Dr. Jones: A Software Archaelogist's Magic Lens

Mark A. Foltz
Design Rationale Group
MIT Artificial Intelligence Lab
May 17, 2001

 mfoltz@ai.mit.edu

The Problem

"Add wish list"

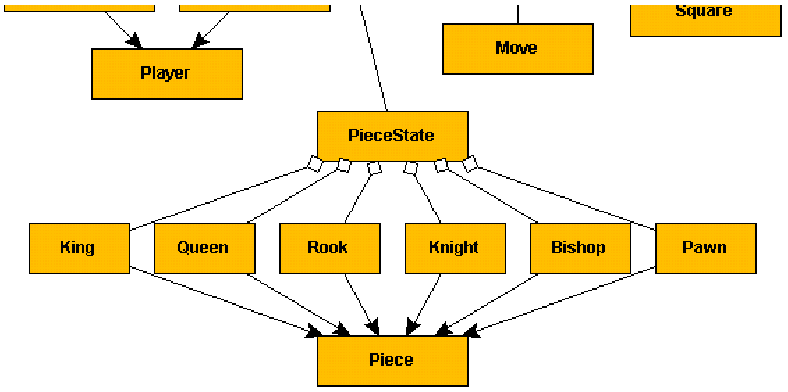
```
Buffers Files Tools Edit Search Mule Classes JDE Java Sena
// CourseVIMap.java --
// Author: Mark Foltz <mfoltz@ai.mit.edu>
// Maintainer: Mark Foltz <mfoltz@ai.mit.edu>
// Version:
// Created: <Mon Jun  1 19:36:45 1998>
// Keywords:


package is.course_vi;
import java.awt.*;
import java.awt.image.*;

public class CourseVIMap extends Map {

  public CourseVIMap(int w, int h, double small, double large
                     double size, Color bg, Component source) {
```

Sarah needs a mental model of the program



Programs are structured so tools can help

Thesis

Programmer's mental models of programs are organized around a small number of tightly coupled and hierarchical program views.

- **Structure (parts and relationships)**
- **Behavior (interactions)**
- **Tightly coupled (structure ←→ behavior)**
- **Hierarchical (library, package, class, method)**

Contributions

- **Software Engineering**
  - **Better tools for program comprehension**
- **Design Rationale**
  - **Reverse engineering**
  - **What information should we be capturing during design?**
- **Artificial Intelligence**
  - **Hierarchical recognition of design patterns in structure and behavior of software**

Why is source code hard to understand?

Source Code	Good Documentation
Modularized in small pieces	Holistic – how it all fits together
Linear	Hypertextual
Unclear starting point	Walkthroughs, examples
Small windows	Context and overviews

Approach

- **User Studies of Program Comprehension**
  - **What info is sought? (program concepts)**
  - **How is info extracted? (strategies)**
  - **Why? (software maintenance tasks)**

- **Dr. Jones, a software archaeologist's assistant**
  - **Visualization of structure and behavior**
  - **Incremental exploration**
  - **Design pattern recognition**

User Studies: Methods

Given an unfamiliar program, subjects …

- **Navigate source code as hypertext**
  - **Verbally describe movement through program**
  - **Explain program later**
- **Browse/edit source code to improve design**
  - **Screen capture of editing actions**
- **Still debugging methods….**

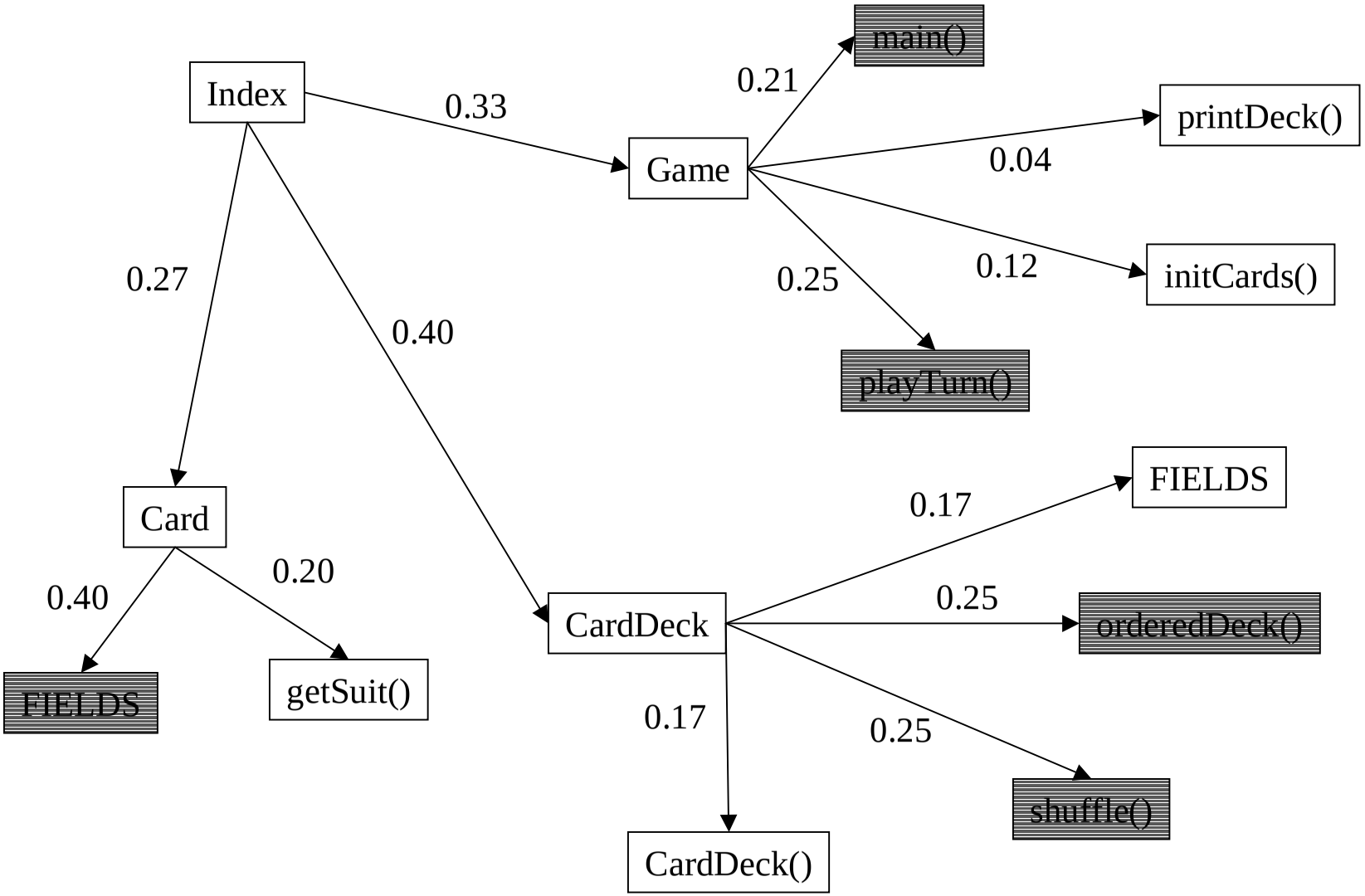
User Studies: Results

- **From prior studies:**
  - **Explaining from the UI backward**
  - **Draw UI, write pseudocode**
- **From recent studies:**
  - **Top-down vs. Bottom-up**

Page Type	Duration (secs)	Frequency
Class Index	6	15
Class	7	41
Methods	57	24
Fields	8	7

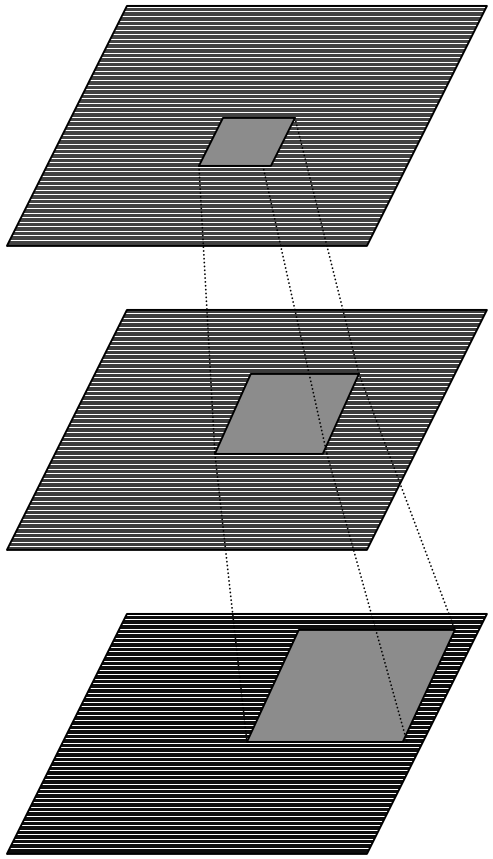
User Studies: Results

Markov transition matrix



Viewing structure

Multiscale browsing of abstractions



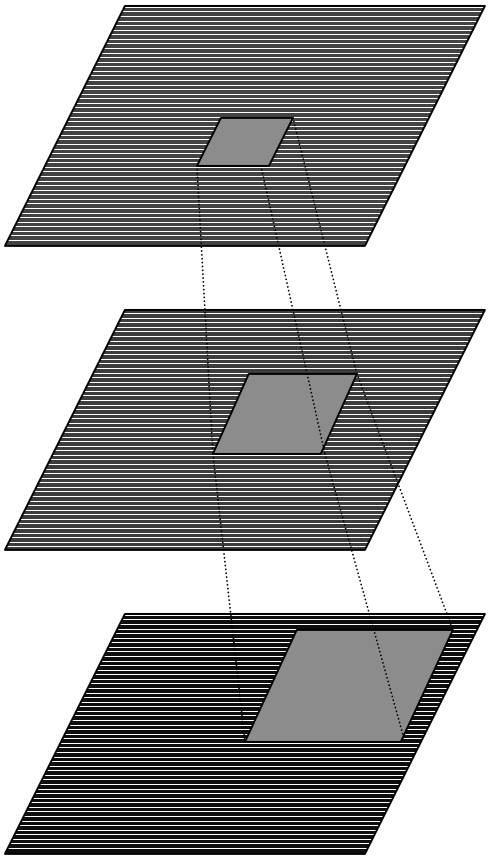
Application Semantics:
“Buy hockey tape”
(very hard to recognize)

Design Patterns:
Transactions
(hard to recognize)

Java Semantics:
Threads, Classes, Exceptions
(easy to recognize)

…all the way down to source code

Application-Specific Knowledge



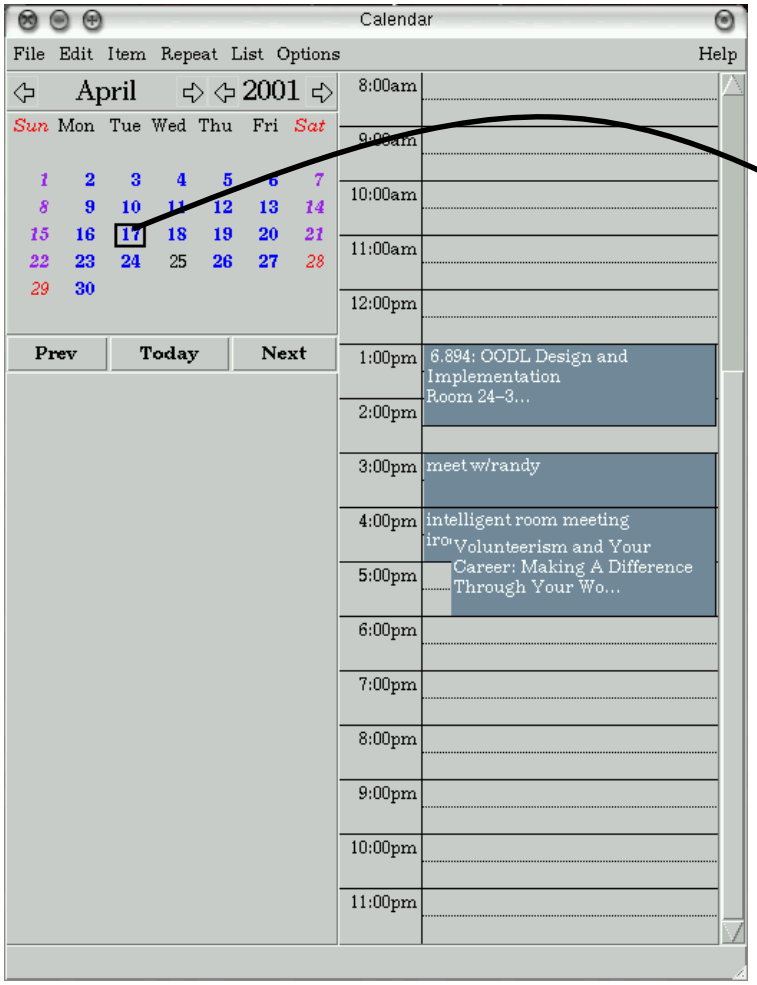
Application Semantics:
Pluggable Domain Models

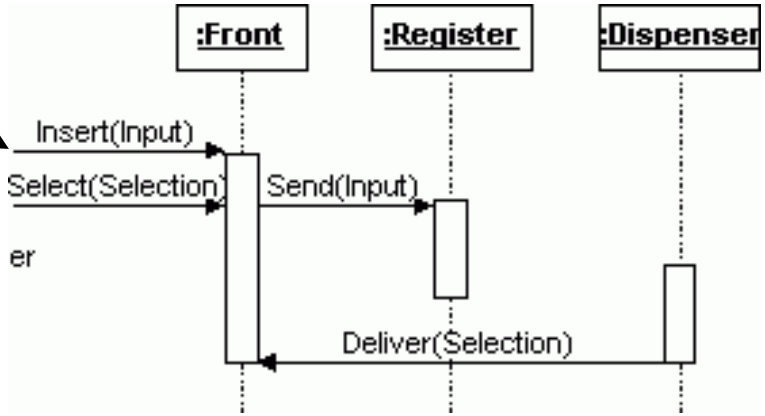
- Program as analogy
- A lot's in the names

"buy hockey tape" → **`BuyItem(theItem)`**

Viewing Behavior

Jiggling software



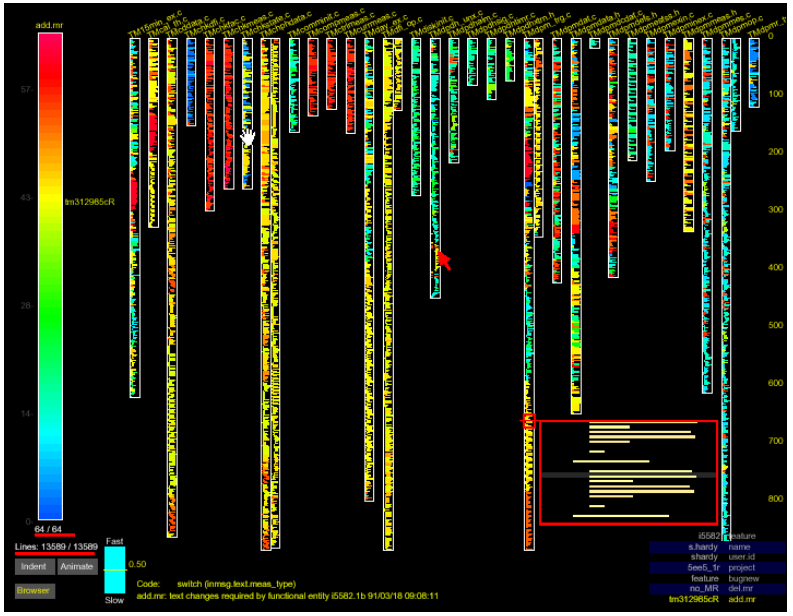


Evaluation

1. **Visualize simple programs**
   - **I.e. solitaire**
2. **Compare with Javadoc**
3. **See how it does on big programs**
   - **10k LOC**

Related Work

- **Cognitively motivated program understanding tools**
  - **Lethbridge 00, Storey 97**
- **Design pattern recognition in programs**
  - **Programmer's Apprentice**
- **Software visualization**
  - **SeeSoft (Eick 94)**



Conclusion

- **Source code is not "self-documenting"**
- **Figure out what programmers are looking for**
- **Abstract from what's in the code to provide that info**
- **Visualize it to see how it all fits together**