Software Visualization and Model Generation

Erik Doernenburg  
Software Developer  
ThoughtWorks, Inc.

Gregor Hohpe  
Software Engineer  
Google, Inc.

Where are the most defects?
Which way do the messages flow?

A picture says more than 1000 words

- The amount of information in current systems is beyond what we can handle (understand?)
- Often we are only interested in a specific angle
  - Relationship between classes – not the entire source
  - Number of messages flowing – not the message content
- We're good at spotting patterns in images...

Being able to control large-scale systems is an illusion. But we can observe what is happening…
Where do we get the picture from?

- Models created upfront convey a vision but usually don’t reflect reality
- Generating a complete model for large systems is nearly impossible
- Systems evolve locally, often uncontrolled
  - Particularly true for loosely coupled, dynamic systems (SOA)
- The best picture very much depends on the question you are trying to answer
  - We need tools that make it easy to create ad-hoc models

Visualizing Software

- source code
- running system
- raw data
- model
- diagram

Scanner/Parser

Instrumentation

Mapper

Renderer

Nodes: A, B
Edges: X(A→B)
Five Steps

1. Select Meta-model
2. Inspection / Instrumentation
3. Mapping to Model
4. Visualization / Output
5. Validation / Analysis

1. Select Meta-model

- “A model that describes a model”
  - The elements a model can be composed of
  - How to combine these elements

- Example: meta-model for a class diagram
  - A class is a box with name, methods, fields,…
  - Available connectors: association, inheritance, aggregation,…
  - Rules: no circles in inheritance etc.

- Sounds more scientific than it really is
- Usually pick from a few popular candidates
Common Metamodels

Metrics (Quantitative)

Directed Graph

Tree

Process Model (e.g. Petri Net)

2. Inspection / Instrumentation

Static Analysis
- Inspect System Design
  - Source code
  - Configuration repository
  - Scan / Parse into intermediate format

Dynamic Analysis
- Inspect Running System
  - Profiling
  - Listen to messages
  - Log files
  - Network sniffer
  - Compiler decorator
3. Mapping to Model

- Map the gathered data onto the meta-model

- Example: Messaging System
  - Capture send / receive actions
  - Map onto directed graph

<table>
<thead>
<tr>
<th></th>
<th>Channel</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Senders</td>
<td>A</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Y</td>
</tr>
<tr>
<td>Receivers</td>
<td>C</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Y</td>
</tr>
</tbody>
</table>

Graph Model

4. Visualization / Output

- Example: GraphViz Dot
  - Automated graph layout tool
  - Takes textual input, produces many graphics formats
  - Developed by AT&T, Common Public License

- Example: Codecrawler
  - Object-Oriented reverse engineering
  - Polymetric views of metrics
  - Hotspots, complexity, inheritance, data storage, etc.
  - All views are interactive
5. Validation / Analysis

- A model is useful for more than pretty pictures
- Can apply rules against the model
- Example: Directed Graph

Identify circles
Identify islands (in domain models)
Identify leaf nodes (in dependency graphs)

Example: Object Dependencies

- Static analysis: Spring bean configuration
- SpringViz, a small XSLT sheet, maps bean configuration to input for GraphViz Dot
- Mapping and format hard-coded in style sheet
- Really simple but really useful
Example: Component Dependencies

- Dynamic Analysis: JAR references
- Decorate compiler with custom Ant task to get dependencies on a JAR level
- Map to a Directed Graph
- Render with GraphViz Dot as clickable SVG
- Navigate through model

Example: Code Crawler, System Complexity

- Static analysis: Source code analysis
- Code Crawler imports XMI and calculates metrics
  - NOA, NOM, WLOC
- Renders polymetric System Complexity view
  - Width, height, color used for metrics
  - Position used for tree layout of inheritance
- Goal of this view is to classify inheritance hierarchies
  - Subsystems
  - Large stand-alone classes
- Can use other views to understand inner workings of specific hierarchies
Example: Asynchronous Messaging

Processes communicate by sending messages across channels

Visualization

- **Dynamic Analysis:** Instrument Message Sender and Receiver
- Collect publication and subscription data centrally
- Map to a Directed Graph model
- Render with GraphViz Dot
- Re-render as model changes
Message Flow Graph

Example: Process Mining

- A system performs a series of activities for each case (process instance)
- Goal: create a process model from running system
- Approach: Analyze activity logs and create process model
- Meta-model: Petri Net
- Variety of algorithms, e.g. $\alpha$-Algorithm (Wil v.d. Aalst)
Process Mining Example

<table>
<thead>
<tr>
<th>Case</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
</tr>
<tr>
<td>5</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
</tr>
</tbody>
</table>

Log File

Mapper

Process Model

Process-Aware information Systems, Dumas et al.

What’s Next?

- Applies to many levels
  - Single module source code
  - Multiple modules
  - Whole systems

- Diagrams hard to scale to huge systems
  - Interactive zoom / drill-down
  - Annotation-aided visualization

- As systems become larger and more dynamic tools like this become a necessity
Resources

- **Tools**
  - [http://www.graphviz.org](http://www.graphviz.org) (Dot)
  - [http://www.eaipatterns.com](http://www.eaipatterns.com) (Messaging visualization)

  - [www.processmining.org](http://www.processmining.org) (Process Mining Tool)

- Michele Lanza’s work (CodeCrawler)
  - [http://www.inf.unisi.ch/faculty/lanza](http://www.inf.unisi.ch/faculty/lanza)

Questions

?