“Visualizing Software Changes”

Stephen G. Eick, Todd L. Graves, Alan F. Karr, Audris Mockus, and Paul Schuster

Paper Presentation by

Istehad Chowdhury

Introduction

- Objective:
  - Come up with visualization tools and techniques to help understand and manage software change process

Change Data

- Components:
  - Time:
  - Software Space
  - Developer
  - Type
  - Size
  - Effort
  - Interval
- These are dimensions and are independent

Measures

- “Measures are the responses the organization wishes to understand and control”
- Includes variables such as
  - Size
  - Effort
  - Interval
  - Number of faults etc.
- The goal of the analysis is to understand the relation between components and measures

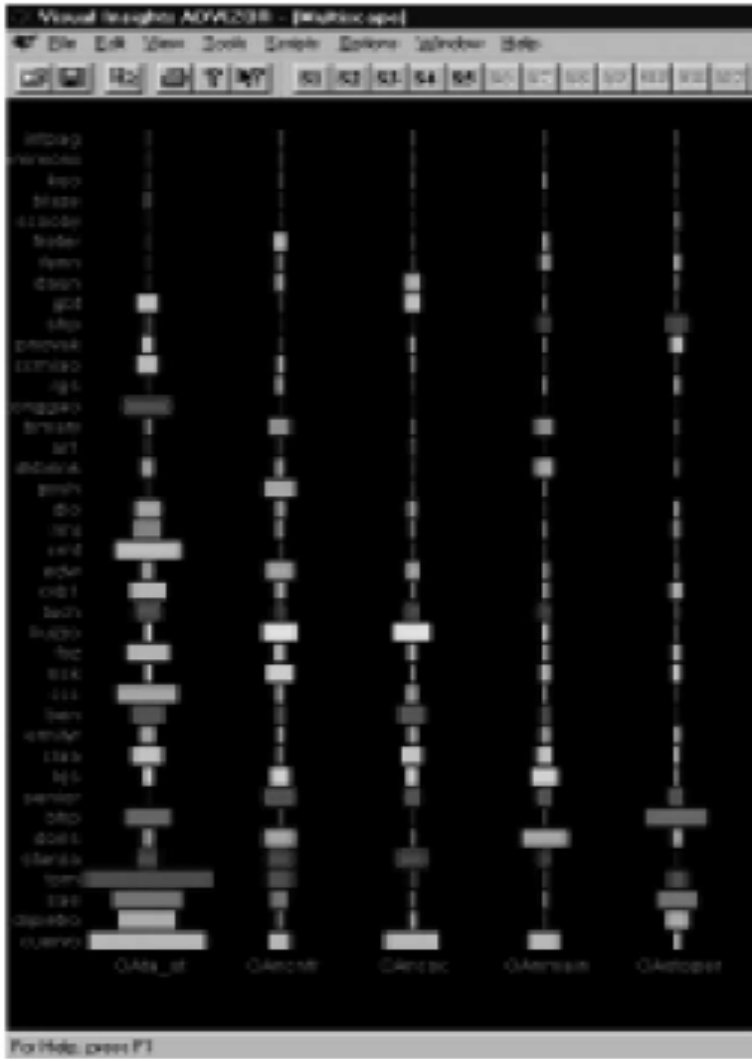
Evaluation

- Evaluated techniques on a large telephone switch software:
  - 15yrs, 5000 modules, 50 sub systems
- Our favourite of telecom software

The five metaphors

Matrix View

- Row = Developer
- Column= Module
- Row vs Column i.e.
- Developer vs Module
- Size of change = width of bar
- Good
  - No overplotting
  - Effectively show huge data sets
  - Flexibility (pan and zoom)
- Bad
  - No natural ordering

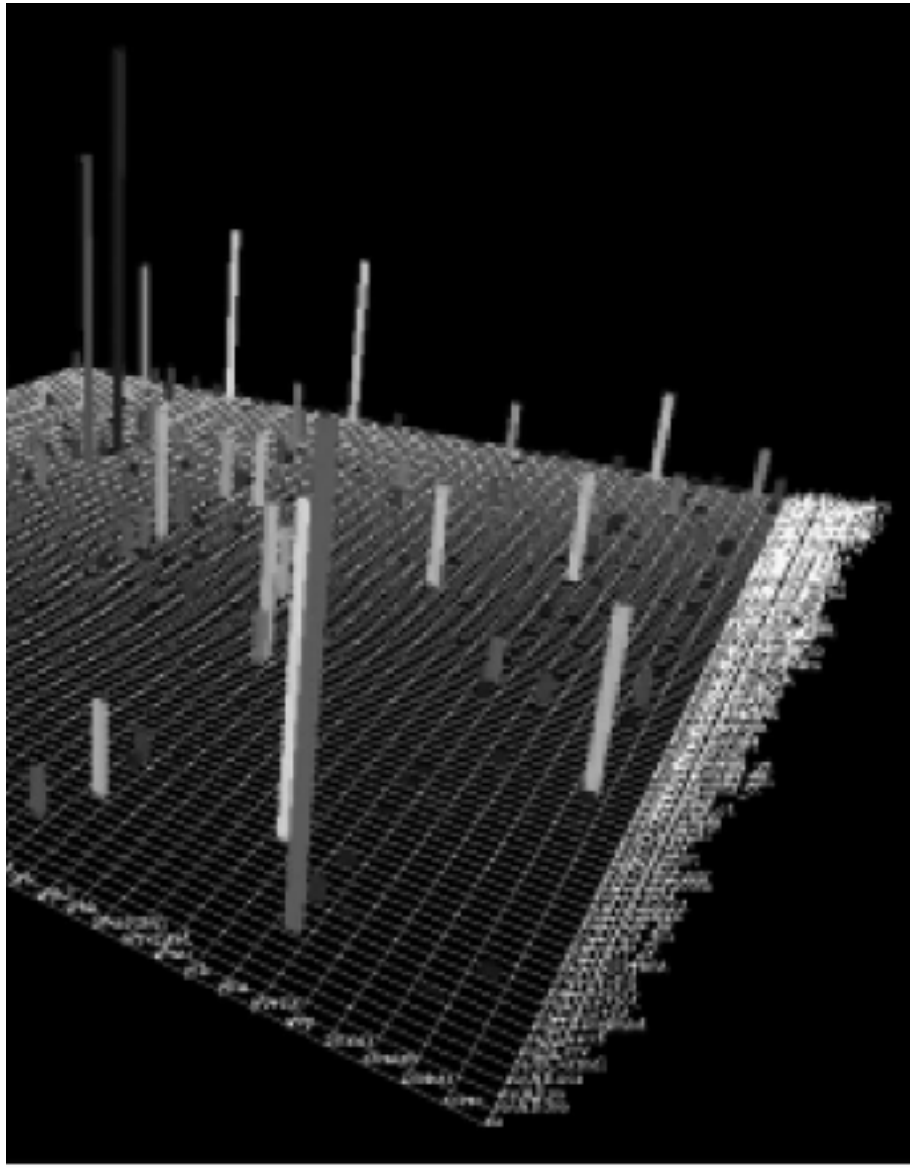


Cityscape View

- Similar to previous
- Height = number of changes
- Color = number of changes (redundant but good!)

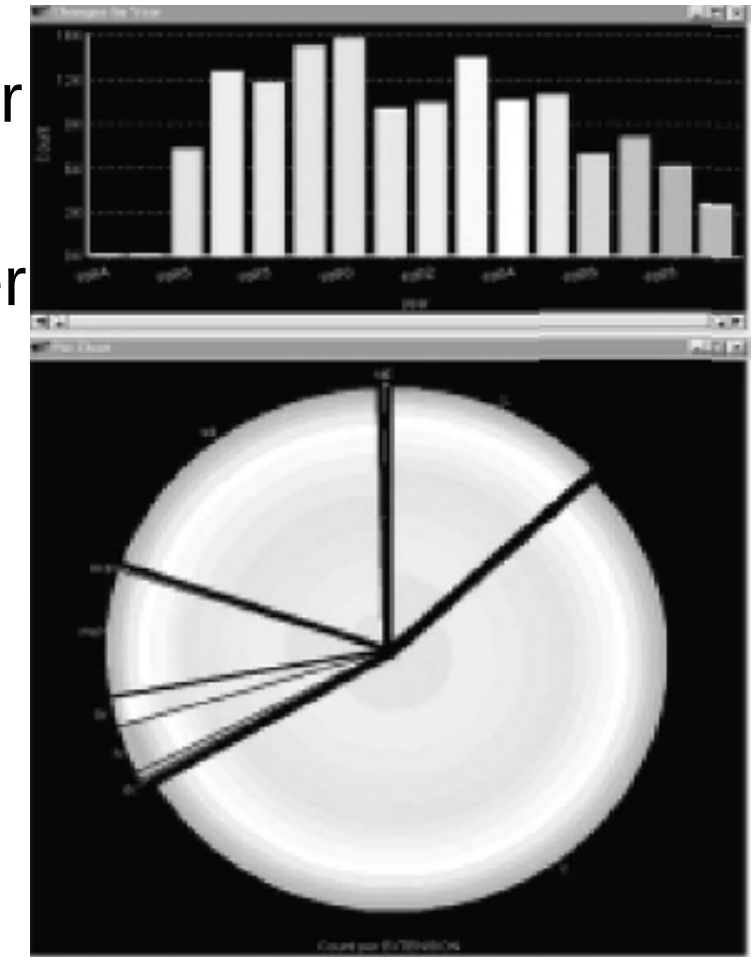
- Good
  - “More compelling”
  - Can use rotation

- Bad
  - Occlusion
  - Rotation is not intuitive



Bar and Pie Charts

- Bar = number of changes per year
- Pie = Number of changes per file type
- Good
  - Trends
  - Common
- Bad
  - Bar: over plotting
  - Bar charts scales not Pies do not do very well.



Bar and Pie Charts *cont`d*

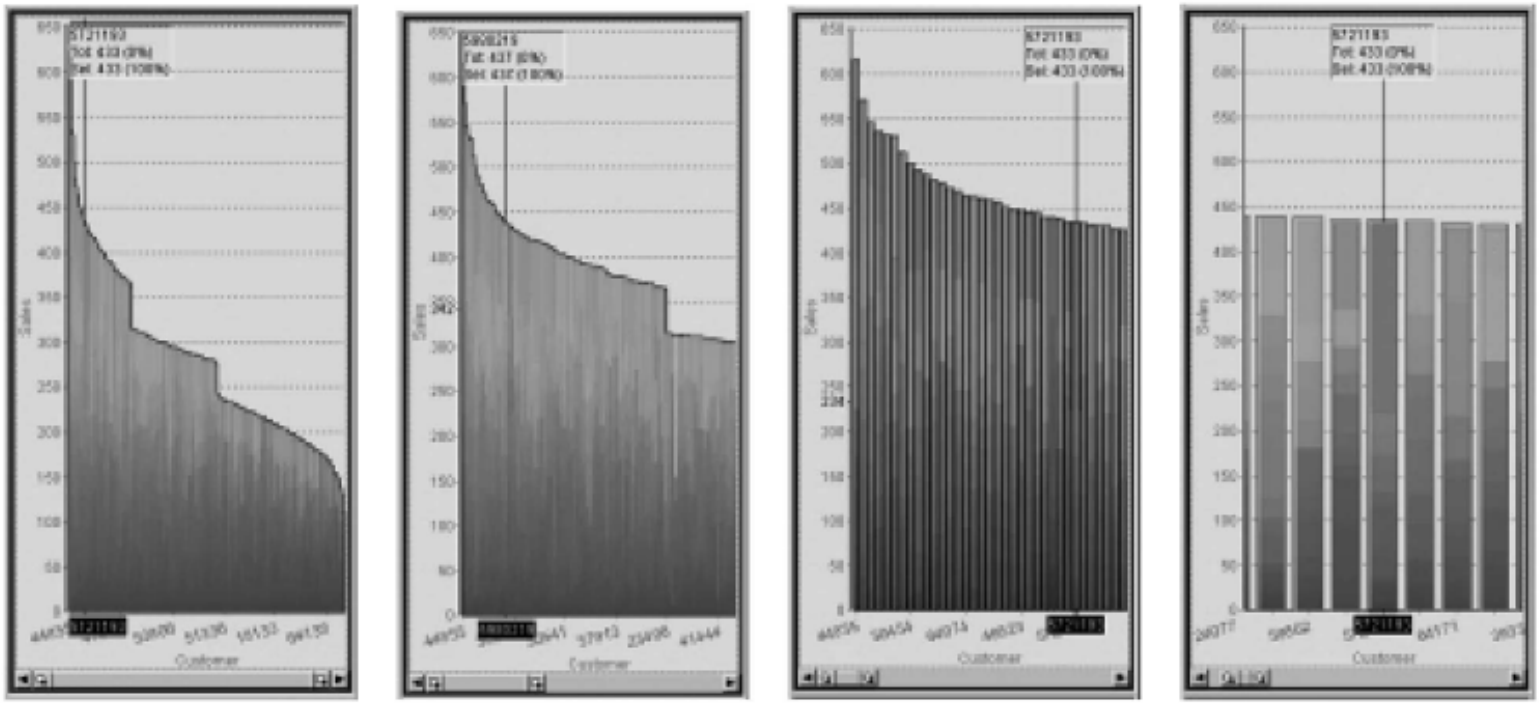
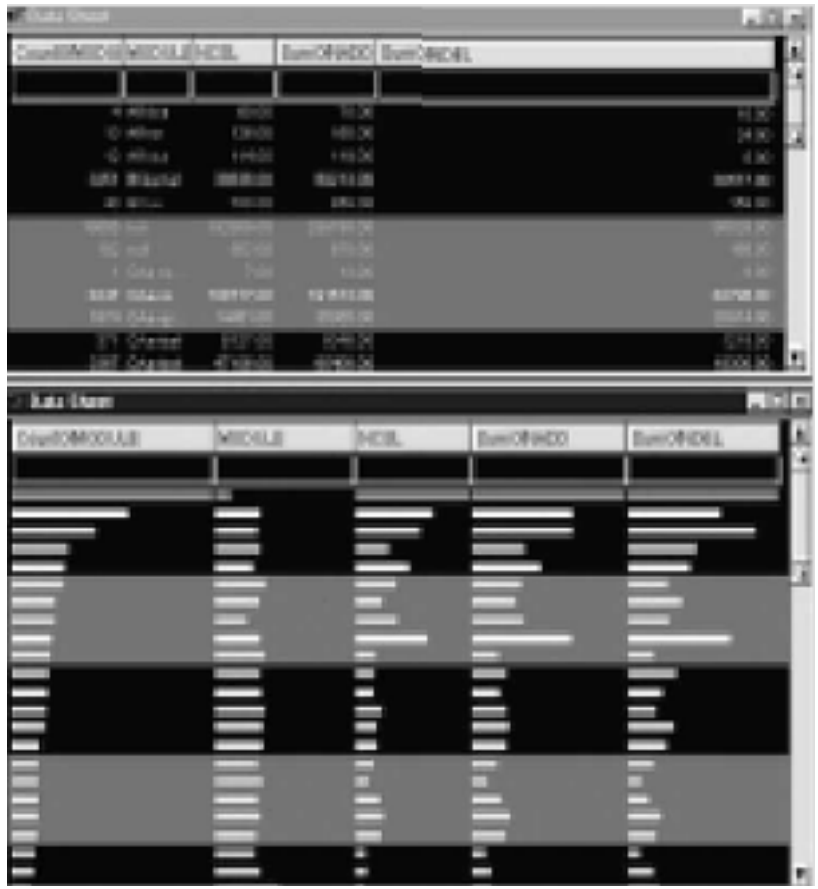


Fig. 4. Bar chart scalability is increased by using levels of rendering detail, a red over-plotting indicator along the top (left), and a 1,000–to–1 zoombar along the bottom.

Bars shows MR by programmers and color tied to severity

Data Sheets

- A scrollable text visualization shows changes in modules, subsystems etc.
- Good
  - Access to details
  - Scalability
- Bad
  - Sometimes to detail



Network Views

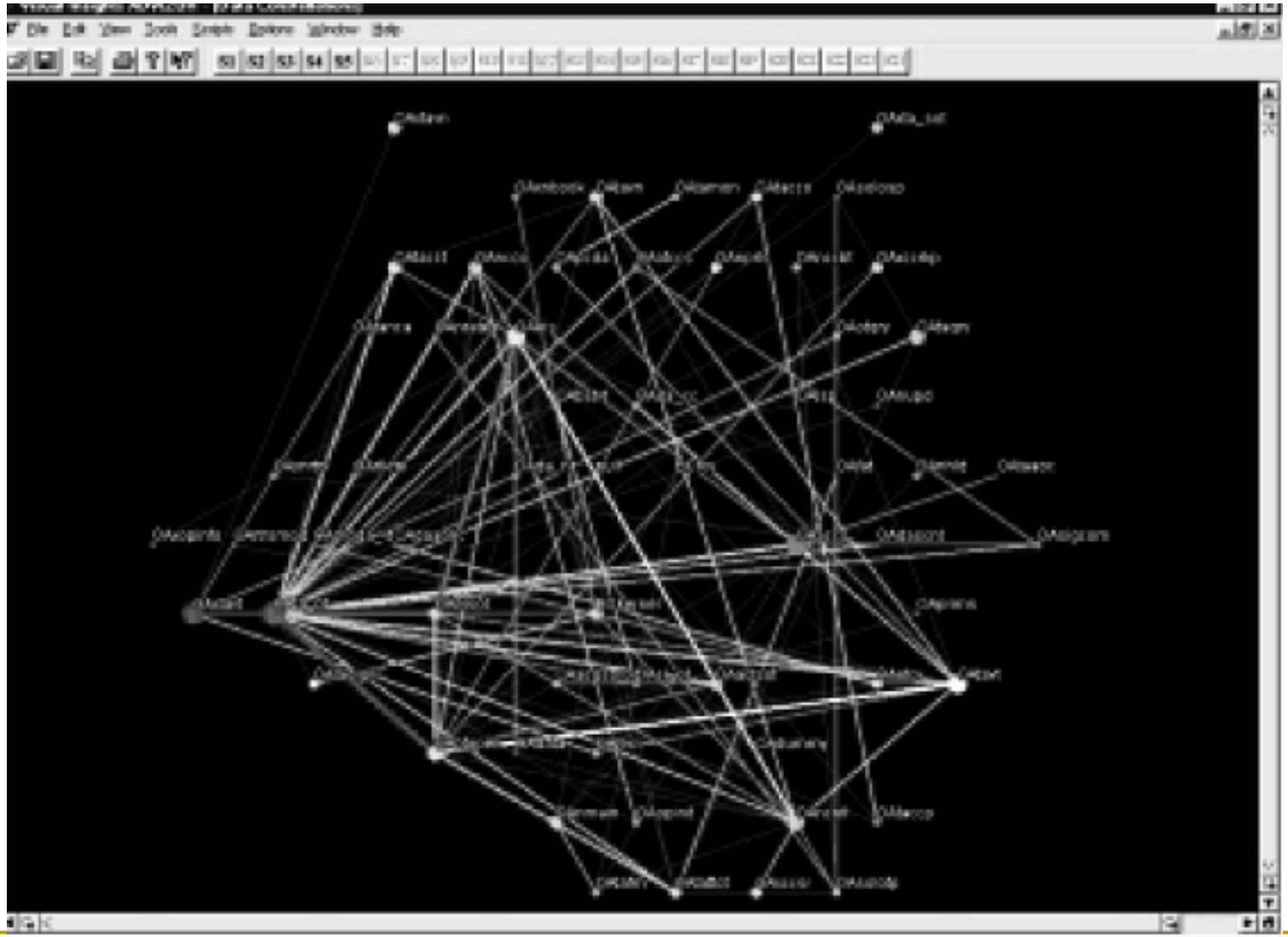


Fig. 11. Relationships between files based on IMR linkages. Nodes are files and linkage is defined by the relative frequency of files being changed as part of the same IMR. Link color shows the strength of this relationship, defined by (1). Discussed in Section 4.6.

Perspectives

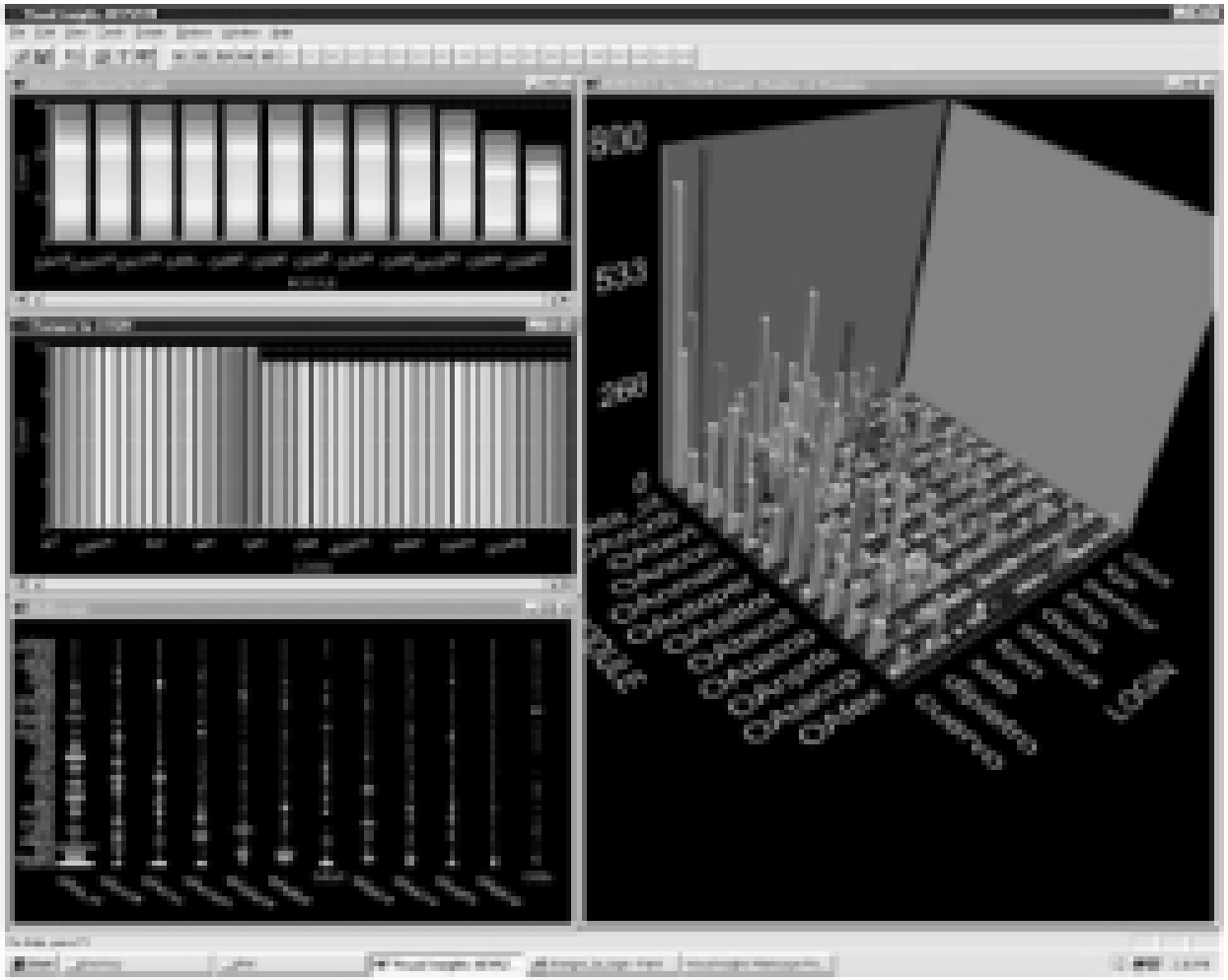


Fig. 10. Number of changes indexed by developer and module. *Bar charts*: changes by module (top) and by developer (bottom), with color encoding developer. *Matrix view* and *City scape view*: number of changes indexed by developer (columns) and module (rows). Discussed in Section 4.5.

Perspectives cont`d

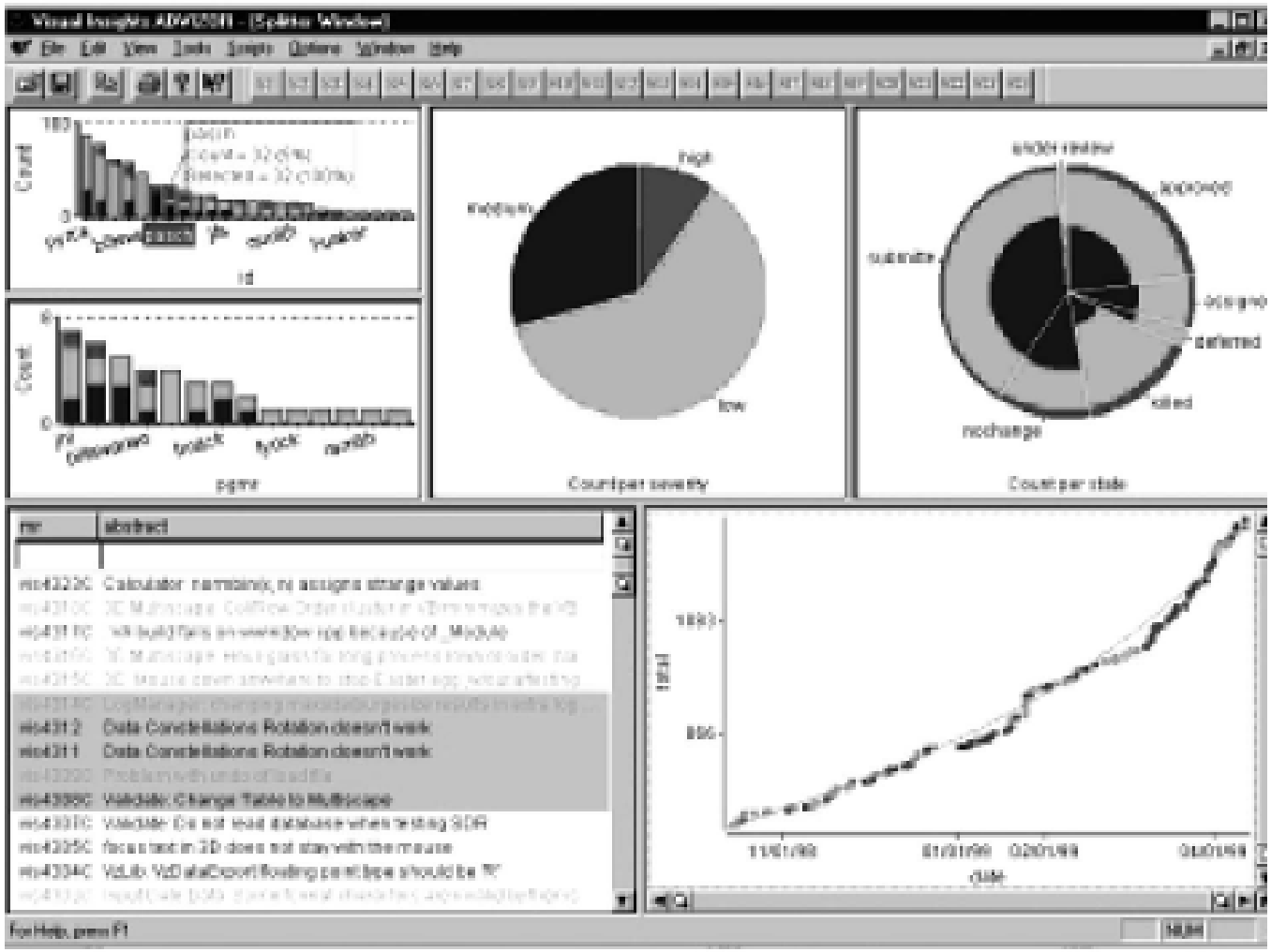


Fig. 12. Perspective showing MRs indexed in multiple ways. *Bar charts:* MRs indexed by creator (i.e., submitting programmer) (top) and assigned programmer (bottom). *Pie charts:* MRs indexed by severity (left) and status (right). *Data sheet:* MR abstract. *Scatterplot:* Cumulative number of MRs, indexed by time. Discussed in Section 5.1.

Changes

- Changes Indexed by time
- Changes Indexed by developer
- Size of changes Indexed by release
- Activity of developer in software space
- The span of changes

Likings

- ``A picture is worth a thousand words``
- Linking views and creating perspectives which are interactive
- Can be handy in practise – ADVIZOR™
- It is well organized and well written.

Disliking

- No color and obscure diagrams
- Had to rely too much on descriptions to decipher what the views mean
- ``Insightful rather than faithful``
- Could give more detail description of what management decision was taken based on the findings on these views and perspectives.