Programming Interfaces
Today

Programming languages:
Visual programming formalisms & languages

Better (?) code editors
Structured editors, spatializing code
System vs. scripting languages
Debugging Interfaces

End-user Programming with/for the Web

Evaluating Languages & APIs
Visual Programming
State diagrams

\[
a^*ba^+ \]

\[
\begin{align*}
S1 & \xrightarrow{b} S2 \\
S1 & \xrightarrow{a} S3 \\
S3 & \xrightarrow{b} S4 \\
S4 & \xrightarrow{a,b} S4
\end{align*}
\]
Statecharts = state diagrams + depth + orthogonality + broadcast communication.
Flowchart
Structogram

DO I = 1 TO N

DO J = 1 TO N

SUM = 0

DO K = 1 TO N

SUM = SUM + A(I,K) * B(K,J)

C(I,J) = SUM
Data Flow Diagram

Order Details

Enter Order

Orders

Process Order

Response

Invalid Order
UML

Diagram

Structure Diagram
- Class Diagram
- Composite Structure Diagram
- Deployment Diagram
- Package Diagram

Behavior Diagram
- Activity Diagram
- Use Case Diagram
- Interaction Diagram
- State Machine Diagram

Notation: UML

Better Code Editors & Debuggers
When flag clicked:

- Move 10 steps
- Turn 15 degrees
- Turn 15 degrees
- Point in direction 90
- Point towards
- Go to x: -148 y: -41
- Glide 1 secs to x: -148 y: -41
- Change x by 10
- Set x to 0
- Change y by 10
- Set y to 0
- If on edge, bounce

- Say Knock knock.... for 2 secs
- Wait 2 secs
- Say Banana for 2 secs
- Wait 2 secs
- Say Knock Knock.... for 2 secs
- Wait 2 secs
- Say Bananaaaaaaa for 2 secs
- Wait 2 secs
- Say Knock! Knock! for 2 secs
- Wait 2 secs
- Say Orange for 2 secs
- Wait 2 secs
- Say Orange you glad I didn’t say banana!!!! for 3 secs
- Wait 2 secs
- Play drum 49 for 0.25 beats
Structured Editors: Alice

http://www.grandtextauto.org/archives/alice_screenshot.jpg
Keyword Programming

```java
public List<String> getLines(BufferedReader in) throws Exception {
    List<String> lines = new Vector<String>();
    while (in.ready()) {
        lines.add(in.readLine());
    }
    return lines;
}
```
Whyline

Debugging is about generating hypotheses about program behavior and testing those hypotheses.

The WhyLine enables users to explore these questions.

http://www.cs.cmu.edu/~NatProg/whyline-java.html
Systems vs Scripting Languages

Ousterhout, IEEE Computer 1998
http://home.pacbell.net/ouster/scripting.html

Degree of Typing

- None
- Strong

Instructions/Statement

- Assembly
- Tcl/Perl
- Visual Basic
- C
- C++
- Java

CS260 - UC Berkeley Fall 2011
Programming (with) the Web
CoScripter

Demonstrations create pseudo-natural language instructions.

Instructions can be modified and shared.

Personal variables maintain data privacy.

Complete a purchase on BOND

- go to "https://w3-1.ibm.com/procurement/
- you click the cart you want to check out
- click the "Check out/finish" button
- click the "Edit accounting one screen (advanced)" button
- enter your bluepages.dept (e.g. NWEB) into the "Department" textbox
- enter your Bond Major (e.g. 700) into the "Major" textbox
Marmite
Wong & Hong, CHI 2007
Corresponding Flickr API calls

- `flickr.photos.getInfo(photo_id = "298655528").title`
  Return the current photo's title.

- `info = flickr.photos.getInfo(photo_id = "298655528")`
  URL = “http://farm”
  + info.farm-id
  + “.static.flickr.com/”
  + info.server-id
  + “/”
  + info.attributed[“id”]
  + “_”
  + info.secret
  + “.jpg”
  Return the static URL for this image.

- `flickr.tags.getListPhoto(photo_id = "298655528")`
  Get the tag list for a given photo.
  `flickr.photos.search(tags = “poignant ...”)`
  Return a list of photos for this user, matching the given tags.
**Original page**

**Proxy Server**

**Rewritten page with API annotations**

---

**Site-to-Service Map**

(written in the d.mix wiki)
Sample 'Images klemmer tagged with climbing'
welcome to the wiki studio. go home. list all pages

climbersportal

```ruby
<hr><h3>Flickr photos from klemmer matching tag climbersportal</h3>
<div class="rubyCode" id="element2007" user_id="78437058"=
Wiki.import :flickr_api; doc = Hpricot(selfcode);
html = FlickrAPI.photos.search(:user_id => div['user_id'],
FlickrAPI.img(p,:width=>200) #output a HTML img
end
```
Evaluating languages & APIs
Programmer Personas

Mort: the opportunistic developer
Elvis: the pragmatic programmer
Einstein: the paranoid programmer

# Cognitive Dimensions of Notation

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>What are the types and availability of abstraction mechanisms?</td>
</tr>
<tr>
<td>Hidden Dependencies</td>
<td>Is every dependency overtly indicated in both directions?</td>
</tr>
<tr>
<td>Premature Commitment</td>
<td>Do programmers have to make decisions before they have the information they need?</td>
</tr>
<tr>
<td>Secondary Notation</td>
<td>Can programmers use layout, color, or other cues to convey extra meaning, above and beyond the ‘official’ semantics of the language?</td>
</tr>
<tr>
<td>Viscosity</td>
<td>How much effort is required to perform a single change?</td>
</tr>
<tr>
<td>Visibility</td>
<td>Is every part of the code simultaneously visible, or is it at least possible to juxtapose any two parts side-by-side at will?</td>
</tr>
<tr>
<td>Closeness of Mapping</td>
<td>Closeness of visual representation to problem domain. What ‘programming games’ need to be learned?</td>
</tr>
<tr>
<td>Consistency</td>
<td>When some of the language has been learnt, how much of the rest can be inferred?</td>
</tr>
<tr>
<td>Diffuseness</td>
<td>How many symbols or graphic entities are required to express a meaning?</td>
</tr>
</tbody>
</table>
Cognitive Dimensions of Notation

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffuseness</td>
<td>How many symbols or graphic entities are required to express a meaning?</td>
</tr>
<tr>
<td>Error-proneness</td>
<td>Does the design of the notation induce ‘careless mistakes’?</td>
</tr>
<tr>
<td>Hard mental operations</td>
<td>Are there places where the user needs to resort to fingers or penciled</td>
</tr>
<tr>
<td></td>
<td>annotation to keep track of what’s happening?</td>
</tr>
<tr>
<td>Progressive evaluation</td>
<td>Can a partially-complete program be executed to obtain feedback?</td>
</tr>
<tr>
<td>Role-expressiveness</td>
<td>Can the reader see how each component of a program relates to the whole?</td>
</tr>
</tbody>
</table>