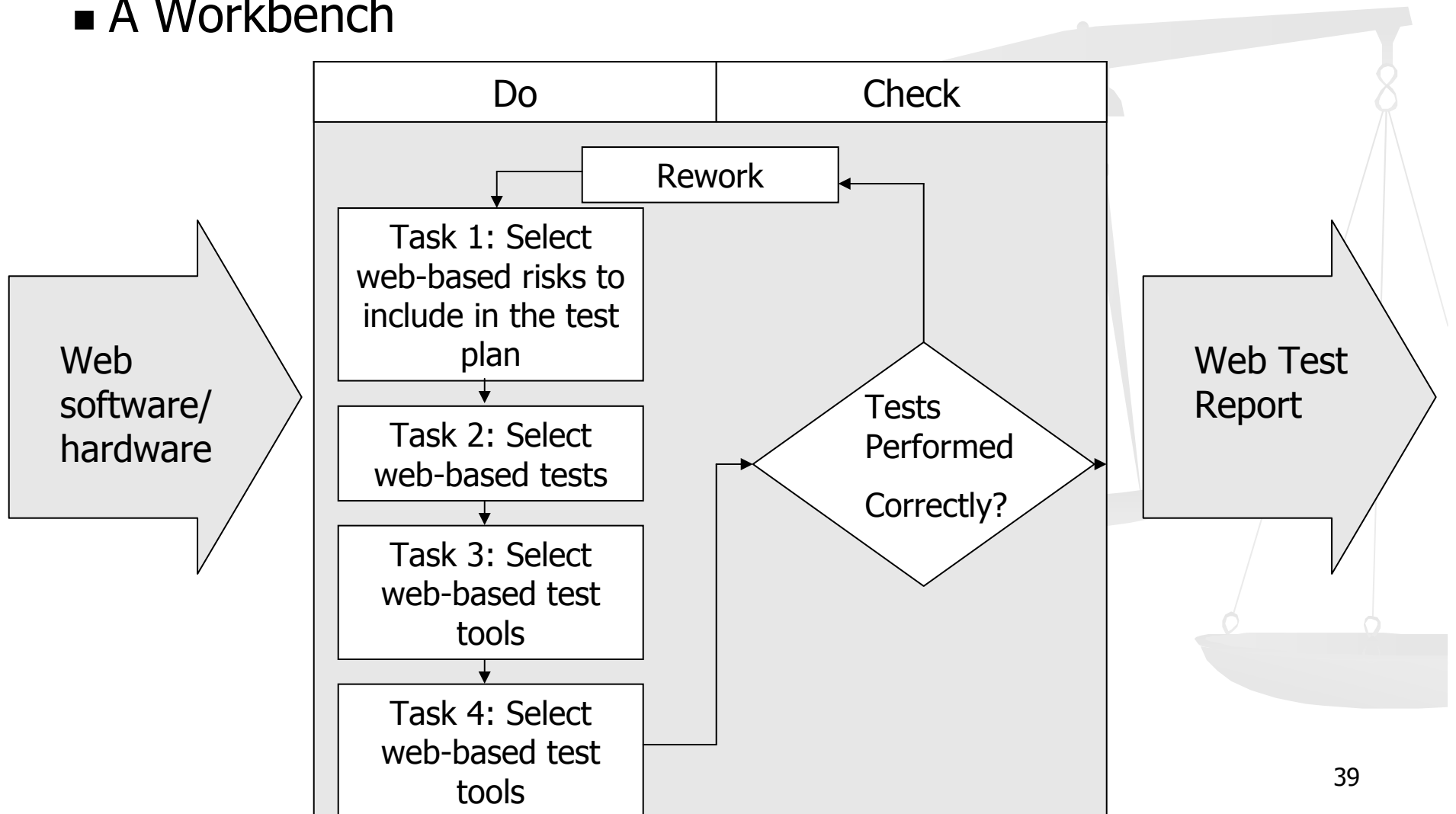
Testing Web-based Systems

■ Concerns

- Usability: Test the overall usability of the web page or the web application, including appearance, clarity, and navigation
- Security: Test the adequacy and correctness of security controls including access controls and authorizations
- Performance: Test the performance of the web application under load
- Verification of code: validate that the code used to create the application (HTML, Java etc.) has been used correctly
 - For example: make sure that there is no nonstandard approach that will malfunction in a different environment

Testing Web-based Systems

■ A Workbench



Testing Web-based Systems

- Input
 - Description of web-based technology for the systems used
 - Differences between web-based systems and other technology
 - Uncontrolled user interfaces
 - Must have knowledge of browsers that will access the system
 - Must keep up with new releases of browsers
 - Complex distributed systems
 - Security issues
 - Protection required from unauthorized access
 - Can corrupt data and applications
 - Possible wide access to confidential information

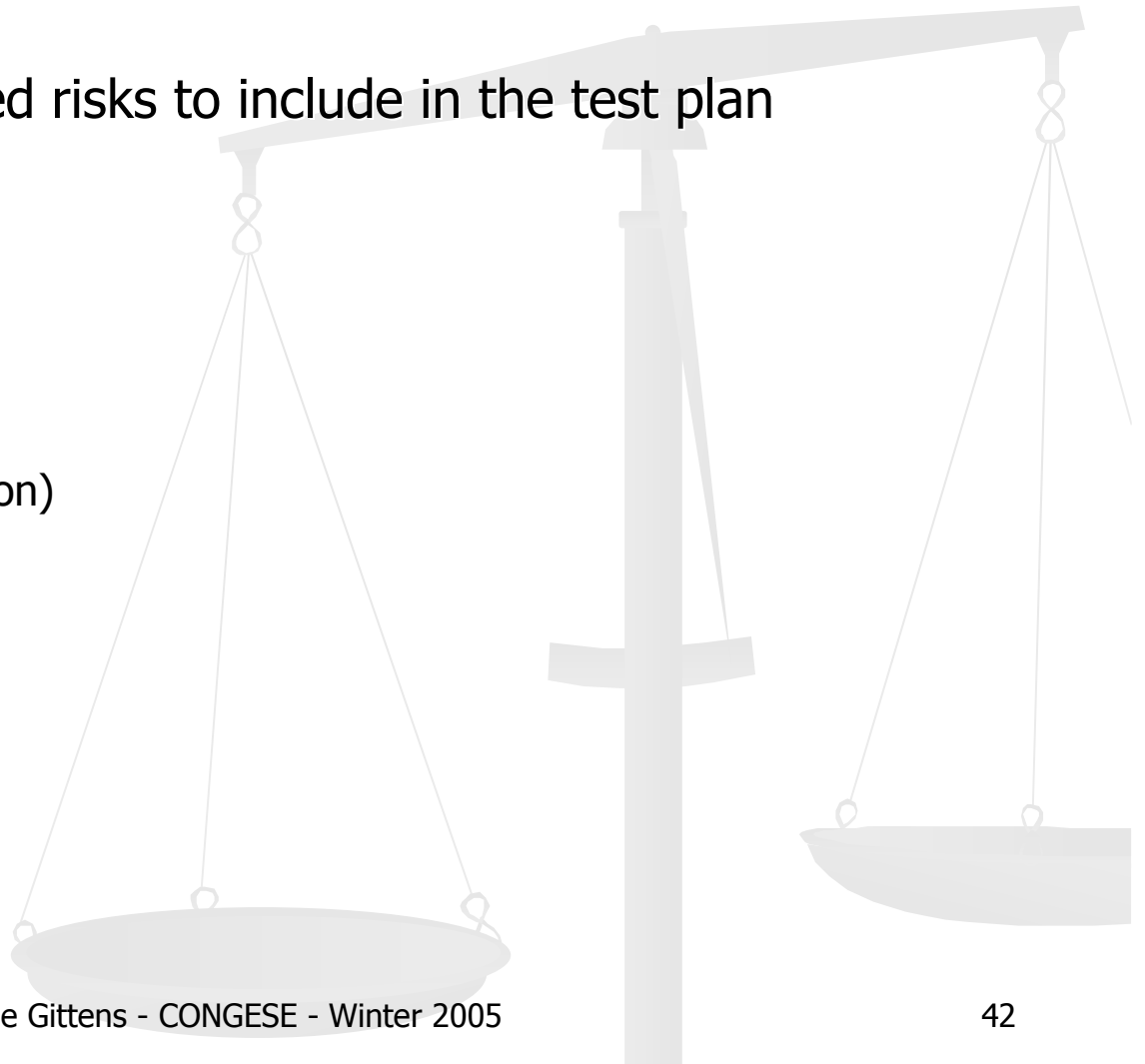
Testing Web-based Systems

- Input
 - Description of web-based technology for the systems used
 - Differences between web-based systems and other technology
 - Multiple layers in architecture
 - Application servers
 - Web servers
 - Back end processing
 - Data warehouses
 - Electronic commerce servers
 - Object-oriented
 - Competing standards
 - Some standards have emerged - Java

Testing Web-based Systems

■ Do Procedures

- Task 1: Select Web-based risks to include in the test plan
 - Risks reveal what to test
 - Risks
 - Security
 - Performance
 - Correctness
 - Compatibility (configuration)
 - Reliability
 - Data Integrity
 - Usability
 - Recoverability



Testing Web-based Systems

■ Do Procedures

■ Task 1: Test each of the following specifics of a given risk

■ Security

- External Intrusions
- Protection of secured transactions
- Viruses
- Access control
- Authorization levels

■ Performance

- Testing performance requires automation
- Load testing is very common – assesses performance under expected usage and greater-than-expected usage
 - Concurrency testing: Evaluate with given number of concurrent users
 - Stress testing: Evaluate performance of an application while stretching crucial aspects to the limit
 - Throughput: Evaluate the ability of the application to handle a given number of transactions in a a given time period – For example some an application may need to process 1000 transactions per hour.

Testing Web-based Systems

- Do Procedures
 - Task 1: Test each of the following specifics of a given risk
 - Correctness
 - Application must function correctly
 - Perform functional tests – test each function – ensure result is as expected
 - Functionality of buttons
 - Calculations – unit test, function test, system test
 - Correct navigation – test links, or navigation through web application
 - Authorization levels

Testing Web-based Systems

- Do Procedures
 - Task 1: Test each of the following specifics of a given risk
 - Compatibility
 - Operating systems and browsers
 - Know you users, test their variety
 - Use access statistics to identify these users
 - Browser configuration
 - Browser's configuration options affect information display
 - Hardware compatibility:
 1. Monitors, video cards and video RAM – offer "low-tech" option
 2. Audio, video, multimedia support – beware of requiring plug-ins – if necessary facilitate download from your site
 3. Memory (RAM) and hard drive space – consider and test with caching and - beware of high RAM requirements
 4. Bandwidth access – lots of people are using dial-up, remember this – be inclusive

Testing Web-based Systems

- Do Procedures

- Task 1: Test each of the following specifics of a given risk

- Compatibility

- Browser configuration (Cont'd)

- Consider that browsers will create differences in the following ways – ensure handling'

1. Print handling – you can add a “print-friendly version”
2. Reload – some browser configurations will not automatically show page updates – you can request “click refresh”
3. Navigation – browsers vary with ease of navigation – e.g. revisiting previously visited pages - offer navigational shortcuts
4. Graphics filters – browsers handle images differently – standardize on jpg and gif

Testing Web-based Systems

- Do Procedures

- Task 1: Test each of the following specifics of a given risk

- Compatibility

- Browser configuration (Cont'd)

- Consider that browsers will create differences in the following ways – ensure handling'
5. Dynamic page generation – some pages change based on input – shopping cart applications; data search applications; advertisements; calculation forms
 6. File downloads – how is movement of data for processing handled – may need to upload data before processing is done – instead of processing on user machine
 7. Email applications – if sending email is required, perhaps a form will facilitate – as opposed to calling up external applications – can have unexpected results

Testing Web-based Systems

- Do Procedures
 - Task 1: Test each of the following specifics of a given risk
 - Reliability
 - Need continuous up time – ensure server and system availability
 - Ensure that results are correct consistently
 - Data Integrity
 - Ensure that only correct data is accepted – validate the data at the page level when entered by the user
 - Ensure that the data stays correct – have procedures to back up data, ensure that they work, control data update

Testing Web-based Systems

- Do Procedures
 - Task 1: Test each of the following specifics of a given risk
 - Usability
 - Ensure that the application is easy to use and understand
 - Ensure that the users can interpret the data generated by the app
 - Ensure clear and correct navigation
 - Recoverability
 - Ensure recovery from
 - Lost connections – e.g. that data integrity is maintained
 1. Timeouts
 2. Dropped lines – does your session time out too quickly?
 - Client system crashes
 - Server crashes, or the application has other problems

Testing Web-based Systems

- Do Procedures
 - Task 2: Select web-based tests
 - We understand the risk so now what do we need to do to mitigate them?
 - Unit/Component Tests
 - Verify the risks at each of the following levels
 1. Object
 2. Component
 3. Page
 4. Applets
 - Integration
 - Test the passing of data and/or control between units or components
 - Testing navigation – paths the data will follow
 - Test links, data exchanges, flow of control in the application

Testing Web-based Systems

- Do Procedures
 - Task 2: Select web-based tests
 - We understand the risk so now what do we need to do to mitigate them?
 - System Tests
 - Internet web pages
 - Data warehouses
 - Back-end processing
 - Reporting systems
 - User acceptance
 - Business process validation
 - Conduct business correctly, efficiently
 - Internally – management, end-user groups, independent test teams
 - Externally – beta testers, independent test organizations

Testing Web-based Systems

- Do Procedures
 - Task 3: Select web-based test tools
 - Since tooling is critical
 - Get senior management buy-in for purchase and integration of tools
 - Know your requirements – helps to choose best fit tools
 - Have reasonable expectations – start small and grow
 - Have a strong testing process that provides for the incorporation of tools
 - Train your team, or you cannot blame the tool

Testing Web-based Systems

- Do Procedures
 - Task 3: Select web-based test tools
 - Useful list at:
<http://www.w3.org/WAI/ER/existingtools.html#Evaluation>
- HTML test tools
 - Many web development packages include an HTML checker
 - If not you can use a standalone tool like Doctor HTML (Imagineware) <http://www2.imagiware.com/RxHTML/>
 - Try it!

Testing Web-based Systems

Doctor HTML is a Web page analysis tool for use over the World Wide Web or across company intra-nets behind a firewall. The main program is written in Perl for easy adaptation to specific tasks. To purchase a license of **Doctor HTML** now go to our [Secure Order Form](#).

Features

The following features are available in the current release (v6.1) of **Doctor HTML**.

- Easy-to-use Web Interface for report configuration
- Informative, nicely-formatted report with hyperlinks to additional information.
- Fifteen report options:
 - Check the document for spelling errors (provides suggestions for potentially misspelled words)
 - Perform an analysis of the images (size, number of colors, etc.)
 - Test the document structure and flag invalid HTML
 - Examine image syntax (flag missing, but recommended elements)
 - Test browser support of tags used
 - Examine fonts in page to check for cross-platform support
 - Examine table structure
 - Verify that all hyperlinks are valid
 - Examine form structure
 - Show command hierarchy
 - Produce a "squished" version of the page
 - Produce a "formatted" version of the page
 - Expand frameset for further testing
 - Check Meta tags for search engine relevance
 - Show cookies set by the page



Testing Web-based Systems

- Do Procedures
 - Task 3: Select web-based test tools
 - Site Validation
 - These tools check the web application for inconsistencies and errors such as:
 - Moved pages
 - Orphaned pages
 - Broken links
 - SQA Site Check - Rational
 - <http://www.wilsonmar.com/robot7/sitecheck.pdf>

Testing Web-based Systems

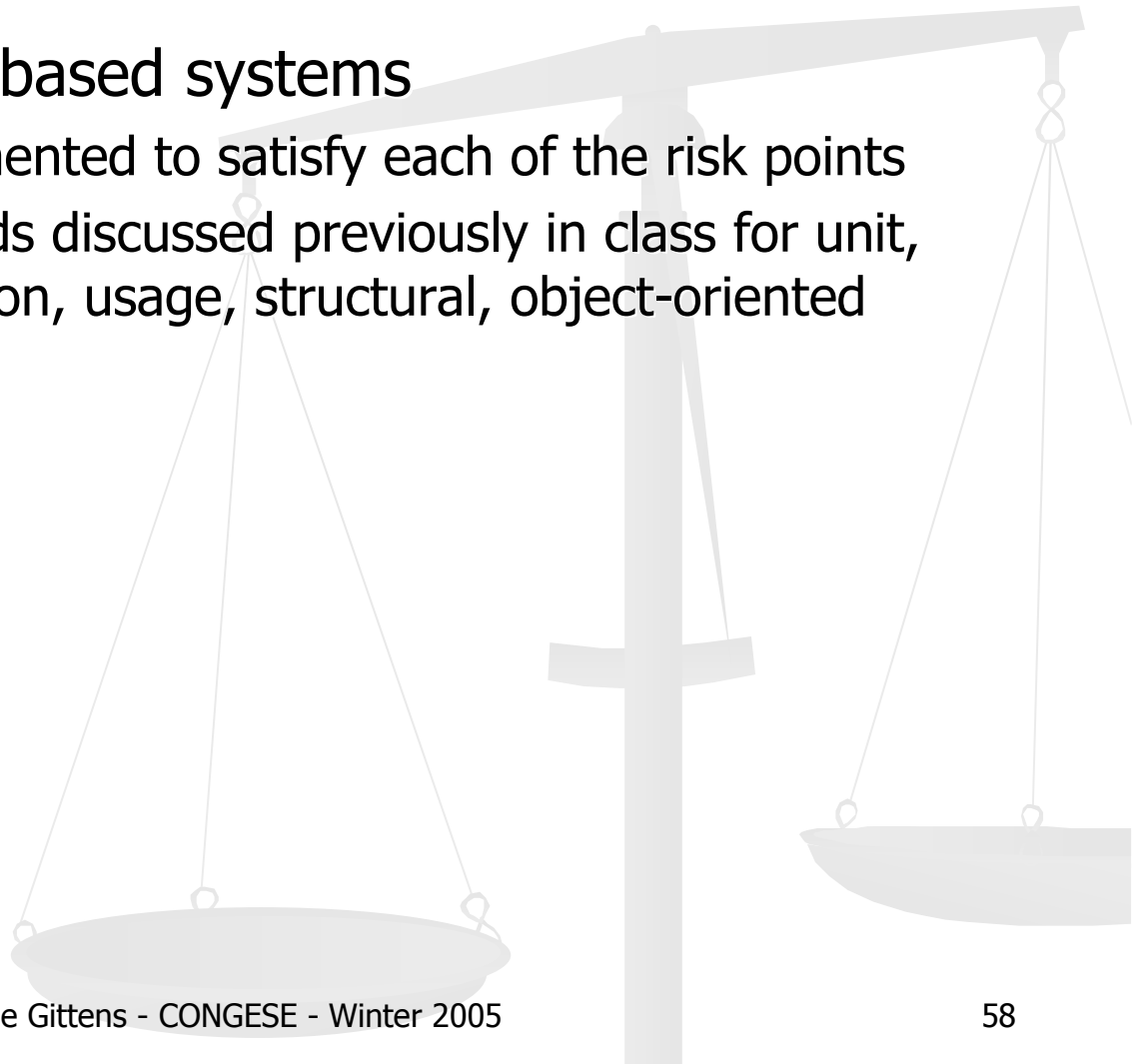
- Do Procedures
 - Task 3: Select web-based test tools
 - Java Test tools
 - Specific to Java applications
 - Sun Test Suite – JavaStar (testing GUIs), JavaSpec (testing APIs), JavaScope (measure code coverage)
 - JUnit – Unit tests for java code – helps regression
 - SilkTest by Segue – capture, playback for web applications
 - SilkScope by Segue – Code coverage evaluation
 - SilkSpec by Segue – tests the non-GUI code of applications and applets

Testing Web-based Systems

- Do Procedures
 - Task 3: Select web-based test tools
 - Load/Stress testing tools
 - Evaluate web systems under high data or transactions
 - The tools simulate several virtual users with varying transaction rates
 - Silk Performer – Segue <http://www.segue.com/products/load-stress-performance-testing/silkperformer.asp>
 - Rational Robot <http://www-306.ibm.com/software/awdtools/tester/robot/>

Testing Web-based Systems

- Do Procedures
 - Task 4: Test web-based systems
 - Tests are implemented to satisfy each of the risk points
 - Based on methods discussed previously in class for unit, system, integration, usage, structural, object-oriented methods etc.



Testing Web-based Systems

- Check Procedures
 - Must verify that testing was effective
- Output
 - There must be a report of the web-based system
 - Brief description of the system
 - Risks addressed and not addressed by the test team
 - Types of tests performed and not performed
 - Test tools used
 - Web-based functionality that performed correctly
 - Web-based functionality that did not perform correctly
 - The web test team's evaluation of the readiness of the system for production

Next Class

- 9:30 am
 - Project Presentations
- 12 noon
 - Project write-up due
 - Peer evaluation sheets will be distributed April 1
 - Peer evaluations due with write-ups
- 1pm
 - **Dr. Matunda Nyanchama – IBM National Security Leader**
 - ***Software Development Practices & Security***

Next Class

Software Development Practices & Security

Abstract

Security breaches make headlines on an ongoing basis while companies lose valuable time and incur losses in responding to security incidents following the exploitation of software flaws. According to the 2005 CSI/FBI Computer Crime and Security Survey, viruses, worms and other malware continue to cause substantial losses in industry. Some think that the reported losses constitute a tip of the iceberg of cumulative loss. Risks associated with flaws in software could be much larger. Security flaws have origins in how security software is developed. The talk will focus on the need for a disciplined approach to software development and the adoption of engineering practices as a basis for ensuring secure software.