Program Comprehension

Idaho State University | Computer Science

Isaac Griffith

CS 4423 and CS 5523
Department of Computer Science
Idaho State University

ROAR

Idaho State University
Computer Science

Outcomes

After today's lecture you will be able to:

- Understand and describe the general idea of protocol analysis and why it is used
- Understand and describe the basic ideas behind software visualization as it is applied to program comprehension

ROAR

Protocol Analysis

CS 4423/5523

ROAR

Protocol Analysis

- Novice programmers can learn by observing how experienced programmers behave during program comprehension.
  - Similarly, researchers can learn by observing how both novice and experienced programmers behave during program comprehension.
- Ideally, we want to observe all aspects of programmers' behavior while they are trying to understand the code:
  - What is the programmer studying?
  - What is the programmer thinking when he sees something interesting?
  - What does the programmer do after he finds something interesting?
  - What is the rational thinking behind the programmer's action?
- **Protocol analysis**, studied in the field of psychological research, is a key concept used in finding answers to the above questions.

Protocol Analysis

- **Protocol analysis** is a methodology for eliciting verbal reports from participants (programmers in this case) about their thought sequences as a valid source of data on thinking.
- Protocol analysis is composed of two steps:
  - **Concurrent verbalization of a comprehension task:** This step produces textual data (aka protocol data) representing the thought sequence of a programmer as he performs the comprehension task by reading the code.
  - **Analysis of protocol data:** The protocol data is analyzed to understand the characteristics of the thinking performed by the programmer.

Idaho State University
Computer Science

Protocol Analysis

Concurrent verbalization of a comprehension task

- Programmers are asked to verbalize their thoughts while working on a specific task, and it is recorded on audio-visual systems.
  - Programmers are asked to “think aloud”: they say loudly everything they think, evaluate, and (mentally) move.
  - This is called **think aloud protocol** (TAP)
- Some examples of concurrent verbalization are:
  - I want to read the external documentations. What? No external documentations! I wanted to speak with the developers who designed and implemented the system, but they are all gone! I mean they have left the company.
  - Okay. I am reading the code prologue.
  - Now I know that the system is for enabling customers to make seat reservations in a restaurant and placing orders.
  - I find interesting keywords in the prologue: phone, cell phone, and laptops. I think one could make reservations by calling a restaurant or on the Web. I guess … customers might be able to place orders from their cell phones.
  - Let me read the module called MakeReservation.
  -

Analysis of protocol data

- There is no common, detailed procedure to analyze protocol data.
- Rather, a very general description of protocol analysis is as follows:
  - Divide protocol data into several segments, say, speech sentences.
  - Assign the segments to different predefined categories. This is called **encoding**.
    - Coding categories are selected with a model of the verbalization process in mind.
    - Coding system of Ericsson and Simon: There are four kinds of segments: **intentions**, **cognitions**, **planning**, and **evaluations**.
  - Analyze the categorized protocol data to build a comprehension model of the programmer.
    - A comprehension model can be represented as a transition net, which resembles finite-state machines, where computations (represented with cognitions and intentions) are associated with states, and planning and evaluations are associated with transitions

Software Visualization

CS 4423/5523

ROAR

Visualization for Comprehension

- Visualization is supported with tools for program comprehension.
  - PUNS (Program Understanding Support environment)
  - PAT (Program Analysis Tool)
  - Fisheye view
  - UML (Unified Modeling Language)
  - City metaphor

Idaho State University
Computer Science

Visualization for Comprehension

- PUNS (Program Understanding Support environment)
  - Developed at IBM to provide **multiple views** of a program.
    - A Call graph for a set of procedures
    - A Control flow graph for an individual procedure
    - A graph showing the relationship between a file and a procedure that uses it
    - A data flow graph
    - A definition-use chain for a variable
  - By performing static analysis of the code, the tool detects low-level relationships and organizes them in a user-friendly environment so that the user can easily navigate through the graphs while switching between low-level and high-level objects.

Program Analysis Tool

- Presents a heuristic-based concept recognition mechanism to extract high-level functional concepts from source code.
- Assists programmers answer the following questions:
  - What high-level concepts does the program implement?
  - How are the high-level concepts coded in terms of low-level details?
- Explicitly represents two types of knowledge:
  - Program knowledge
    - Represented by programming concepts found in the code.
  - Analysis knowledge
    - Represented by information contained in program plans.
- Manages two databases to manipulate the two types of knowledge:
  - A data base of coding heuristics, data structure definitions, and functional coding pattern.
  - A data base of rules for program plans covering value accumulation, counting, sequential search of ordered and unordered structures, different types of searching, and sorting.

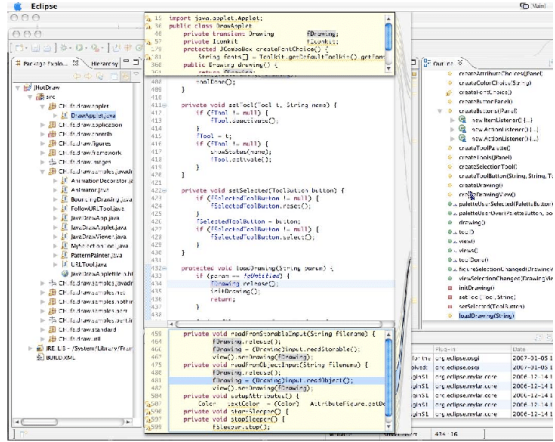
Fisheye View

- It supports programmer's navigation and comprehension.
- A fisheye view displays those parts of the source code that have the highest degree of interest related to the current focus of the programmer.
- Shows both **overview** and **details**.
  - An overview of the entire document is displayed to the right of the detailed view window.
  - The overview displays the source code reduced in size to fit the entire document within the space of the overview area.
  - The portion of the code shown in the detail area is visually connected with its location in the overview.

Idaho State University
Computer Science

Fisheye View

- The fisheye interface of Jakobsen and Hornbaek possess the following features:
  - Focus and context area
  - Degree of interface function
  - Magnification function
  - User interaction



Jakobsen and Hornbaek, “Transient Visualizations”, 2007

ROAR

UML

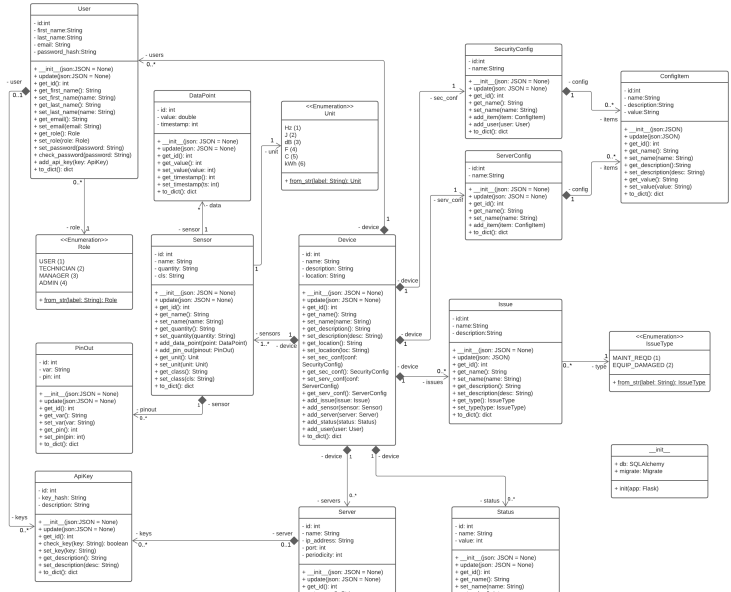
- **Class diagrams** can aid programmers in code comprehension.
  - The concepts of **perceptual organization** and **perceptual segregation** can be applied to organize UML class diagrams.
- Perceptual organization indicates when entities are organized in near proximity.
- Perceptual segregation indicates when entities are separated.
- The followings are some important principles of perceptual organization:
  - Good figure
  - Similarity
  - Proximity
  - Familiarity (Meaningfulness)
  - Element connectedness

UML

- Perceptual segregation is further explained as follows:
  - When one looks at the environment, what is seen is a whole picture – and not separate parts.
  - The following factors make an entity more like a figure that can be easily recognized.
    - Symmetry
    - Orientation
    - Contours

Idaho State University
Computer Science

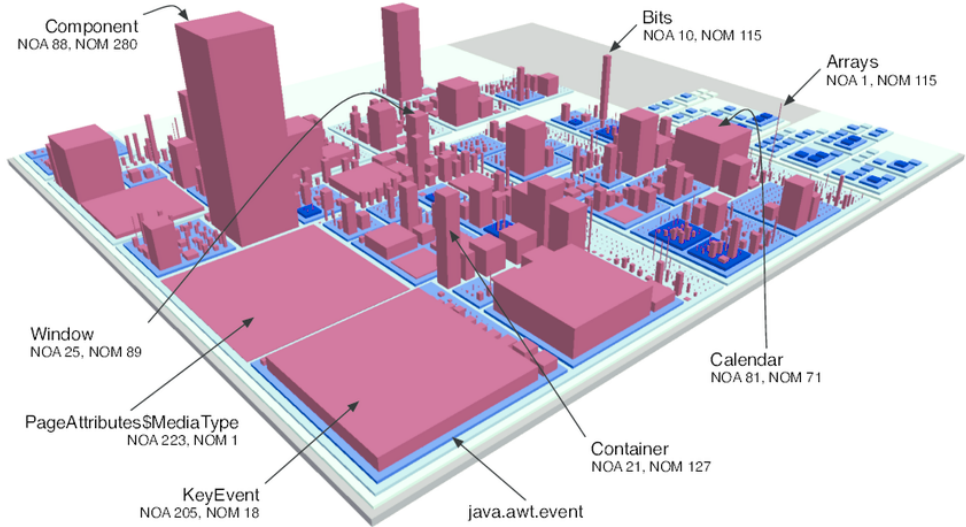
UML



City Metaphor (Wettel and Lanza)

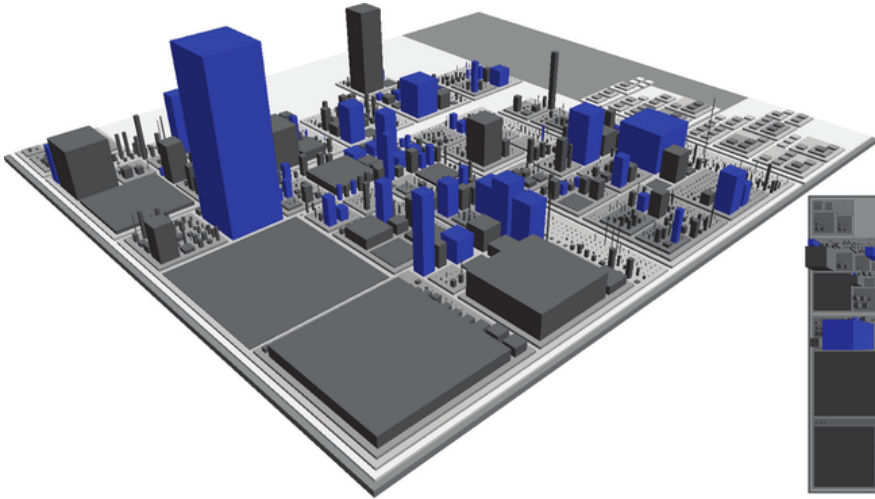
- This is a 3-dimensional visualization concept.
- Classes are represented as buildings located in city districts which in turn represent packages.
- The concept of habitability is at the core of the city metaphor, and the corresponding programming concept is familiarity.
  - The more familiar a programmer is with the code, the easier it is to understand the code.
- The concept of locality is supported by providing a navigable environment.)

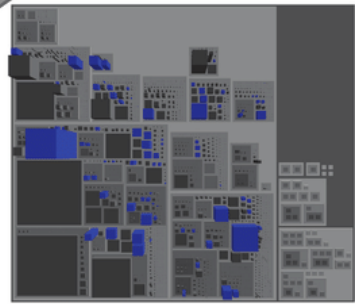
City Metaphor



Wettle and Lanza, "Visually localizing design problems with disharmony maps", SOFTVIS 2008, 2008

City Metaphor

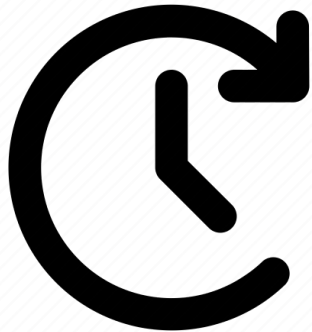




Wettle and Lanza, “Visually localizing design problems with disharmony maps”, SOFTVIS 2008, 2008

For Next Time

- Review EVO Chapter 8.4 - 8.6
- Read EVO Chapter 9.1 - 9.2
- Watch Lecture 24



Are there any questions?