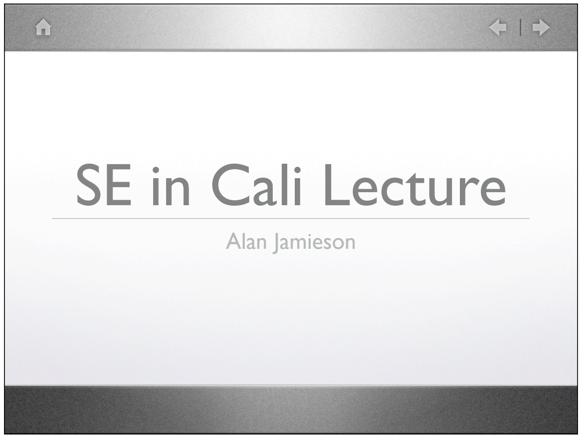
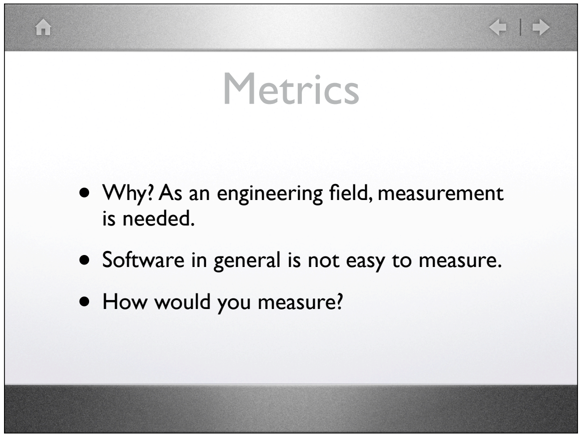


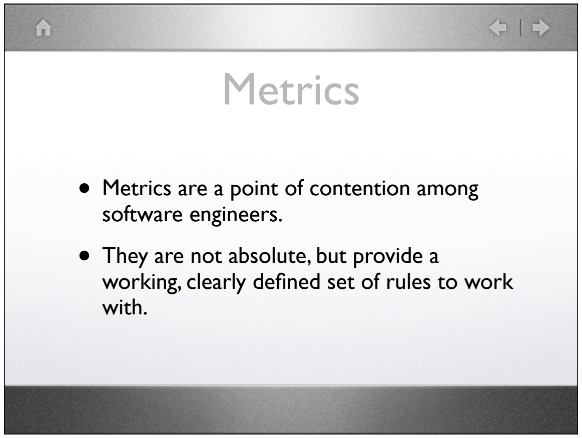
SE in Cali Lecture

Alan Jamieson



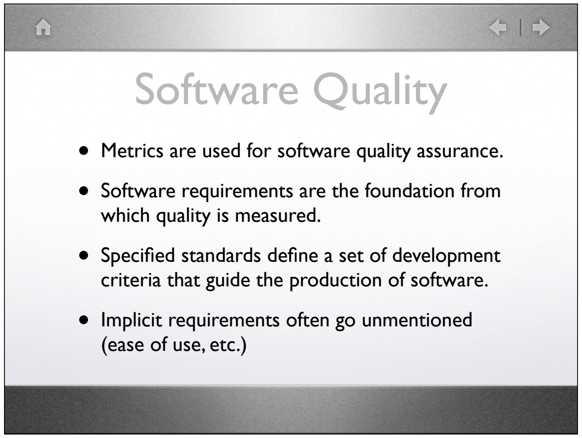
Metrics

- Why? As an engineering field, measurement is needed.
- Software in general is not easy to measure.
- How would you measure?



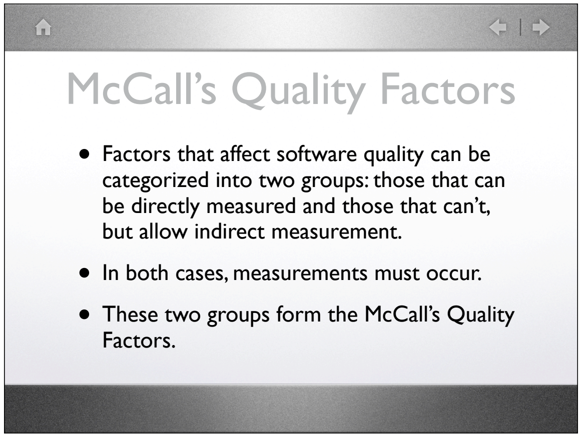
Metrics

- Metrics are a point of contention among software engineers.
- They are not absolute, but provide a working, clearly defined set of rules to work with.



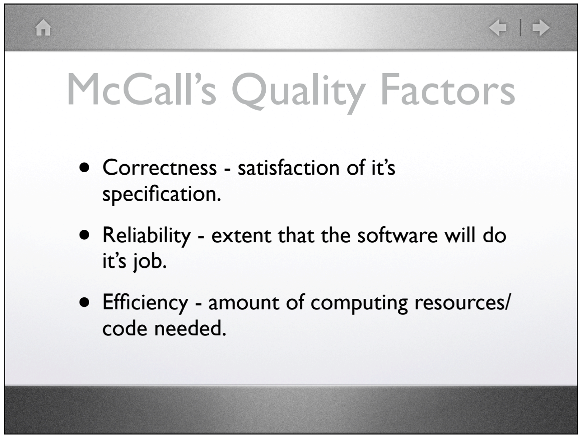
Software Quality

- Metrics are used for software quality assurance.
- Software requirements are the foundation from which quality is measured.
- Specified standards define a set of development criteria that guide the production of software.
- Implicit requirements often go unmentioned (ease of use, etc.)



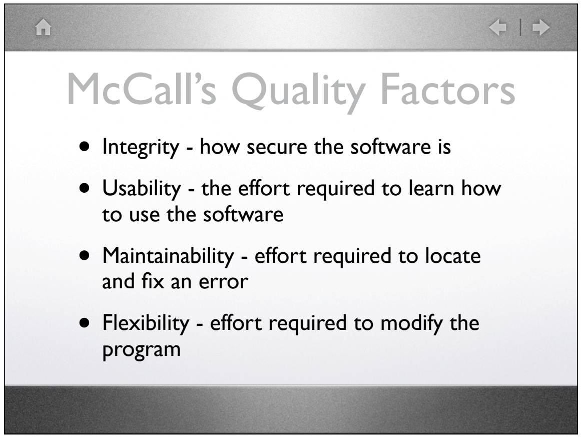
McCall's Quality Factors

- Factors that affect software quality can be categorized into two groups: those that can be directly measured and those that can't, but allow indirect measurement.
- In both cases, measurements must occur.
- These two groups form the McCall's Quality Factors.



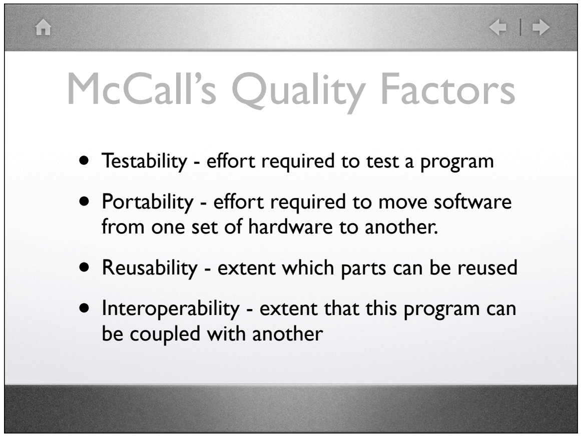
McCall's Quality Factors

- Correctness - satisfaction of it's specification.
- Reliability - extent that the software will do it's job.
- Efficiency - amount of computing resources/ code needed.



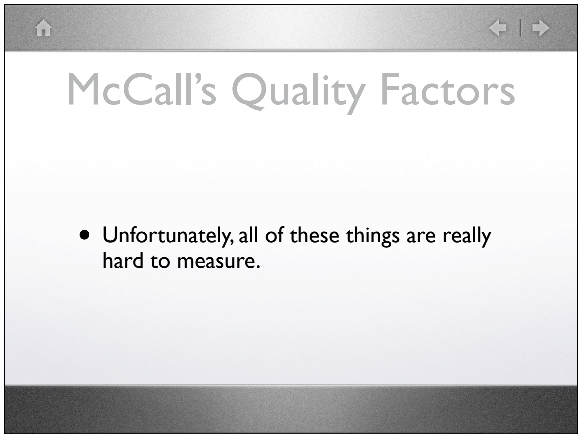
McCall’s Quality Factors

- Integrity - how secure the software is
- Usability - the effort required to learn how to use the software
- Maintainability - effort required to locate and fix an error
- Flexibility - effort required to modify the program



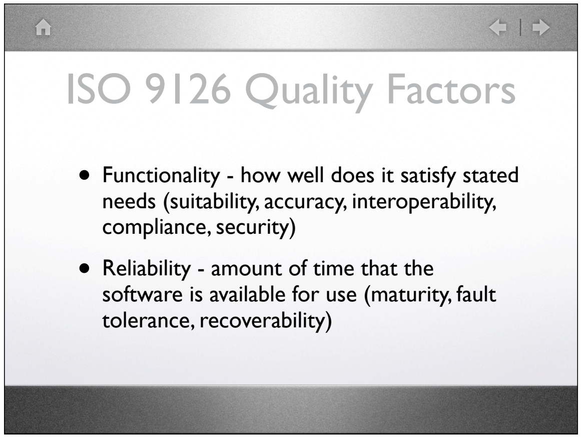
McCall’s Quality Factors

- Testability - effort required to test a program
- Portability - effort required to move software from one set of hardware to another.
- Reusability - extent which parts can be reused
- Interoperability - extent that this program can be coupled with another



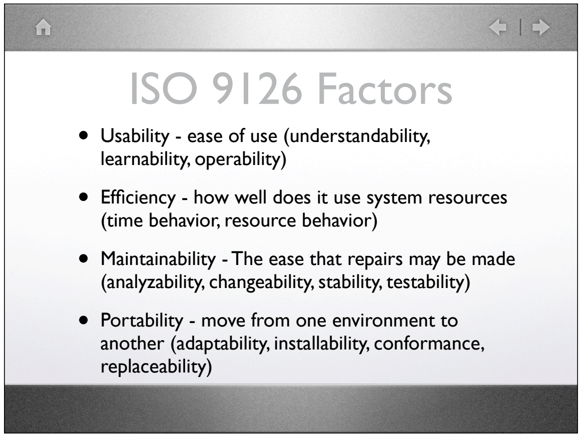
McCall’s Quality Factors

- Unfortunately, all of these things are really hard to measure.



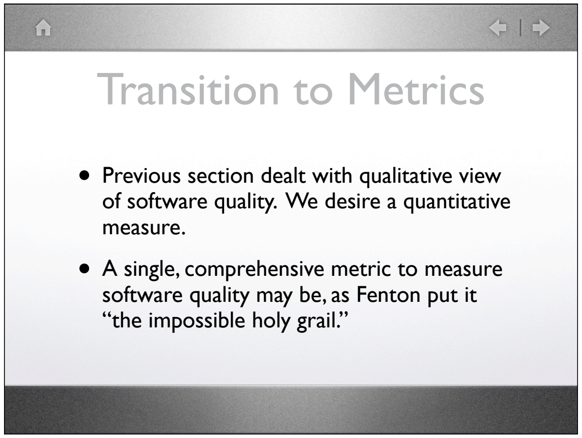
ISO 9126 Quality Factors

- Functionality - how well does it satisfy stated needs (suitability, accuracy, interoperability, compliance, security)
- Reliability - amount of time that the software is available for use (maturity, fault tolerance, recoverability)



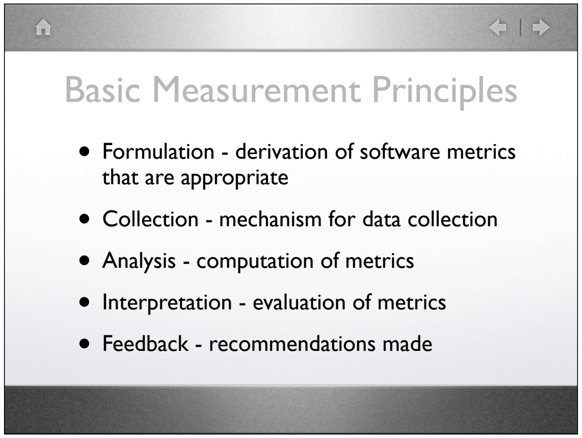
ISO 9126 Factors

- Usability - ease of use (understandability, learnability, operability)
- Efficiency - how well does it use system resources (time behavior, resource behavior)
- Maintainability - The ease that repairs may be made (analyzability, changeability, stability, testability)
- Portability - move from one environment to another (adaptability, installability, conformance, replaceability)



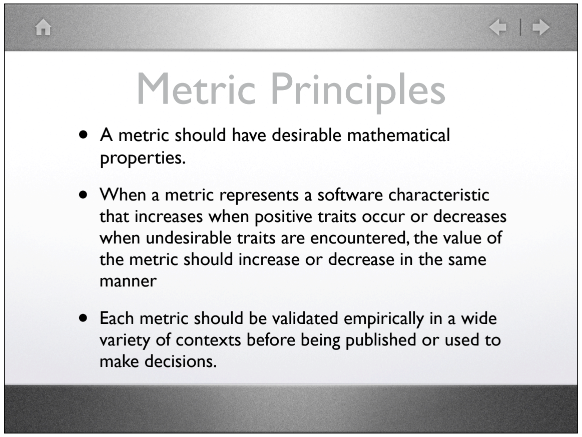
Transition to Metrics

- Previous section dealt with qualitative view of software quality. We desire a quantitative measure.
- A single, comprehensive metric to measure software quality may be, as Fenton put it "the impossible holy grail."



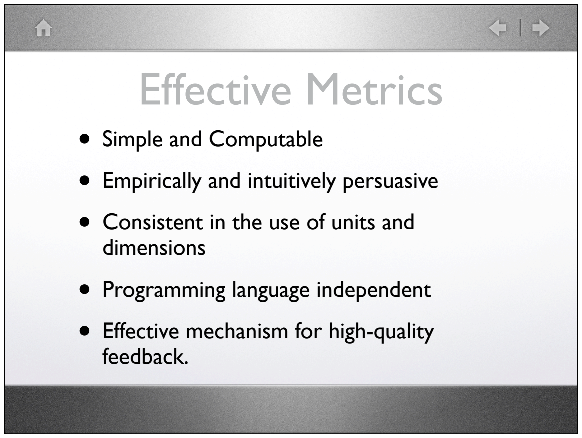
Basic Measurement Principles

- Formulation - derivation of software metrics that are appropriate
- Collection - mechanism for data collection
- Analysis - computation of metrics
- Interpretation - evaluation of metrics
- Feedback - recommendations made



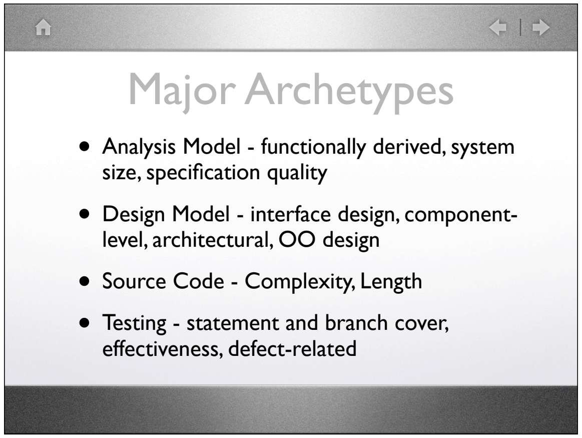
Metric Principles

- A metric should have desirable mathematical properties.
- When a metric represents a software characteristic that increases when positive traits occur or decreases when undesirable traits are encountered, the value of the metric should increase or decrease in the same manner
- Each metric should be validated empirically in a wide variety of contexts before being published or used to make decisions.



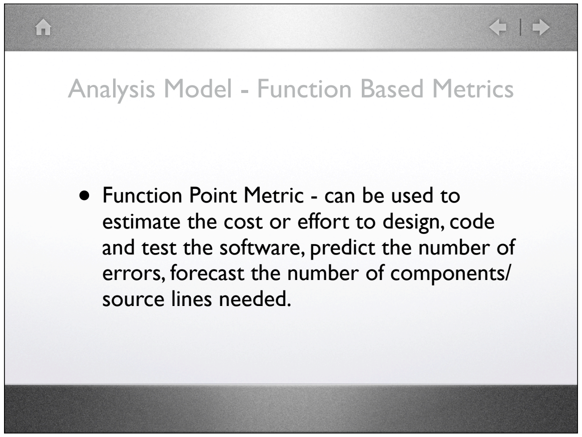
Effective Metrics

- Simple and Computable
- Empirically and intuitively persuasive
- Consistent in the use of units and dimensions
- Programming language independent
- Effective mechanism for high-quality feedback.



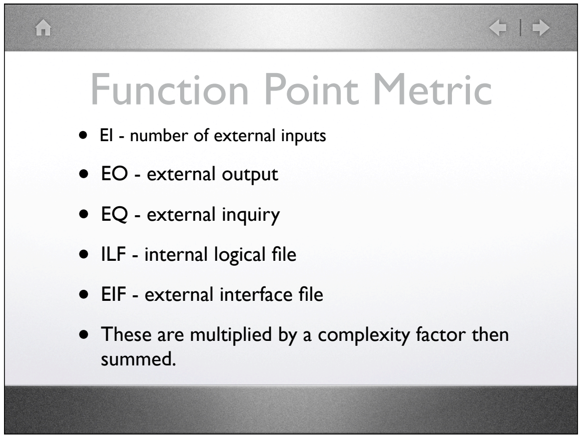
Major Archetypes

- Analysis Model - functionally derived, system size, specification quality
- Design Model - interface design, component-level, architectural, OO design
- Source Code - Complexity, Length
- Testing - statement and branch cover, effectiveness, defect-related



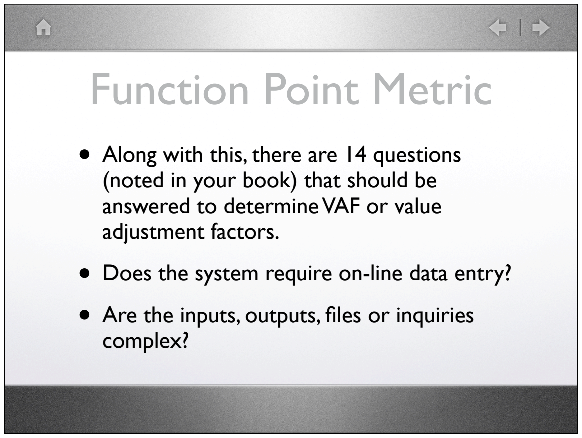
Analysis Model - Function Based Metrics

- Function Point Metric - can be used to estimate the cost or effort to design, code and test the software, predict the number of errors, forecast the number of components/ source lines needed.



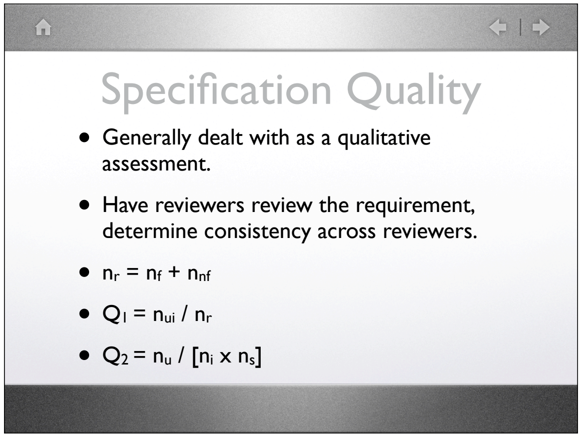
Function Point Metric

- EI - number of external inputs
- EO - external output
- EQ - external inquiry
- ILF - internal logical file
- EIF - external interface file
- These are multiplied by a complexity factor then summed.



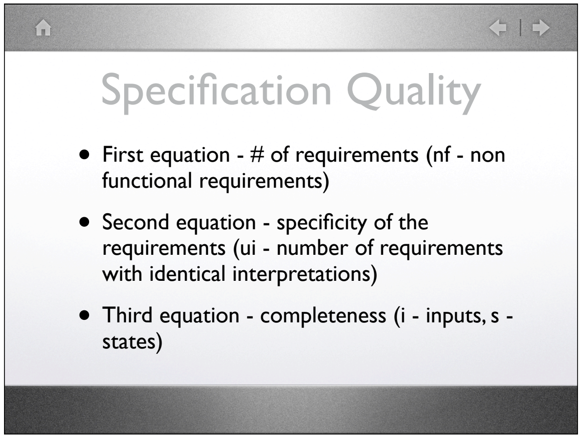
Function Point Metric

- Along with this, there are 14 questions (noted in your book) that should be answered to determine VAF or value adjustment factors.
- Does the system require on-line data entry?
- Are the inputs, outputs, files or inquiries complex?



Specification Quality

- Generally dealt with as a qualitative assessment.
- Have reviewers review the requirement, determine consistency across reviewers.
- $n_r = n_f + n_{nf}$
- $Q_1 = n_{ui} / n_r$
- $Q_2 = n_u / [n_i \times n_s]$



Specification Quality

- First equation - # of requirements (nf - non functional requirements)
- Second equation - specificity of the requirements (ui - number of requirements with identical interpretations)
- Third equation - completeness (i - inputs, s - states)

Architectural Design

- Card and Glass define three software design complexity measures:
  - structural complexity
  - data complexity
  - system complexity

Structural Complexity

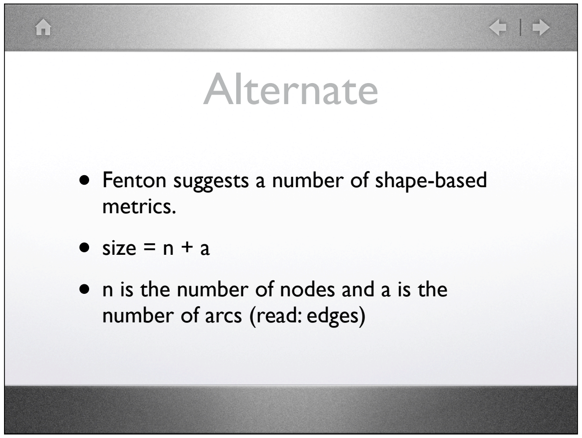
- for a module i -

  $S(i) = f^2_{out}(i)$

- f_{out} is the fan-out, or the number of modules directly subordinate

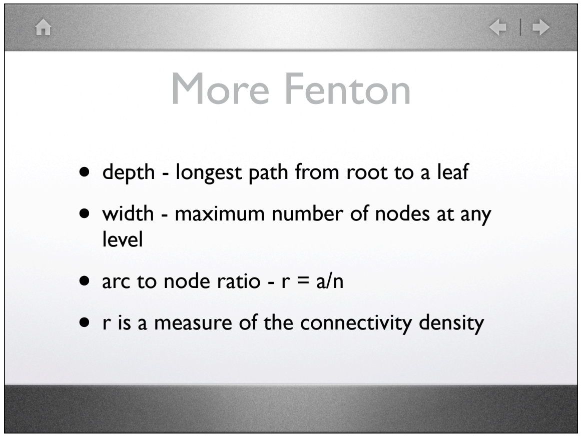
Data Complexity

- $D(i) = v(i)/[f_{out}(i) + 1]$
- v(i) is the number of input and output variables passed to and from i
- System complexity is the sum of D(i) and S(i)



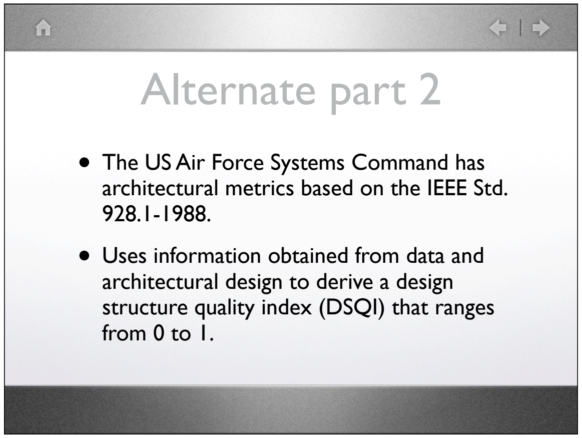
Alternate

- Fenton suggests a number of shape-based metrics.
- size = n + a
- n is the number of nodes and a is the number of arcs (read: edges)



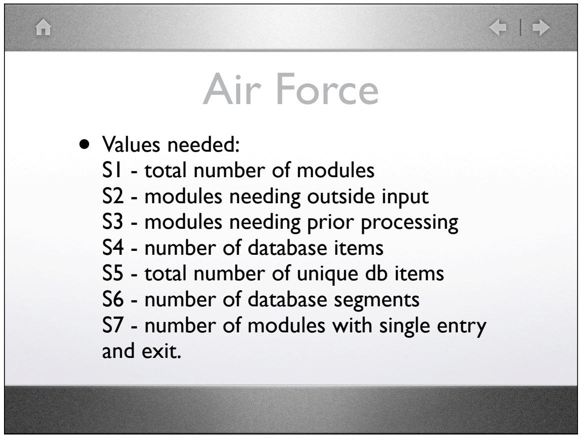
More Fenton

- depth - longest path from root to a leaf
- width - maximum number of nodes at any level
- arc to node ratio - r = a/n
- r is a measure of the connectivity density



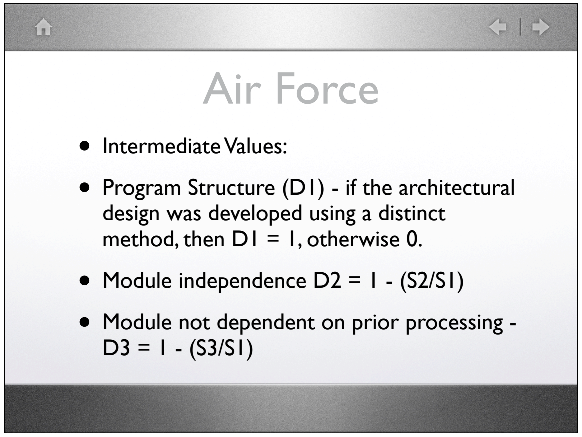
Alternate part 2

- The US Air Force Systems Command has architectural metrics based on the IEEE Std. 928.1-1988.
- Uses information obtained from data and architectural design to derive a design structure quality index (DSQI) that ranges from 0 to 1.



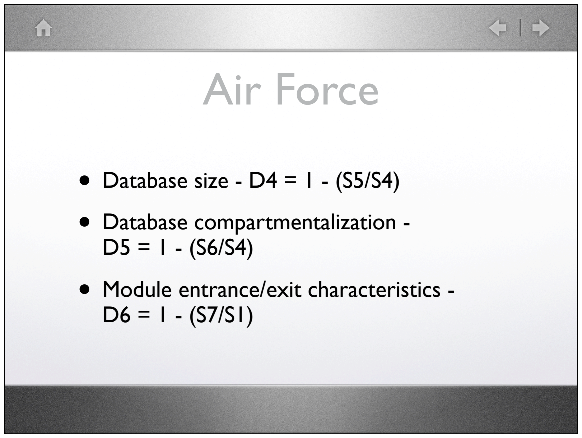
Air Force

- Values needed:
  S1 - total number of modules
  S2 - modules needing outside input
  S3 - modules needing prior processing
  S4 - number of database items
  S5 - total number of unique db items
  S6 - number of database segments
  S7 - number of modules with single entry and exit.



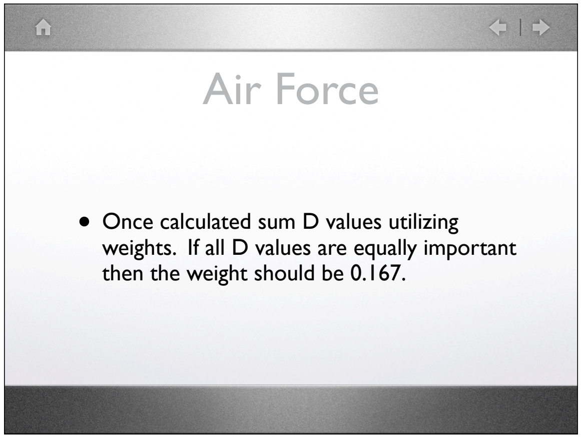
Air Force

- Intermediate Values:
- Program Structure (D1) - if the architectural design was developed using a distinct method, then D1 = 1, otherwise 0.
- Module independence D2 = 1 - (S2/S1)
- Module not dependent on prior processing - D3 = 1 - (S3/S1)



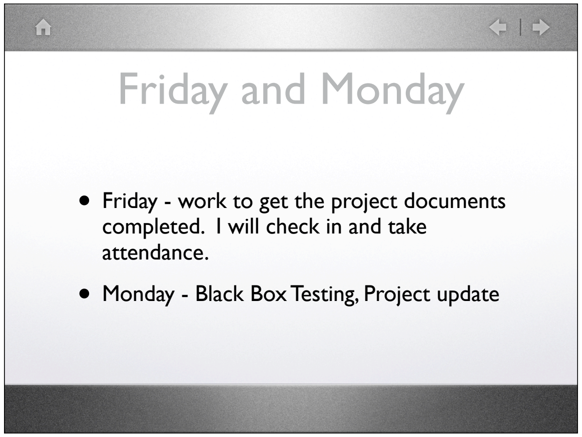
Air Force

- Database size - D4 = 1 - (S5/S4)
- Database compartmentalization - D5 = 1 - (S6/S4)
- Module entrance/exit characteristics - D6 = 1 - (S7/S1)



Air Force

- Once calculated sum D values utilizing weights. If all D values are equally important then the weight should be 0.167.



Friday and Monday

- Friday - work to get the project documents completed. I will check in and take attendance.
- Monday - Black Box Testing, Project update

SEE YA