# Metrics based field problem prediction

Paul Luo Li

ISRI – SE - CMU


Windows
An exception  06 has occured at 0028:C11B3ADC in VxD DiskTSD(03) +
00001660.  This was called from 0028:C11B40C8 in VxD voltrack(04) +
00000000.  It may be possible to continue normally.
*  Press any key to attempt to continue.
*  Press CTRL+ALT+RESET to restart your computer.  You will
lose any unsaved information in all applications.
Press any key to continue

# Field problems “happen”

Program testing can be used
to show the presence of bugs,
but never to show their absence!
*-Dijkstra*

Statement coverage, branch coverage, all definitions coverage, all p-uses coverage, all definition-uses coverage finds only 50% of a sample of field problems in TeX
- *Foreman and Zweben 1993*

Better, cheaper, faster… pick two
-Anonymous

# Take away

- Field problem predictions can help lower the costs of field problems for software producers and software consumers
- Metrics based models are better suited to model field defect when information about the deployment environment is scarce
- The four categories of predictors are product, development, deployment and usage, and software and hardware configurations
- Depending on the objective, different predictions are made and different predictions methods are used

# Benefits of field problem predictions

- Guide testing (Khoshgoftaar et. al. 1996)
- Improve maintenance resource allocation (Mockus et. al. 2005)
- Guide process improvement (Bassin and Santhanam 1997)
- Adjust deployment (Mockus et. al. 2005)
- Enable software insurance (Li et. al. 2004)

# Lesson objectives

- ☑ Why predict field defects?
- When to use time based models?
- When to use metrics based models?
- What are the component of metrics based models?
  - What predictors to use?
  - What can I predict?
  - How do I predict?

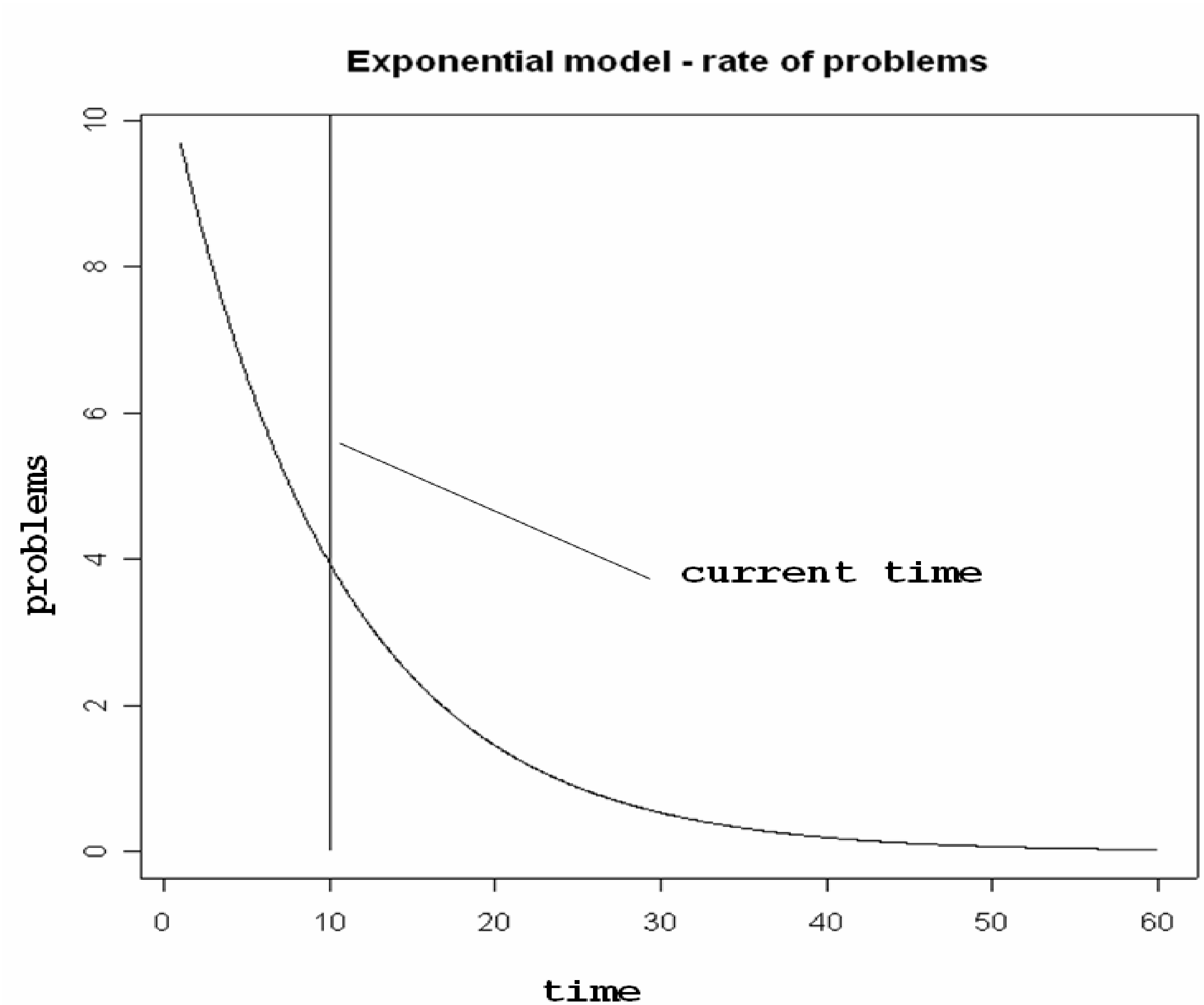
# Methods to predict field problems

- Time based models
  - Predictions based on the **time** when problems occur
- Metrics based models
  - Predictions based on **metrics** collected before release and field problems

# The idea behind time based models

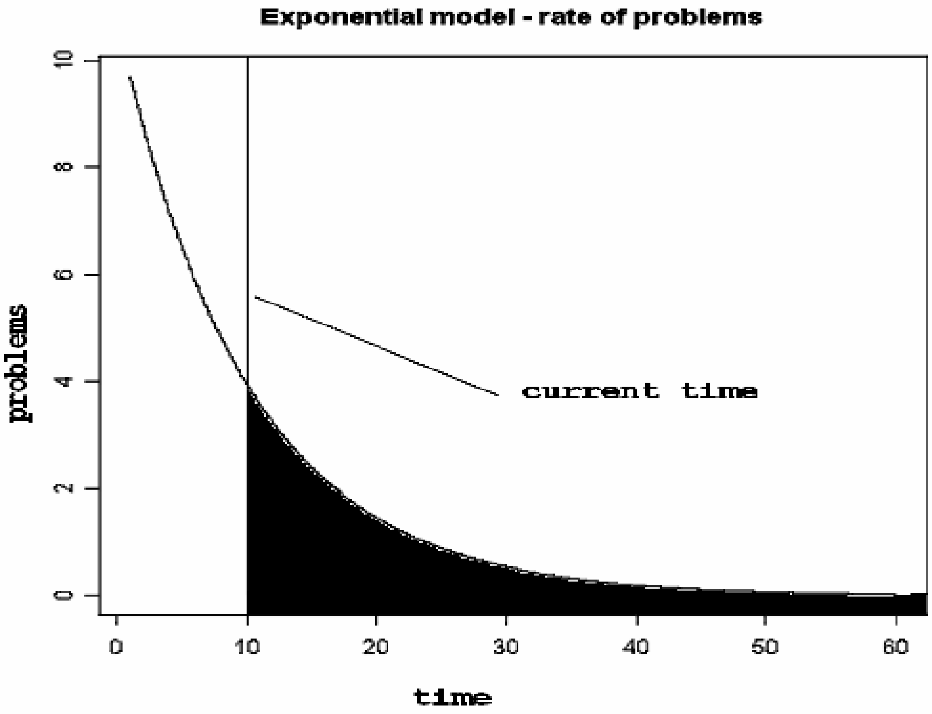
- The software system has a chance of encountering problems remaining during every execution
  - More problems there are in the code, higher the probability a problem will be encountered
- Assuming that a problem is discovered and is removed, the probability of encountering a problem during the next execution decreases.
- The more executions, higher the number of problems found

# Example



Exponential model - rate of problems
problems
0
2
4
6
8
10
0
10
20
30
40
50
60
time
current time

# Example




- $\lambda(t) = 107.01 * 10 * e^{-10 * t}$
- Integrate the function from t=10 to infinity, to get ~43 problems

# Key limitation

- In order for the defect occurrence pattern to continue into future time intervals, testing environment ~ operating environment
  - Operational profile
  - Hardware and software configurations in use
  - Deployment and usage information

# Situations when time based models have been used

- Controlled environment
  - McDonell Douglas (defense contractors building airplanes) studied by Jelinski and Moranda
  - NASA projects studied by Schneidewind

# Situations when time based models may not appropriate

- Operating environment is not known or infeasible to test completely
  - COTS systems
  - Open source software systems

# Lesson objectives

- ☑ Why predict field defects?
- ☑ When to use time based models?
- When to use metrics based models?
- What are the component of metrics based models?
  - What predictors to use?
  - What can I predict?
  - How do I predict?

# The idea behind metrics based models

- Certain characteristics make the presences of field defects more or less likely
  - Product, development, deployment and usage, software and hardware configurations in use
- Capture the relationship between predictors and field problems using past observations to predict field problems for future observations

# Difference between time based models and metrics based models

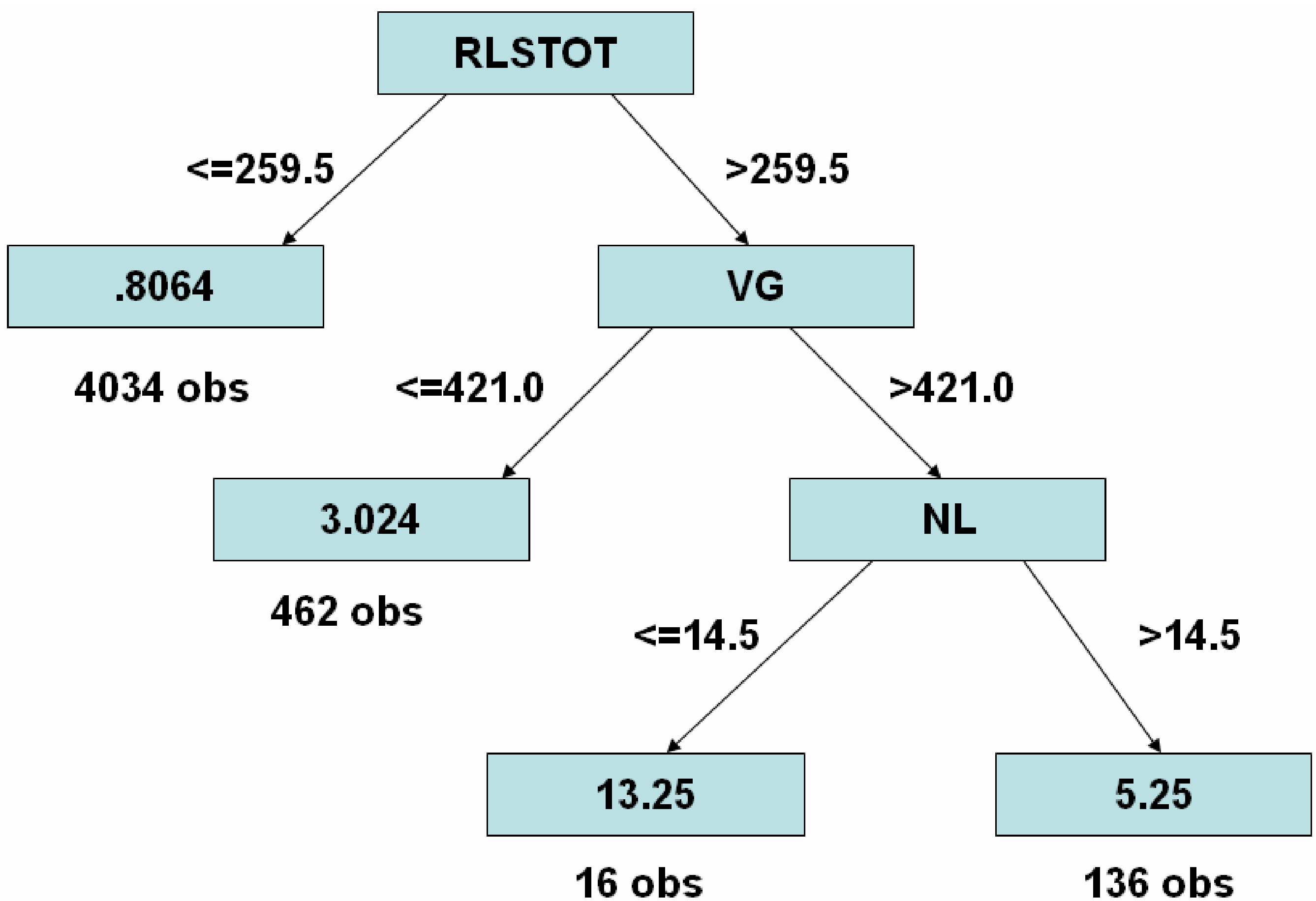
- Explicitly account for characteristics that can vary
- Model constructed using historical information on predictors and field defects

# Difference between time based models and metrics based models

- Explicitly account for characteristics that can vary
- Model constructed using historical information on predictors and field defects

Upshot: more robust against differences between development and deployment

# An example model




RLSTOT: vertices plus arcs within loops in flow graph
NL: loops in a flow graph
VG: Cyclomatic complexity

Khoshgoftaar et. al 1993

# Lesson objectives

- ☑ Why predict field defects?
- ☑ When to use time based models?
- ☑ When to use metrics based models?
- What are the component of metrics based models?
  - What predictors to use?
  - What can I predict?
  - How do I predict?

# Definition of metrics and predictors

- Metrics are outputs of measurements, where measurement is defined as the process by which values are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.
  – *Fenton and Pfleeger*
- Predictors are metrics available before release

# Categories of predictors

- Product metrics
- Development metrics
- Deployment and usage metrics
- Software and hardware configurations metrics

# Categories of predictors

- Product metrics
- Development metrics
- Deployment and usage metrics
- Software and hardware configurations metrics

Help us to think about the different kinds of attributes that are related to field defects

# The idea behind product metrics

- Metrics that measure the attributes of any intermediate or final product of the development process
  - Examined by most studies
  - Computed using snapshots of the code
  - Automated tools available

# Sub-categories of product metrics

- Control: Metrics measuring attributes of the flow of the program control
  - Cyclomatic complexity
  - Nodes in control flow graph

# Sub-categories of product metrics

- Control
- Volume: Metrics measuring attributes related to the number of distinct operations and statements (operands)
  - Halstead's program volume
  - Unique operands

# Sub-categories of product metrics

- Control
- Volume
- Action: Metrics measuring attributes related to the total number of operations (line count) or operators
  - Source code lines
  - Total operators

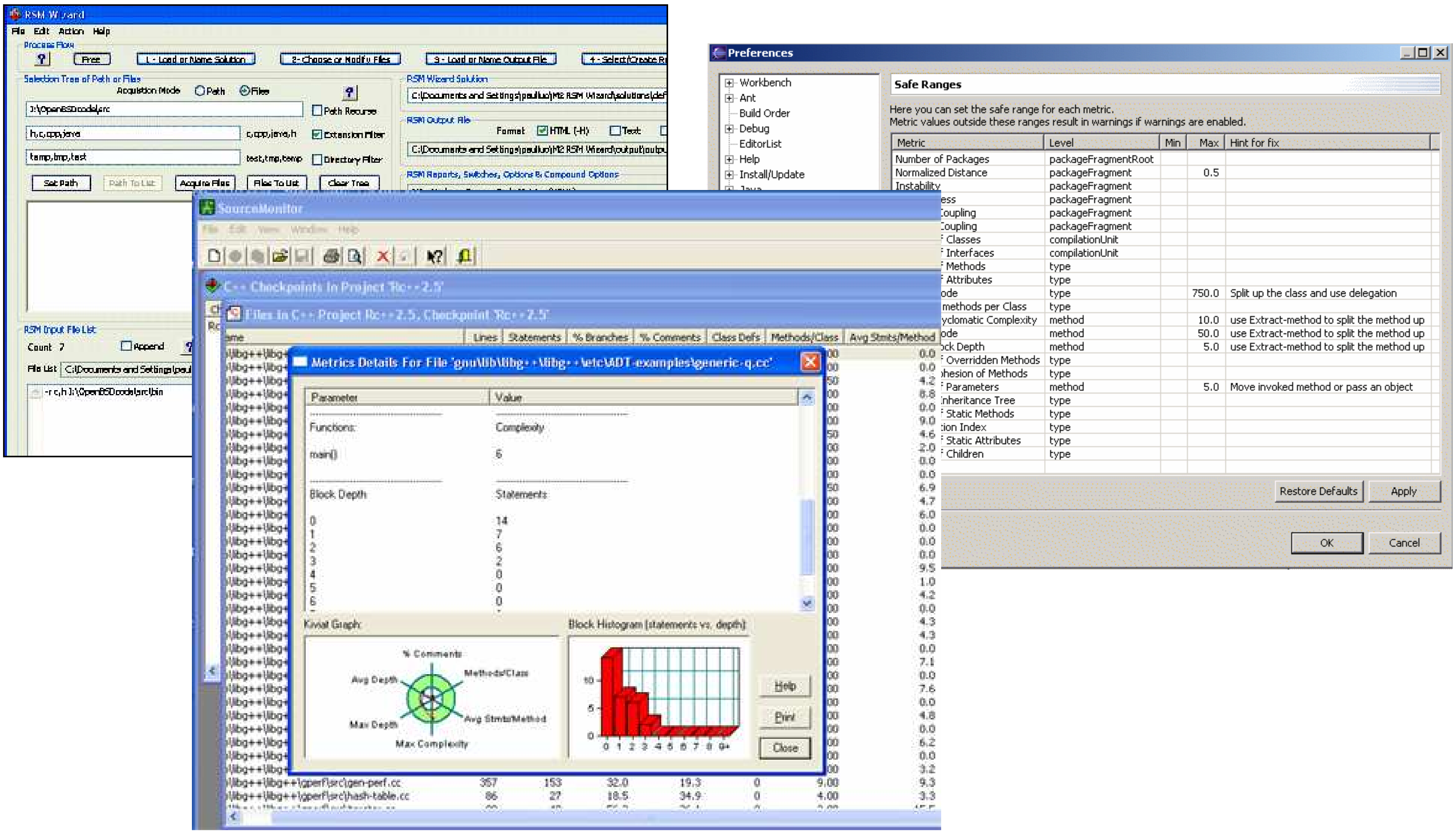
# Sub-categories of product metrics

- Control
- Volume
- Action
- Effort: Metrics measuring attributes of the mental effort required to implement
  - Halstead's effort metric

# Sub-categories of product metrics

- Control
- Volume
- Action
- Effort
- Modularity: Metrics measuring attributes related to the degree of modularity
  - Nesting depth greater than 10
  - Number of calls to other modules

# Commercial and open source tools that compute product metrics automatically



# The idea behind development metrics

- Metrics that measure attributes of the development process
  - Examined by many studies
  - Computed using information in change management and version control systems

# Rough grouping of development metrics

- Problems discovered prior to release: metrics that mention measuring attributes of the problems found prior to release.
  - Number of field problems in the prior release, Ostrand et. al.
  - Number of development problems, Fenton and Ohlsson
  - Number of problems found by designers Khoshgotaar et. al.

# Rough grouping of development metrics

- Problems discovered prior to release
- Changes to the product: metrics that mention measuring attributes of the changes made to the software product.
  - Reuse status, Pighin and Marzona
  - Changed source instructions, Troster and Tian
  - Number of deltas, Ostrand et. al.
  - Increase in lines of code Khoshgotaar et. al.

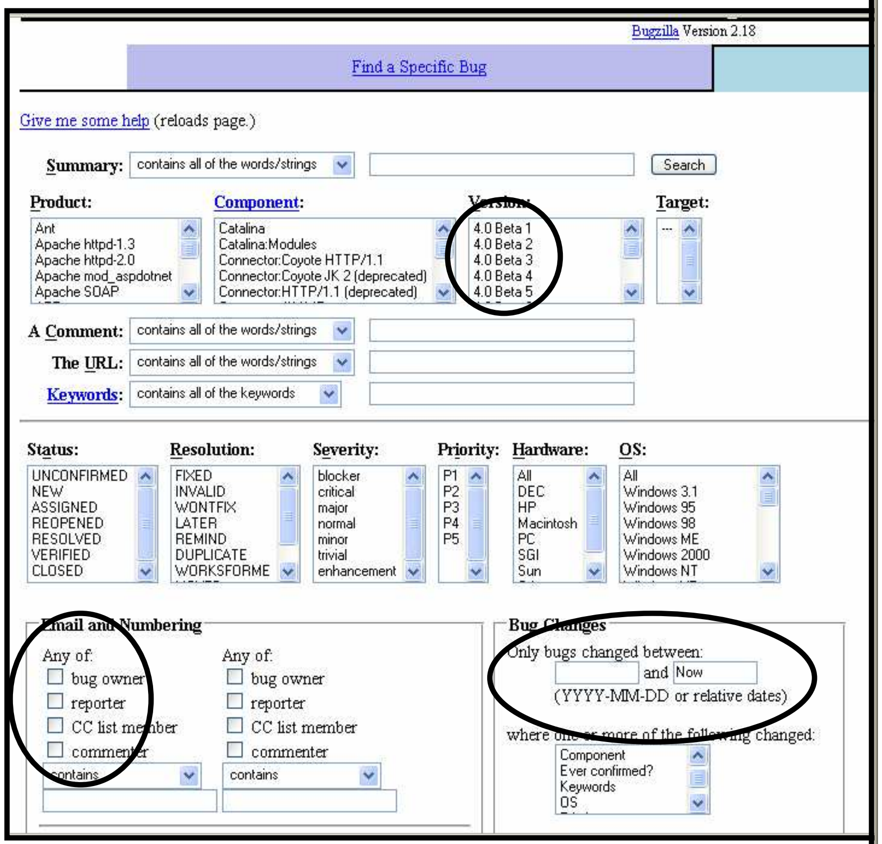
# Rough grouping of development metrics

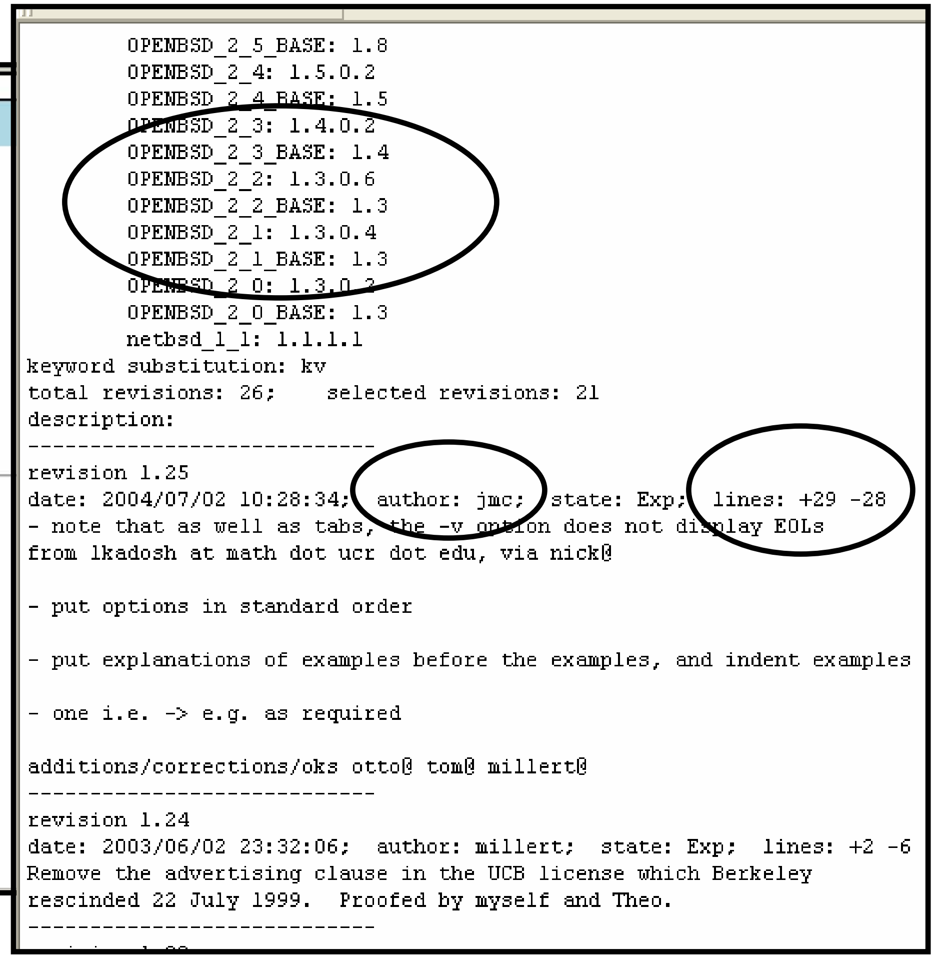
- Problems discovered prior to release
- Changes to the product
- People in the process: metrics that measure attributes of the people in the development process.
  - Number of different designers making changes, Khoshgoftaar et. al.
  - Number of updates by designers who had 10 or less total updates in entire company career, Khoshgoftaar et. al.

# Rough grouping of development metrics

- Problems discovered prior to release
- Changes to the product
- People in the process
- Process efficiency: metrics that measure attributes of the efficiency of the development process.
  - CMM level, Harter et. al.
  - Total development effort per 1000 executable statements, Selby and Porter

# Development metrics in bug tracking systems and change management systems



# The idea behind deployment and usage metrics

- Metrics that measure attributes of the deployment of the software system and usage in the field
  - Examined by few studies
  - No data source is consistently used

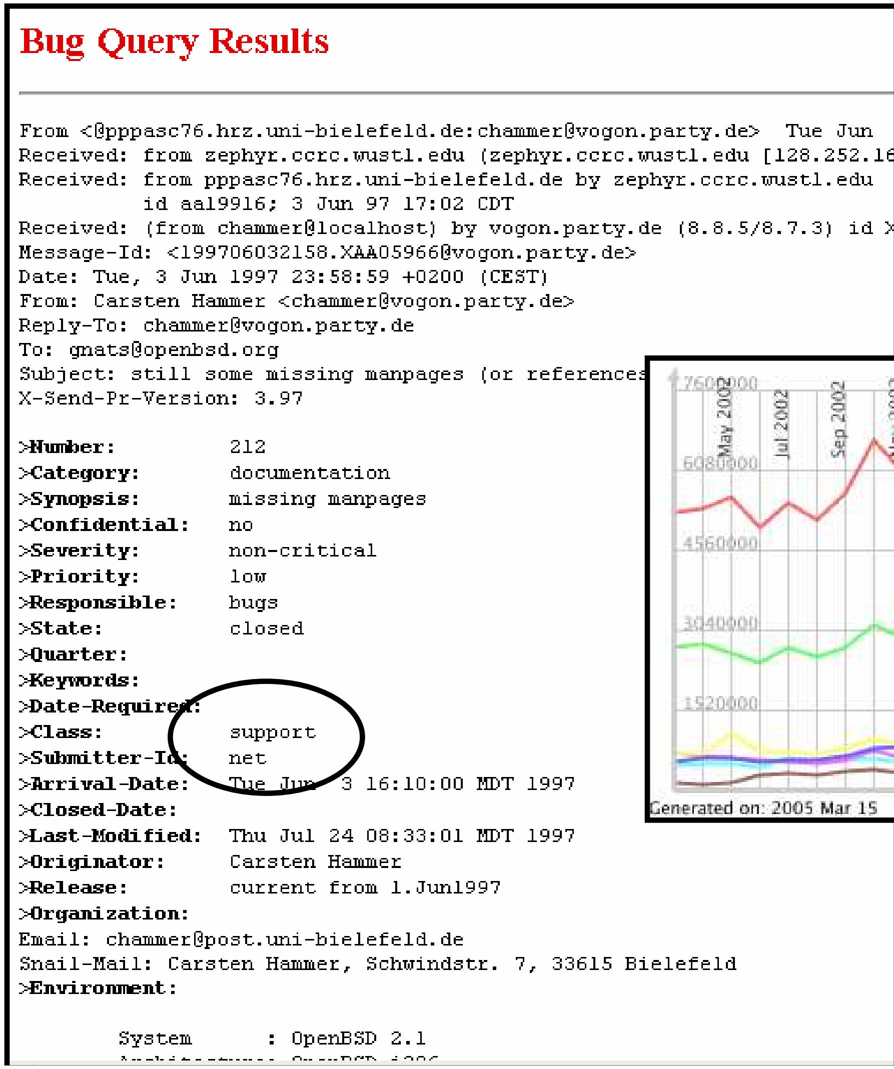
# Examples of deployment and usage metrics

- Khoshgoftaar et. al. (unit of observation is modules)
  - Proportion of systems with a module installed
  - Execution time of an average transaction on a system serving customers
  - Execution time of an average transaction on a systems serving businesses
  - Execution time of an average transaction on a tandem system

# Examples of deployment and usage metrics

- Khoshgoftaar et. al.
- Mockus et. al. (unit of observation is individual customer installations of telecommunications systems)
  - Number of ports on the customer installation
  - Total deployment time of all installations in the field at the time of installation

# Deployment and usage metrics may be gathered from download tracking systems or mailing lists

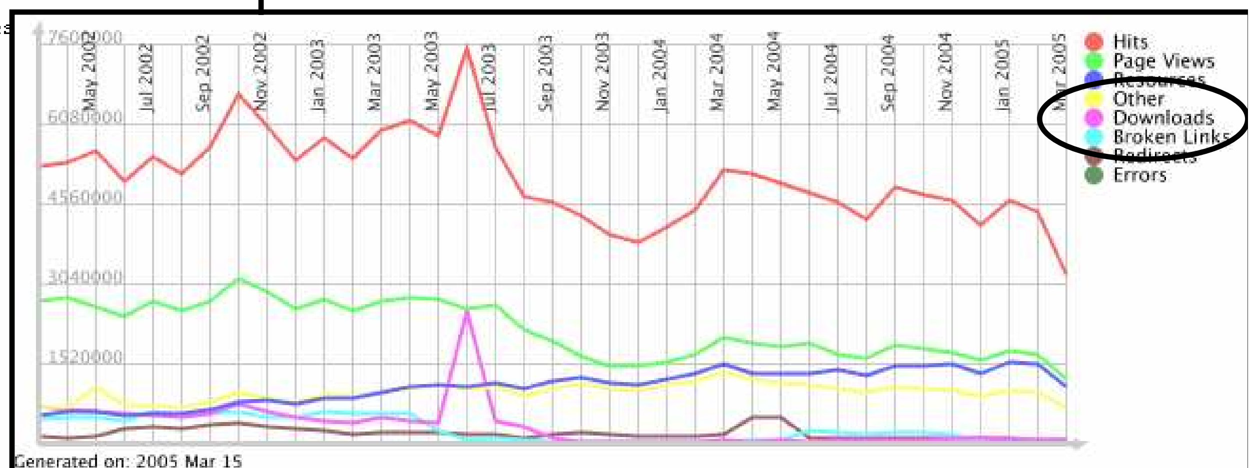


**Bug Query Results**

```
From <@pppasc76.hrz.uni-bielefeld.de:chammer@vogon.party.de>  Tue Jun
Received: from zephyr.ccrc.wustl.edu (zephyr.ccrc.wustl.edu [128.252.16
Received: from pppasc76.hrz.uni-bielefeld.de by zephyr.ccrc.wustl.edu
          id aa19916; 3 Jun 97 17:02 CDT
Received: (from chammer@localhost) by vogon.party.de (8.8.5/8.7.3) id X
Message-Id: <199706032158.XAA05966@vogon.party.de>
Date: Tue, 3 Jun 1997 23:58:59 +0200 (CEST)
From: Carsten Hammer <chammer@vogon.party.de>
Reply-To: chammer@vogon.party.de
To: gnats@openbsd.org
Subject: still some missing manpages (or references
X-Send-Pr-Version: 3.97

>Number:          212
>Category:        documentation
>Synopsis:        missing manpages
>Confidential:    no
>Severity:        non-critical
>Priority:        low
>Responsible:     bugs
>State:           closed
>Quarter:
>Keywords:
>Date-Required:
>Class:           support
>Submitter-Id:    net
>Arrival-Date:    Tue Jun  3 16:10:00 MDT 1997
>Closed-Date:
>Last-Modified:   Thu Jul 24 08:33:01 MDT 1997
>Originator:      Carsten Hammer
>Release:         current from 1.Jun1997
>Organization:
Email: chammer@post.uni-bielefeld.de
Snail-Mail: Carsten Hammer, Schwindstr. 7, 33615 Bielefeld
>Environment:

        System       : OpenBSD 2.1
```



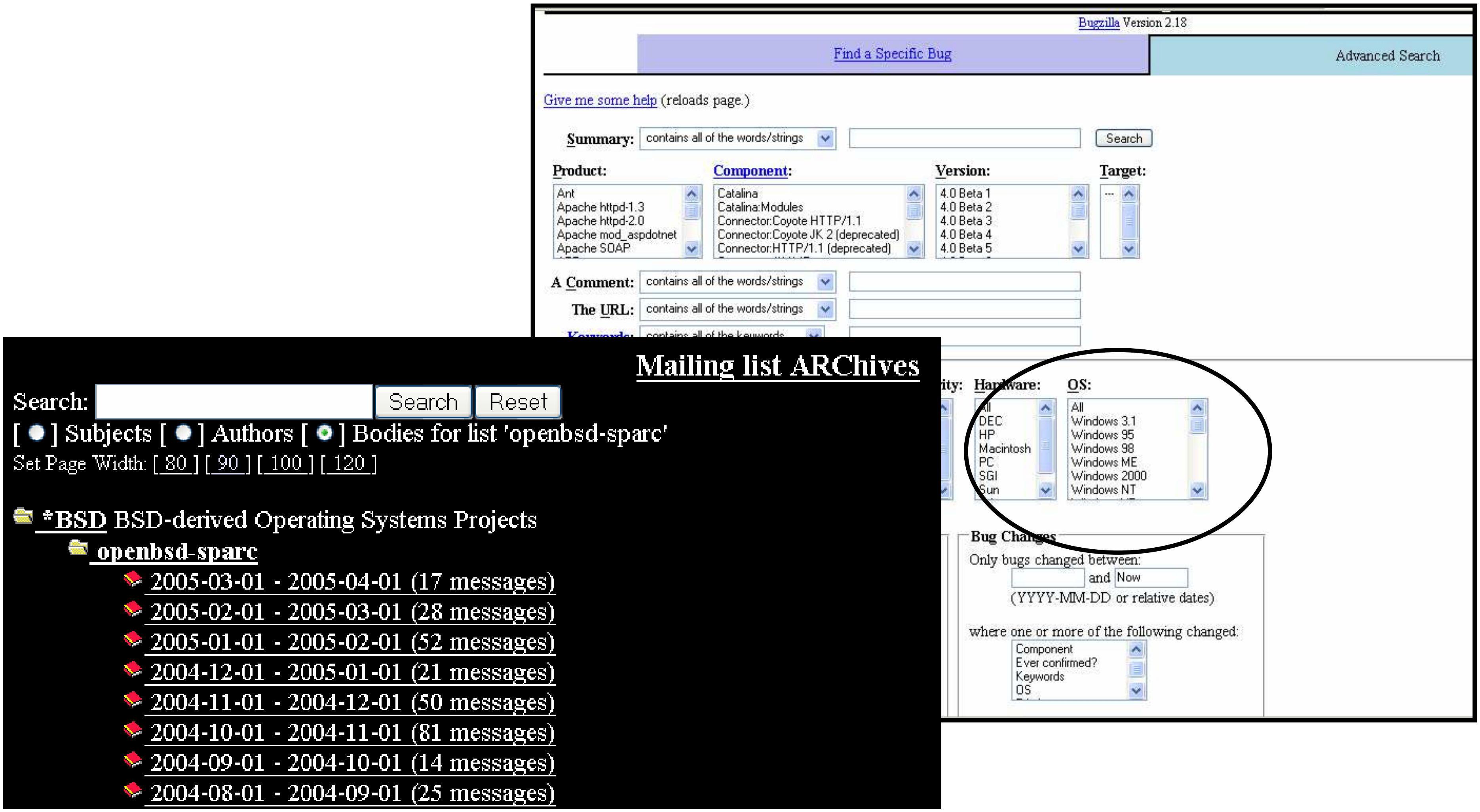
# The idea behind software and hardware configurations metrics

- Metrics that measure attributes of the software and hardware systems that interact with the software system in the field
  - Examined by few studies
  - No data source is consistently used

# Examples of hardware and software configurations metrics

- Mockus et. al. (unit of observation is individual customer installations of telecommunications systems)
  - Systems size of the installation (large or small/medium)
  - Operating system  of the installation (proprietary, Linux, or Windows)

# Software and hardware configurations metrics can be gathered from bug tracking systems and mailing lists



# Metrics to collect

- Prior work shows each category of metrics to be important
  - In general, more metrics will result in more accurate predictions
- A cost-benefit analysis is recommended (IEEE standard on software quality metrics)

# Lesson objectives

- ☑ Why predict field defects?
- ☑ When to use time based models?
- ☑ When to use metrics based models?
- What are the component of metrics based models?
  - ☑ What predictors to use?
  - What can I predict?
  - How do I predict?

# Predictions

- A relationship
  - What predicts field problems?
- A categorization
  - Is it risky or not (is the number of field problems above a threshold)?
- A number
  - What is the number of field problems?

# Importance of relationships

- Evaluation of the development process
- Better allocation of maintenance resources
- Improvement of testing efforts

Harter et. al. evaluated the development process by examining the CMM level of the organization

Bassin and Santhanam evaluate the development process by examining the distribution of ODC triggers of problems found during development

# Importance of relationships

- Evaluation of the development process
- Better allocation of maintenance resources
- Improvement of testing efforts

Mockus et. al establish the relationship between the operating systems platform (i.e. a proprietary OS, Linux, and Windows) and field problems

# Importance of relationships

- Evaluation of the development process
- Better allocation of maintenance resources
- Improvement of testing efforts
- Categorization predictions and number predictions are based on relationships

# How to evaluate relationships

1. Show high correlation between the predictor and field defects
2. Show that the predictor is selected using a model selection method
3. Show that the accuracy of predictions improves with the predictor included in the prediction model

# Importance of categorizations

- Focus testing in the appropriate places
  - Cost of fixing problems later is 10x times more expensive

# How to evaluate categorizations

- Type I error (false positive)
  - An observation is classified as risky when the observation is actually not risky

# How to evaluate categorizations

- Type I error
- Type II error (false negative)
  - An observation is classified as not risky when the observation is actually risky

# How to evaluate categorizations

- Type I error
- Type II error
- Overall rate of error
  - Either type I or type II error

# Trade-offs between type I and type II error

- Reducing false negatives is usually more important
  - Main objective of classification is to focus resources on risky modules to prevent field problems (Jones et. al.)
- Resources are limited so high type I errors and overall errors are also not desirable
  - The costs of misclassification need to be considered in each setting to select an optimal balance (Khoshgoftaar et. al.)

# Importance of a numerical output

- Allocate the appropriate amount of maintenance resources
  – Not having sufficient resources may delay fixing field problems, which results in reduced customer satisfaction (Chulani et. al.)
  – Allocating too many maintenance resources hinders other efforts (e.g. development)

# Importance of a numerical output

- Allocate the appropriate amount of maintenance resources
- Plus all the benefits of a categorization and a relationship

# How to evaluate a numerical output

- The absolute average error (AAE) and its standard deviation
  - How much a typical prediction will be off by on average

# How to evaluate a numerical output

- The absolute average error (AAE) and its standard deviation
- The average relative error (ARE) and its standard deviation
  - The AAE can be misleading when the predicted number of field problems differs significant between observations
  - Relative to the actual number of field problems, how much a typical prediction will be off by on average

# Lesson objectives

- ☑ Why predict field defects?
- ☑ When to use time based models?
- ☑ When to use metrics based models?
- What are the component of metrics based models?
  - ☑ What predictors to use?
  - ☑ What can I predict?
  - How do I predict?

# The idea behind modeling methods

- Build models using historical information on the predictors and the observed field defects
- Predicts for a new observation given predictors’ values

# Level 1 modeling techniques

- Linear modeling (logistic regression)
- Trees
- Discriminant analysis
- Rules
- Neural networks
- Clustering
- Sets
- Linear programming
- Heuristics or any level 2 method with heuristics

# Example: the trees technique

- Creating partitions based on predictor value that minimizes the error in classifications within partitions
- Repeat process until
  - Error within each partition is below some limit
  - Number of observations within each partition is below some limit
- The observations within each partition determine the class of the partition

# Example description

- Predictor A has three values:
  - 1, 2, 3
- Predictor B has two values:
  - 1, 2
- The field problem metric has two classes (values):
  - 1 (at least 1 field problem), 0 (no field problems)

# Example training set

| Value of Predictor A | Value of Predictor B | Class of the field problems metric |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 2 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 3 | 2 | 0 |
| 3 | 1 | 0 |
| 3 | 2 | 1 |

# Example stopping criteria

- The measure of error is:
  - $\Sigma_{\text{partitions}} \Sigma_{\text{all observations in partition}} |y_i - \tilde{y}|$
- $\tilde{y}$ = mean of classifications in the partition
- The minimum error in partition:
  - 0
- The minimum number of observation in partition:
  - 2

# Example iteration 1

- Predictor A <=1
  - error in partition 1 (A<=1) (0 + 0 + 0 + 0) = 0

| Value of Predictor A | Value of Predictor B | Class of the field problems metric |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 2 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 3 | 2 | 0 |
| 3 | 1 | 0 |
| 3 | 2 | 1 |

# Example iteration 1

- Predictor A <=1
  - error in partition 1 (A<=1) (0 + 0 + 0 + 0) = 0
  - error in partition 2 (A>1) 1/2 + 1/2 +1/2 + 1/2 + 1/2 + 1/2) = 3
- total error = 3

| **Value of Predictor A** | **Value of Predictor B** | **Class of the field problems metric** |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 2 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 3 | 2 | 0 |
| 3 | 1 | 0 |
| 3 | 2 | 1 |

# Example iteration 1

- Predictor A<=2
  - error in partition 1 (A<=2) (1/3 + 1/3 +1/3 + 1/3 + 2/3 + 2/3) =2.667

| Value of Predictor A | Value of Predictor B | Class of the field problems metric |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 2 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 3 | 2 | 0 |
| 3 | 1 | 0 |
| 3 | 2 | 1 |

# Example iteration 1

- Predictor A<=2
  - error in partition 1 (A<=2) (1/3 + 1/3 +1/3 + 1/3 + 2/3 + 2/3) =2.667
  - error in partition 2 (A>2) (1/4 + 1/4 +1/4 +3/4) = 1.5
- total error = 4.167

| Value of Predictor A | Value of Predictor B | Class of the field problems metric |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 2 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 3 | 2 | 0 |
| 3 | 1 | 0 |
| 3 | 2 | 1 |

# Example iteration 1

- Predictor B <=1
  - error in partition 1 (B<=1) (1/3 + 1/3 +1/3 + 1/3 + 2/3 + 2/3) = 2.667

| Value of Predictor A | Value of Predictor B | Class of the field problems metric |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 2 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 3 | 2 | 0 |
| 3 | 1 | 0 |
| 3 | 2 | 1 |

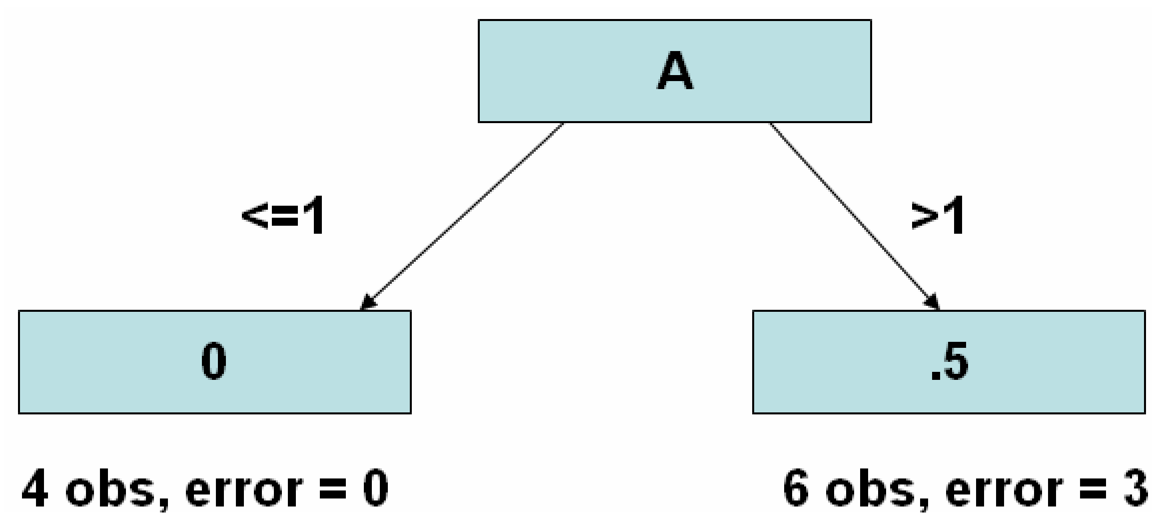
# Example iteration 1

- Predictor B <=1
  - error in partition 1 (B<=1) (1/3 + 1/3 +1/3 + 1/3 + 2/3 + 2/3) = 2.667
  - error in partition 2 (B>1) (1/4 + 1/4 +1/4 +3/4) = 1.5
- total error = 4.167

| Value of Predictor A | Value of Predictor B | Class of the field problems metric |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 2 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 3 | 2 | 0 |
| 3 | 1 | 0 |
| 3 | 2 | 1 |

# Example iteration 1

- Based on error, partition using A<=1



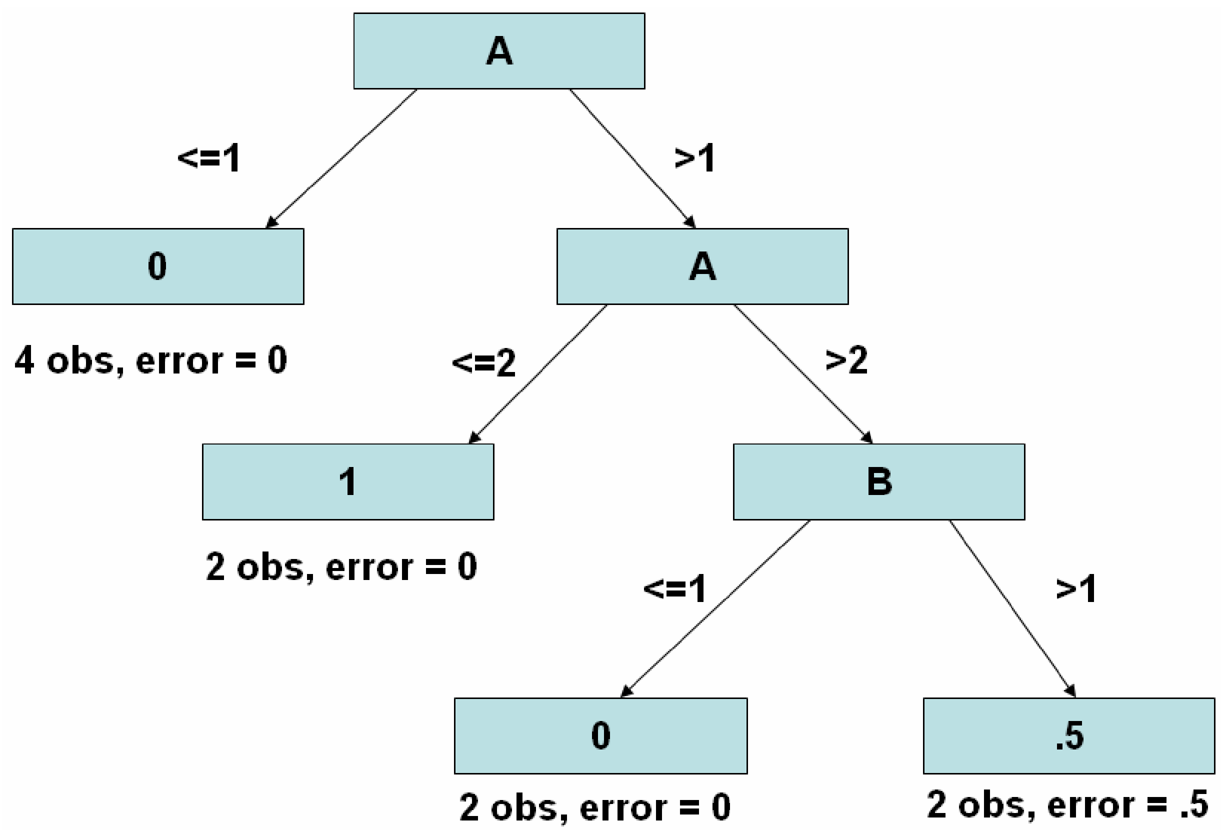

Recall stopping criteria is error = 0 or obs <=2

# Example iteration 2

- Predictor A <=2
  - error in partition 1 (A<=2) (0 + 0) = 0

| Value of Predictor A | Value of Predictor B | Class of the field problems metric |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 1 | 1 | 0 |
| 1 | 2 | 0 |
| 2 | 1 | 1 |
| 2 | 1 | 1 |
| 3 | 1 | 0 |
| 3 | 2 | 0 |
| 3 | 1 | 0 |
| 3 | 2 | 1 |

# Example iteration 3

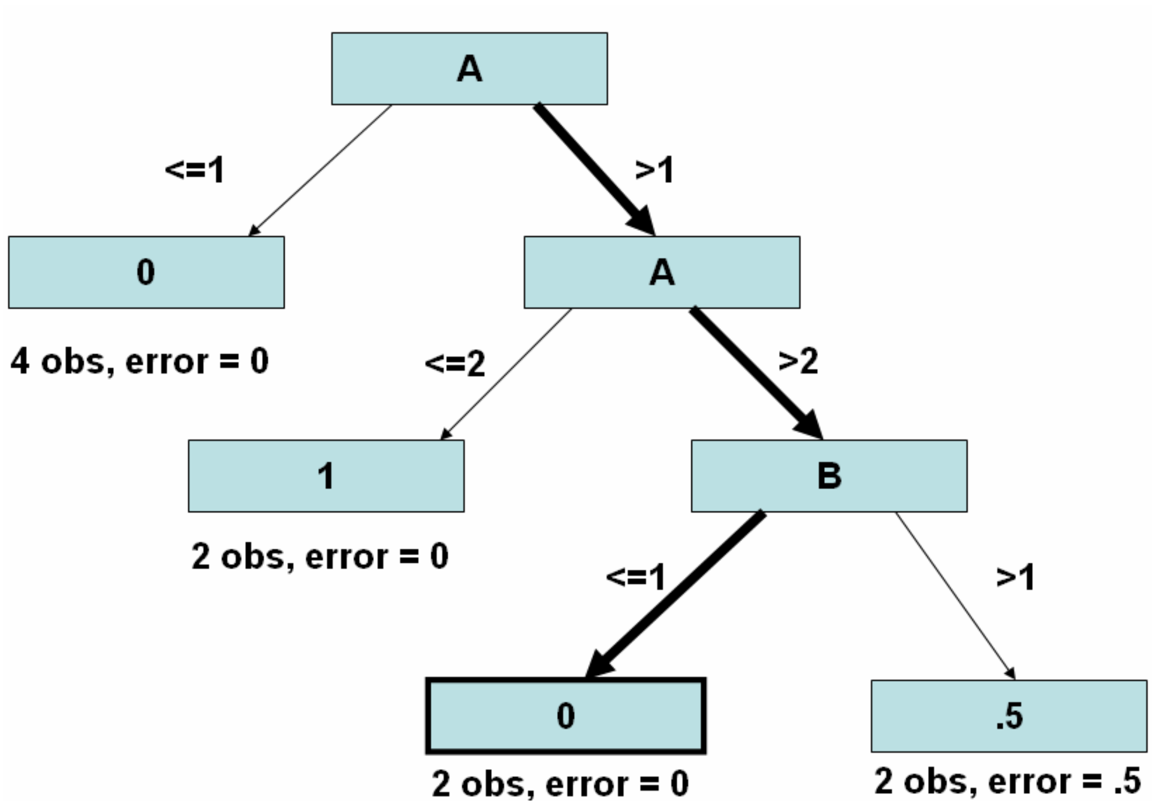


# Example: the trees technique

- To predict, an observation is sent through the tree until it reaches a leaf
- Class of the leaf (i.e. partition) is taken to be the predicted value

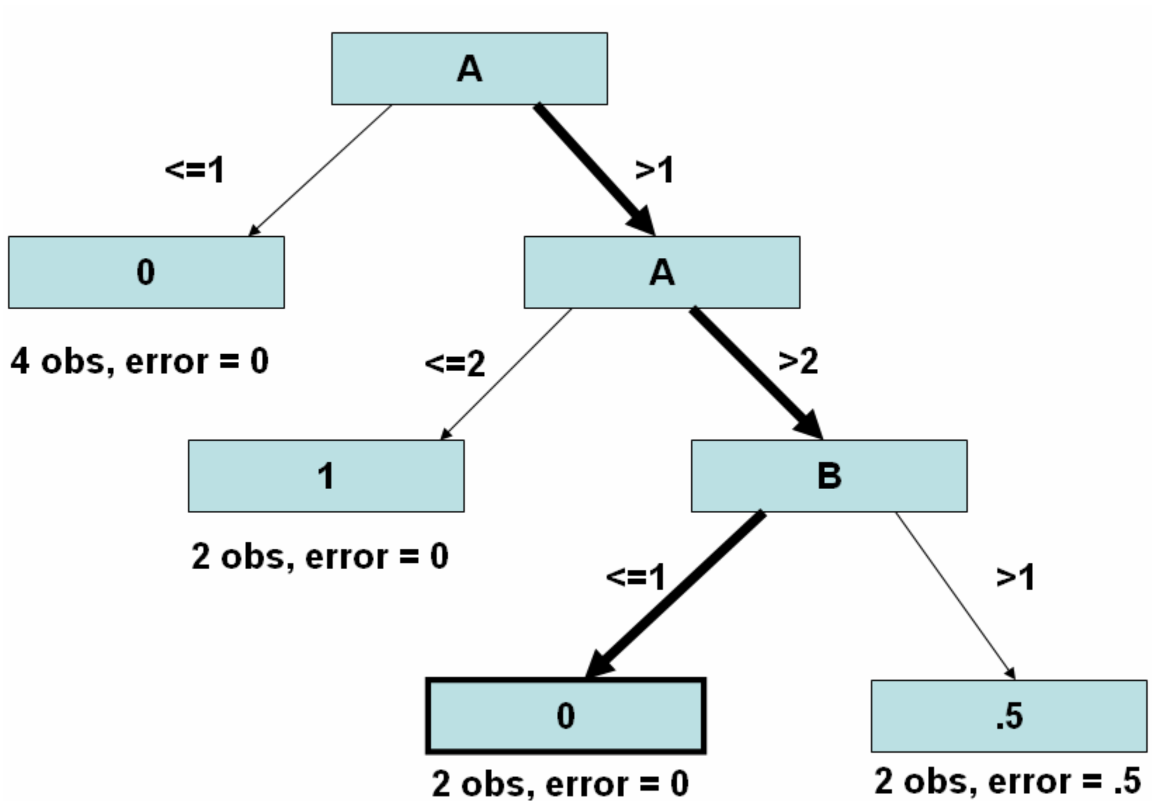
# Example prediction

- Example: A = 3, B = 1



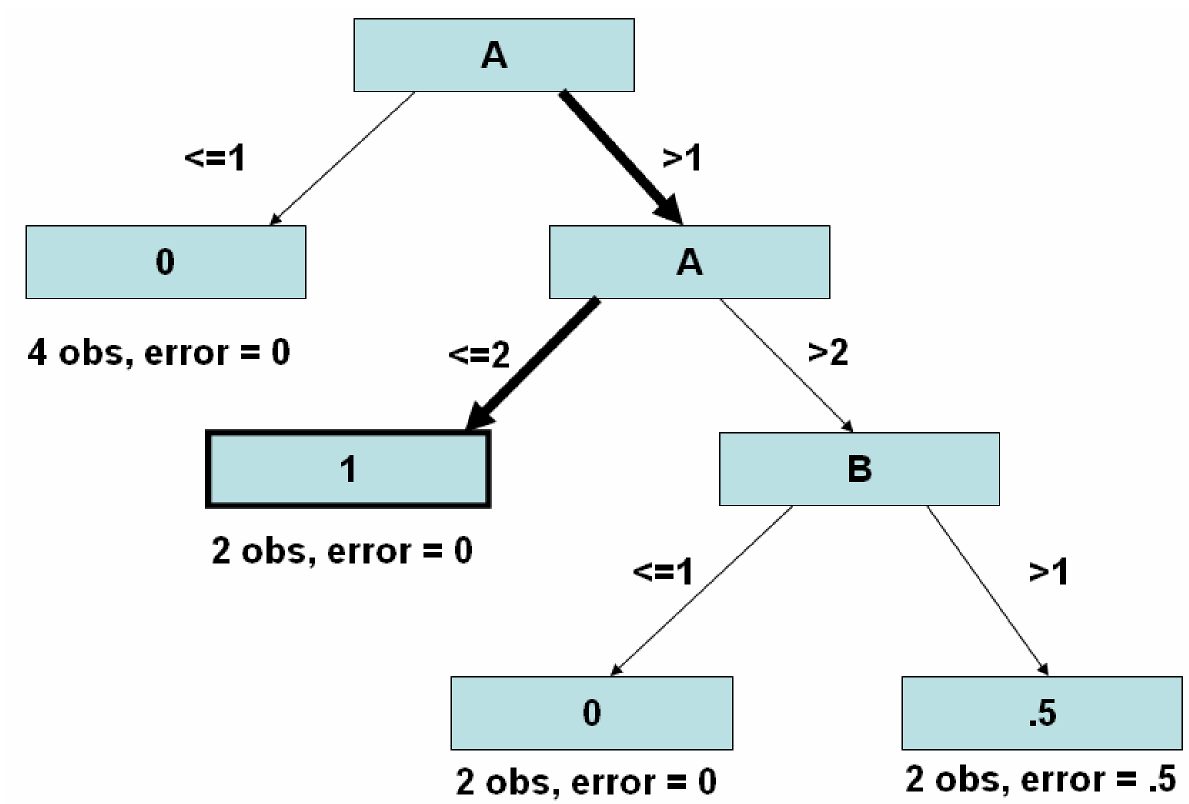
# Example prediction

- Example: A = 3, B = 1
- Classification = 0 (not risky)



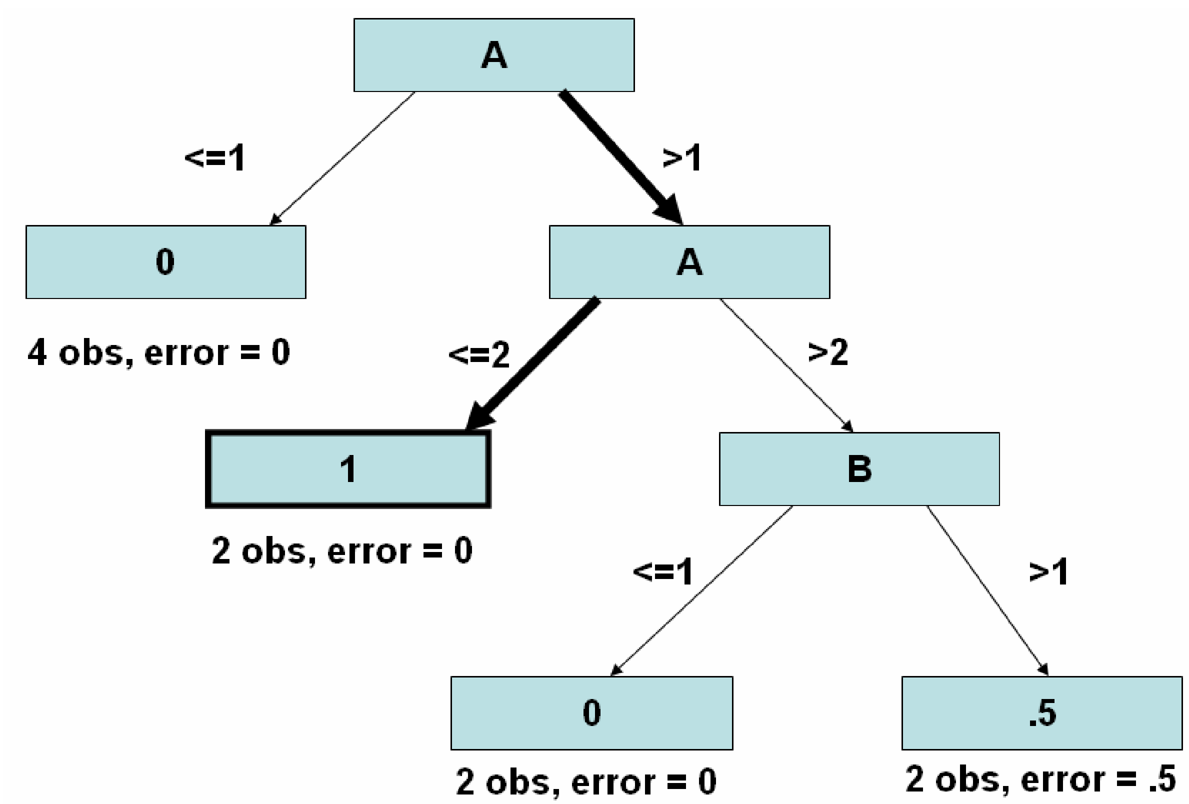
# Example prediction

- Example: A = 2, B = 2



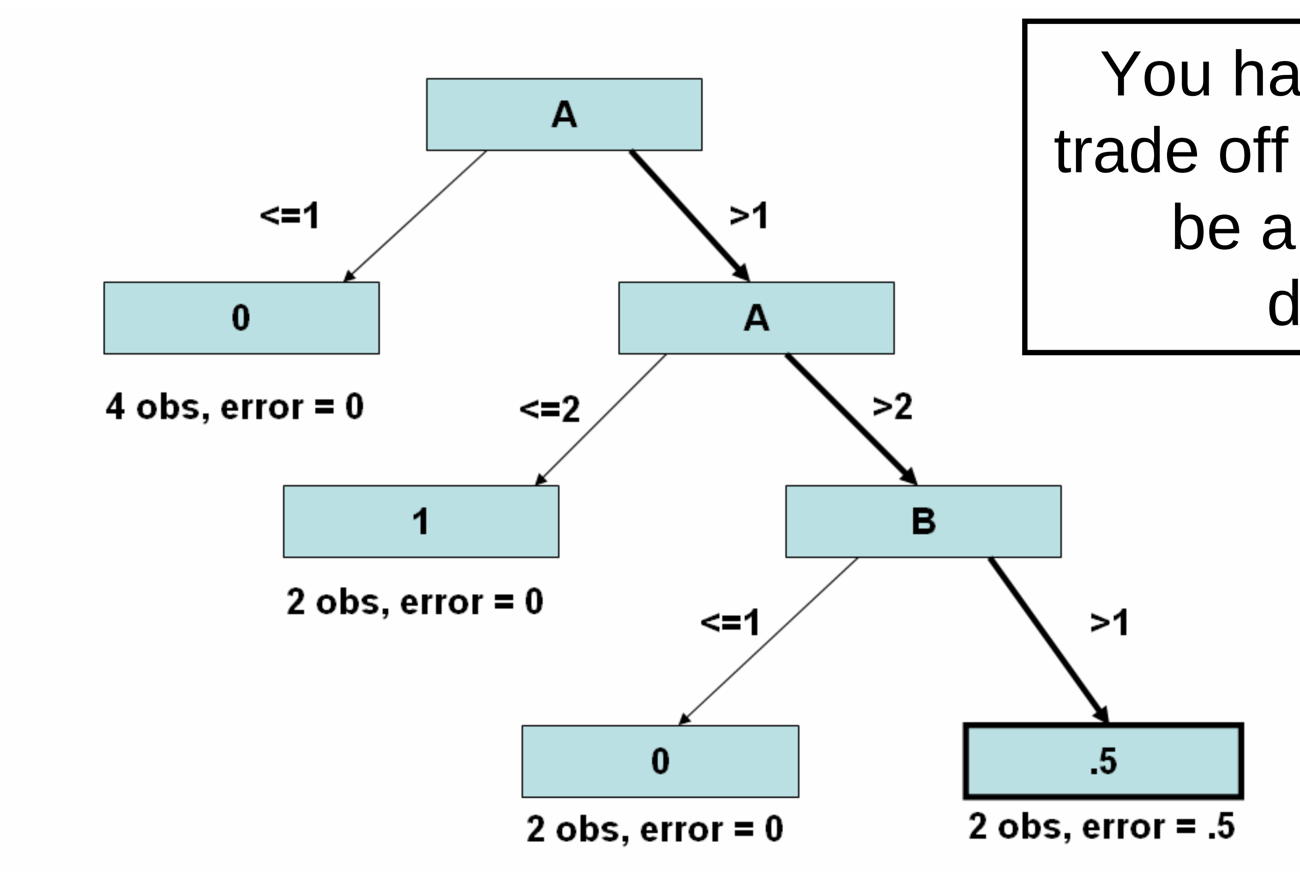
# Example prediction

- Example: A = 2, B = 2
- Classification =1 (risky)



# Example prediction

- Example A=3, B=2
- Classification = .5 ?




You have to make a trade off here, but it will be an informed decision

# Level 2 modeling techniques

- Linear modeling (linear regression and negative binomial regression)
- Non-linear regression
- Trees
- Neural networks

# Lesson objectives

- ☑ Why predict field defects?
- ☑ When to use time based models?
- ☑ When to use metrics based models?
- What are the component of metrics based models?
  - ☑ What predictors to use?
  - ☑ What can I predict?
  - ☑ How do I predict?

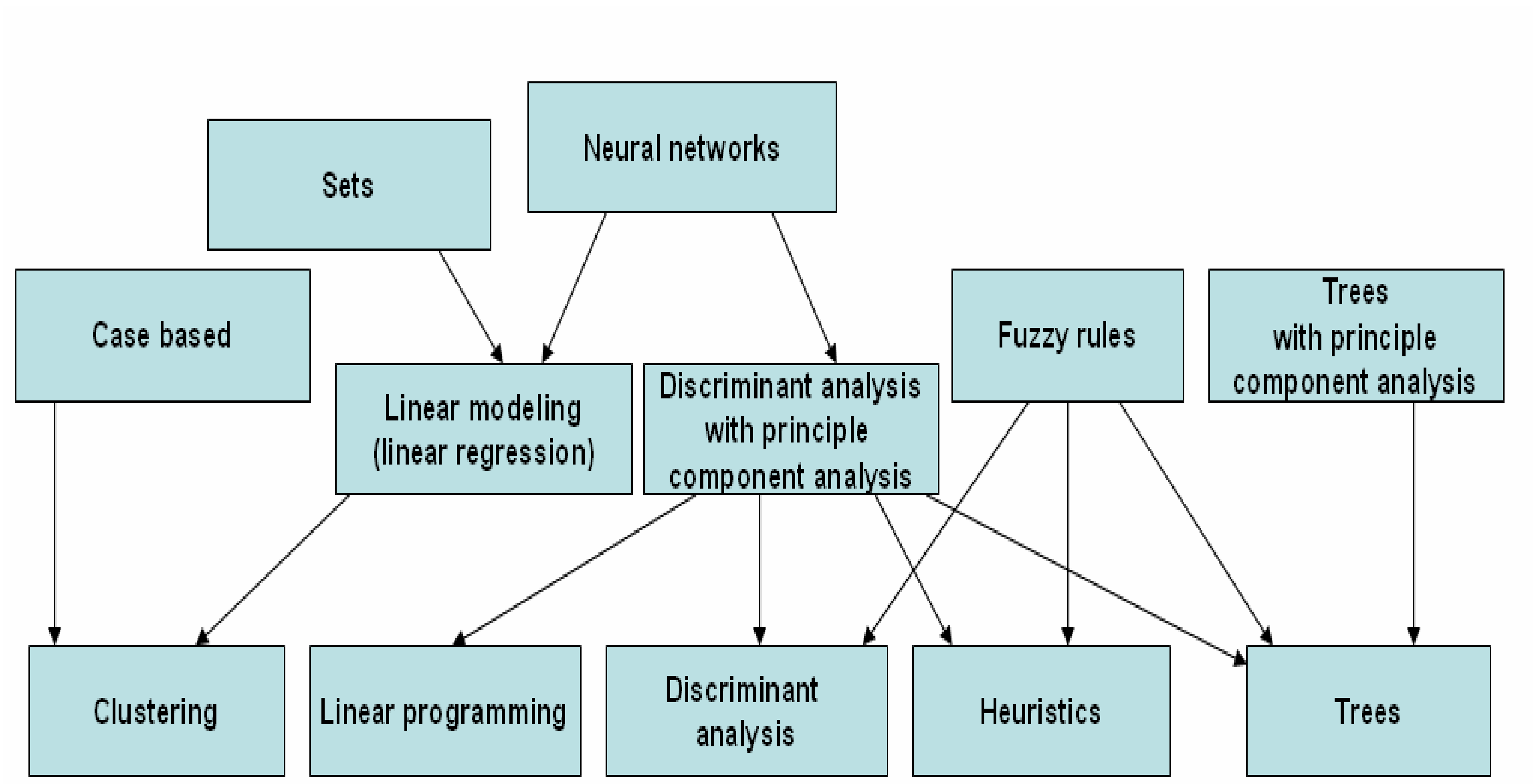
# Recap

- Field defect predictions can lower the costs of field defects by:
  - Guiding testing
  - Improving maintenance resource allocation
  - Guiding process improvement
  - Adjusting deployment
  - Enabling software insurance
- Metrics based models are better when deployment and testing environment differ or when there is insufficient resources to test all configurations

# Recap

- Metrics based models use:
  - Product metrics (most often)
  - Development metrics (next most often)
  - Deployment and usage metrics (infrequently)
  - Software and hardware configurations metrics (infrequently)
- Trees is the most widely used method to produce a level 1 output

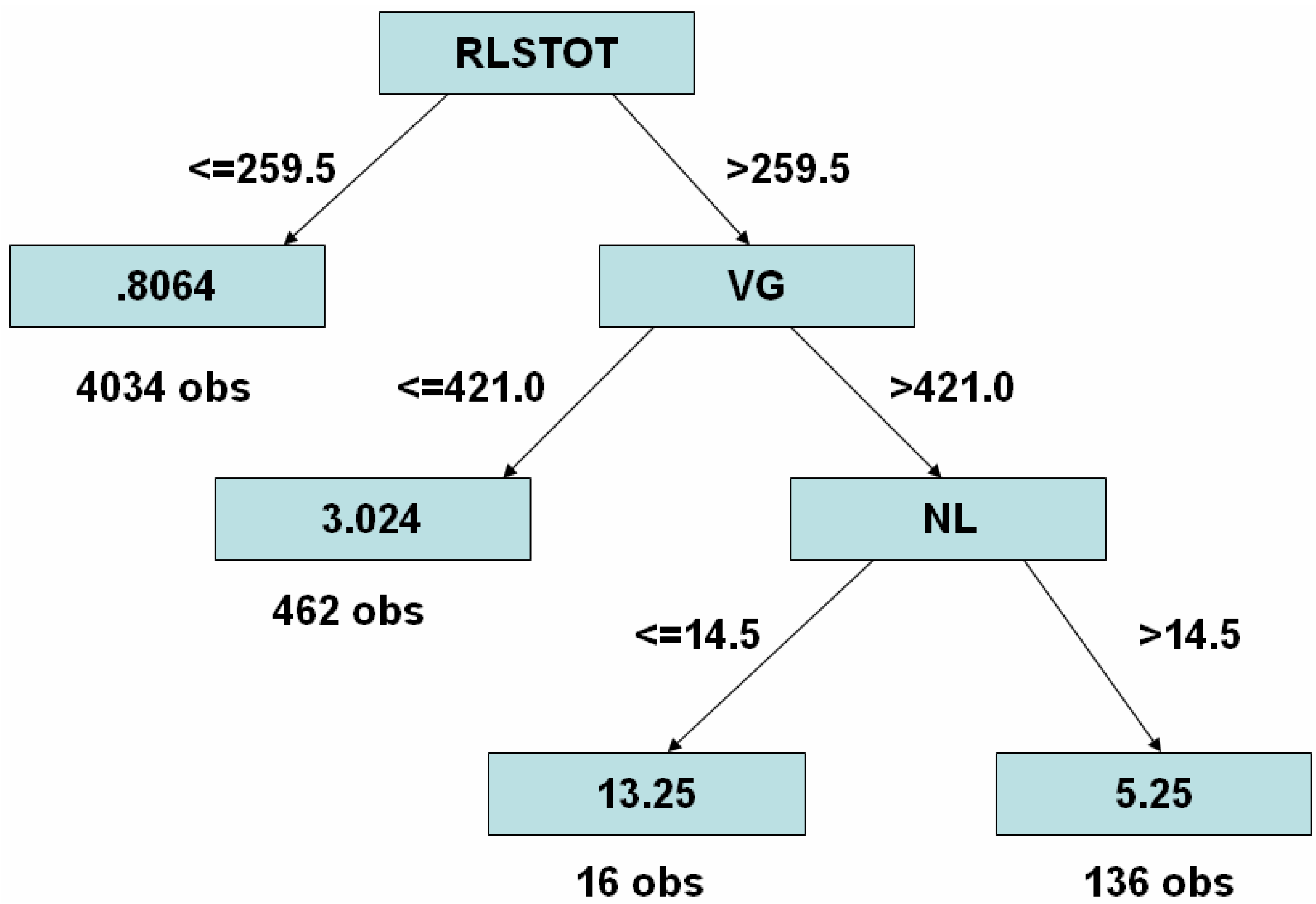
# Partial ordering of methods using accuracy



# Drawback of using accuracy as the only criterion of evaluation

- Sometimes accurate predictions is not the objective:
  - Planning for improvement

# An explicable model



# A less explicable model

| Metric | Component 1 | Component 2 | Component 3 |
|---|---|---|---|
| RLSTOT | .901 | .359 | .137 |
| NL | .880 | .370 | .134 |
| PCSTOT | .719 | .545 | .316 |
| NELTOT | .683 | .593 | .334 |
| TCT | .359 | .864 | .216 |
| UCT | .426 | .830 | .245 |
| VG | .597 | .724 | .309 |
| IFTH | .599 | .681 | .357 |
| NDI | .177 | .265 | .939 |

Field problems =

.520

+ 1.233 (ISCHG)

+ .541 (ISNEW)

+ .577 (Component 3)

+ .368 (Component 1)

+ .338 (Component 2)