Lecture 9 – Software visualization

- Software visualization
- Sub-fields
- Evolution
- Examples
- Current researches

Lecture 8 – Application of machine learning in software testing

- Software testing
- Role of machine learning in software testing
- Artificial neural network
- Use of ANN in software testing
Software visualization

a discipline that makes use of various forms of imagery to provide insight and understanding and to reduce complexity of the existing software system under consideration

Investigates approaches and techniques for static and dynamic graphical representations of algorithms, programs (code), and processed data.

The goal is to improve our understanding of inherently invisible and intangible software, particularly when dealing with large information spaces that characterize domains like software maintenance, reverse engineering, and collaborative development.

The main challenge is to find effective mappings from different software aspects to graphical representations using visual metaphors.
Software visualization

has been applied in various areas like algorithm animation, software engineering, concurrent program execution, static and dynamic visualizations of object-oriented code, fault diagnostics, debugging, requirements analysis.

properties of a software visualization system:

- Scope and content: What is the aspect of the program being visualized?
- Abstraction: What kind of information is conveyed by the visualization?
- Form and technique: How is the graphical information being conveyed?
- Method: How is the visualization specified?
- Interaction: How can the user interact with the visualization?
- Level of automation

Task-oriented view:
- Tasks: Why is the visualization needed?
- Audience: Who will use the visualization?
- Target: What is the data source to represent?
- Representation: How should it be represented?
- Medium: Where should the visualization be represented?

Software visualization sub fields:

- PROGRAM VISUALIZATION: actual program code or data structures in either static or dynamic form
- ALGORITHM VISUALIZATION: showing a abstract representation of an algorithm. Visualizations usually show the data and the effect on that data as the algorithm runs.
- ALGORITHM ANIMATION: presents the running of an algorithm as a movie where the visual representation of objects of the program smoothly change their location and appearance
- TEST DATA VISUALISATION: present test function, test data, coverage
- PROGRAM AURALIZATION: the use of sound to assist in the formation of mental images of the behavior, structure and function of a program or algorithm
- VISUAL PROGRAMMING: is a type of programming, which uses graphical objects to build software
Software Visualization Evolution

**2D visualization**

Two-dimensional SV techniques typically involve graph or treelike representations consisting of a large number of nodes and arcs

present pieces of the graph in different views or different windows so that the user can focus on the level of detail he desires. The software system is therefore represented in multiple windows that present to the observer different characteristics of the system under consideration

**3D visualization**

visualizing software in 2D does introduce a cognitive overload by presenting too much information

software objects are mapped to visual 3D objects

experiments analyzed perception in 2D and 3D and conclude that there is encouraging empirical evidence that error rates in perception are less in 3D visualization. One major advantage of 3D visualization is that it allows a user to perceive the depth of a presented structure

3D visualization for: metrics-based visualization of object-oriented programs and visualization to track software errors, isolate problems, monitor progress of development, Three-dimensional UML (Unified Modeling Language) representations

**Virtual environments**

Virtual environments (VEs) open possibilities of “immersion” and “navigation” that may help to better explore software structure.

VEs enable the user to interact with a representation of something familiar, namely a world with familiar objects that he/she can interact with

allow a user to concentrate on one aspect of the world in detail while providing a distant view of other aspects that are situated farther away. As the user moves close to each entity or visual component, it comes to “life” or presents a higher level of detail. (elision - abstracts distant objects and details closer objects)

**Distributed Virtual environments**

multiple users interact with each other in real time, even though those users may be physically located around the world

can be used for collaborative SV-based applications dealing with large and distributed software projects including coding, maintenance, and interactive visualization
Examples of software visualization
Examples of software visualization

Fig. 1. Visualization of a software system

Figure 3: TARANTULA’s “Continuous” view using both hue and brightness changes to encode more details of the test cases executions throughout the system.
Software visualization - Recent papers

IEEE Working Conference on Software Visualization

Session 1: Visualization Techniques

Combining Tiled and Textual Views of Code by Michael Homer, James Noble
Integrating Anomaly Diagnosis Techniques into Spreadsheet Environments by Daniel Kulesz, Jonas Scheurich, Fabian Beck
Action-Based Visualization by Antti Jääskeläinen, Hannu-Matti Järvinen, Heikki Virtanen
Slicing-Based Techniques for Visualizing Large Metamodels by Arnaud Blouin, Naouel Moha, Benoit Baudry, Houari Sahraoui

Session 2: Visualization Techniques, Paradigms, and Languages

Search Space Pruning Constraints Visualization by Blake Haugen, Jakub Kurzak
Livecoding the SynthKit: Little Bits as an Embodied Programming Language by James Noble
A Domain-Specific Language for Visualizing Software Dependencies as a Graph by Alexandre Bergel, Sergio Maass, Stéphane Ducasse, Tudor Girba
Feature Relations Graphs: A Visualisation Paradigm for Feature Constraints in Software Product Lines by Jabier Martinez, Tewfik Ziadi, Raul Mazo, Tegawendé F. Bissyandé, Jacques Klein, Yves Le Traon
Validation of Software Visualization Tools: A Systematic Mapping Study by Abderrahmane Seriai, Omar Benomar, Benjamin Cerat, Houari Sahraoui
Using a Task-Oriented Framework to Characterize Visualization Approaches by Marcelo Schots, Claudia Werner

Session 3: Formal Tool Demos

Visual Clone Analysis with SolidSDD by Lucian Voinea, Alexandru C. Telea
Polyptychon: A Hierarchically-Constrained Classified Dependencies Visualization by Donny T. Daniel, Egon Wuchner, Konstantin Sokolov, Michael Stal, Peter Liggesmeyer

Session 4: Compilers, Control Flow, and Debugging

How Developers Visualize Compiler Messages: A Foundational Approach to Notification Construction by Titus Barik, Kevin Lubick, Samuel Christie, Emerson Murphy-Hill
Lightweight Structured Visualization of Assembler Control Flow Based on Regular Expressions by Sibel Toprak, Arne Wichmann, Sibylle Schupp
Templated Visualization of Object State with Vebugger by Daniel Rozenberg, Ivan Beschastnikh
The Challenge of Helping the Programmer during Debugging
by Steven P. Reiss

Session 5: Evolution

ChronoTwigger: A Visual Analytics Tool for Understanding Source and Test Co-evolution
by Barrett Ens, Daniel Rea, Roiy Shpaner, Hadi Hemmati, James E. Young, Pourang Irani

Visualizing the Evolution of Systems and Their Library Dependencies
by Raula Gaikovina Kula, Coen De_Roover, Daniel German, Takashi Ishio, Katsuro Inoue

AniMatrix: A Matrix-Based Visualization of Software Evolution
by Sébastien Rufiange, Guy Melançon

Session 6: Developers and Teams

Visualizing Developer Interactions
by Roberto Minelli, Andrea Mocci, Michele Lanza, Lorenzo Baracchi

Information Visualization for Agile Software Development
by Julia Paredes, Craig Anslow, Frank Maurer

FAVe: Visualizing User Feedback for Software Evolution
by Emitza Guzman, Padma Bhuvanagiri, Bernd Bruegge
Validation of software visualization tools

Validation of Software Visualization Tools: A Systematic Mapping Study
by Abderrahmane Seriai, Omar Benomar, Benjamin Cerat, Houari Sahraoui

consider 752 articles from multiple sources, published between 2000 and 2012, and study the validation techniques of the software visualization articles

Research questions:
- How mature is software visualization research with respect to validation?
- What validation techniques have been used for the assessment of visualization research?

cover the period spanning from 2000 to 2012 and query two publication databases:
- IEEE Xplore (ieeexplore.ieee.org)
- Scirus (www.scirus.com)
Software Visualization articles

Investigation Types

• **Experiment**: it is possible to control the studied phenomenon, to manipulate the factors which can influence it, to validate or invalidate a hypothesis. The studied hypotheses consist of testing particular cause and effect relationships (e.g., the proposed visualization tool helps decrease the comprehension effort). As usual, the studies try to reject a null hypothesis (e.g., there is no difference in comprehension effort using the proposed visualization tool).

• **Case study**: studying a given case (product, project, organization) to collect detailed information or to make an exploration. Its outcome often serves as preliminary information of an experiment.

• **Survey**: It is a retrospective study of a situation
Validation of software visualization tools

<table>
<thead>
<tr>
<th>Property</th>
<th>Class</th>
<th>Number of articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Investigation Type</td>
<td>Experiment</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Case Study</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Survey</td>
<td>5</td>
</tr>
<tr>
<td>Task</td>
<td>Specific</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>Exploratory</td>
<td>24</td>
</tr>
<tr>
<td>Data Source</td>
<td>Open Source</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Industrial</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Home Data</td>
<td>7</td>
</tr>
<tr>
<td>Subjects</td>
<td>None</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>Students</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Practitioners</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Both</td>
<td>8</td>
</tr>
<tr>
<td>Objective Measures</td>
<td>Yes</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>34</td>
</tr>
<tr>
<td>Comparison</td>
<td>None</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Visualization</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Domain</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Both</td>
<td>4</td>
</tr>
</tbody>
</table>

Fig. 8: Ratio of validated articles per year.