



## Software Visualization

November 9, 2005      Presented By: Shawn Minto

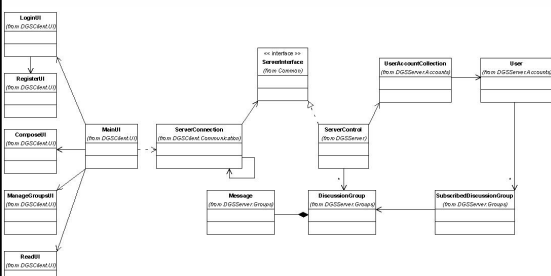


## Introduction


- Traditionally
  - Static software structure
  - Call graphs



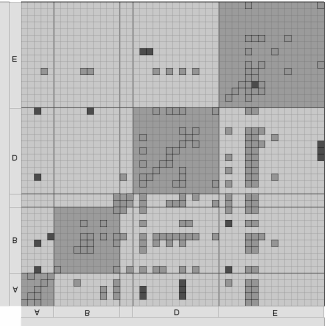
## Introduction




Created with Poseidon for UML, Community Edition. Not for Commercial Use.



## Introduction




From: Using Multilevel Call Matrices in Large Software Projects, Frank van Ham, Proc. InfoVis 2003




## Introduction

- Concerned more than structure
  - Application Behavior
  - Execution Patterns
  - **Testing data**
  - **Managing software projects**



## Encoding Program Test Information

- TARANTULA
  - Test suite visualization
    - Provides overview of test results
    - SeeSoft visualization of code
    - JUnit integrated into Eclipse
  - Potentially thousands of tests with text based output
    - Difficult to analyze as a suite




## Encoding Program Test Information

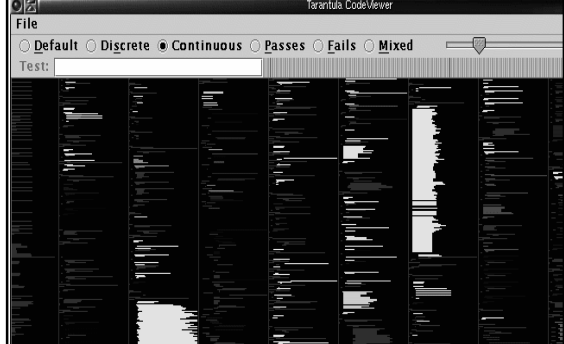

- Visualization input
  - Test case number
  - Whether the test passed or failed
  - Line numbers of covered code

```

1 P 1 2 3 12 13 14 15 ...
2 P 1 2 23 24 25 26 27 ...
3 F 1 2 3 4 5 123 124 125 ...
  
```




## Encoding Program Test Information


## Encoding Program Test Information

- Strengths
  - Elegant solution to a real problem in software development
  - Mini user study done to select coloring scheme
  - Many interaction techniques provided to user
    - Discrete mode
    - Test case subset
    - Lightness of unrelated lines of code




## Encoding Program Test Information

- Weaknesses
  - Not scalable
    - Would be unable to handle a real system
  - Unable to read source code due to color choice and size of code window
  - No mention of how to get the data to the preferred format (or any tool to give the coverage and pass data)
  - Bad description of how the coloring scheme works



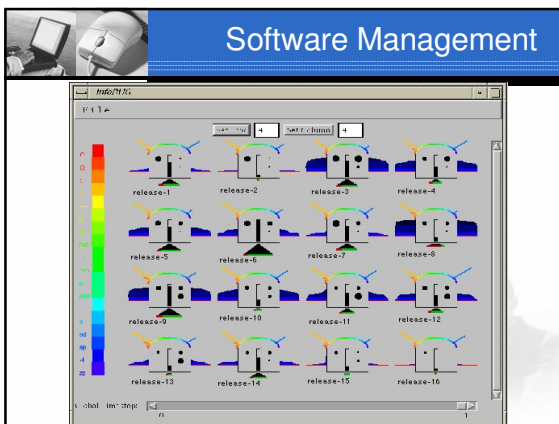
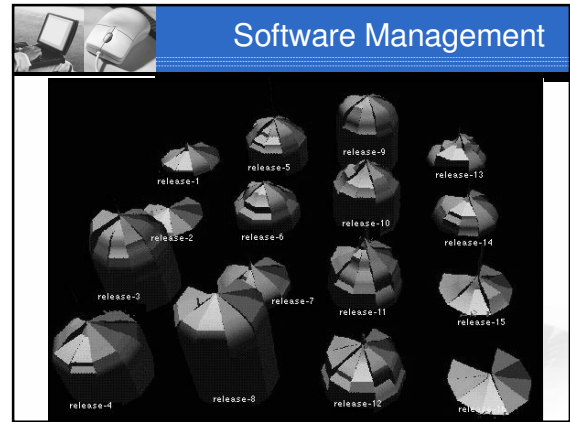
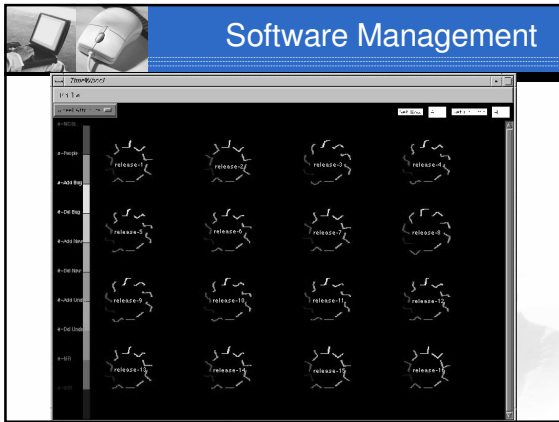
## Software Management

- Project Management
  - Many different types and quantities of resources to track and manage
  - Difficult to manage the amount of data collected with existing tools
    - Large volume of unstructured data
    - Diverse types
    - Resource = "real world" object
  - All data is time oriented



## Software Management

- Solution
  - Use a glyph to group the data and its properties
  - Shows the trends over time



- ### Software Management
- Strengths
    - Good overview of the evolution of a system
    - Potential to compare the evolution of different systems
    - Glyphs use preattentive patterns and reduce eye movement

- ### Software Management
- Weaknesses
    - No way to get detail (no interaction)
    - No mention of how to get this data
      - CVS I assume
    - InfoBug can be complicated
    - Different classes of resources available, but visualization only really dealt with code
      - How would the other types of data be managed and tracked?

- ### Conclusion
- Missing user studies
  - Information Visualization has always been used to show system structure
  - Different areas of software engineering being visualized to help users



## References

Visually Encoding Program Test Information to Find Faults in Software James Eagan and Mary Jen Harrold and James A. Jones and John Stasko Proc InfoVis 2001 pp. 33-36.

Managing Software with New Visual Representations, Mei C. Chuah, Stephen G. Eick, Proc. InfoVis 1997.