## Overview

### Analysis of WIMP applications

**Power vs. Simplicity**

**Interaction model**

**Instrumental Interaction**

**Design Principles**

### Number of commands

<table>
<thead>
<tr>
<th>Criteria</th>
<th>W6</th>
<th>E5</th>
<th>Pe3</th>
<th>P4</th>
<th>X3</th>
<th>C3</th>
<th>Avg</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>#menus</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7.7</td>
<td>0.5</td>
</tr>
<tr>
<td>#cmds</td>
<td>106</td>
<td>84</td>
<td>97</td>
<td>111</td>
<td>99</td>
<td>74</td>
<td>95.2</td>
<td>13.8</td>
</tr>
<tr>
<td>#dlogs</td>
<td>69</td>
<td>44</td>
<td>20</td>
<td>27</td>
<td>40</td>
<td>21</td>
<td>36.8</td>
<td>18.6</td>
</tr>
<tr>
<td>#smenu</td>
<td>1</td>
<td>15</td>
<td>27</td>
<td>26</td>
<td>13</td>
<td>22</td>
<td>17.3</td>
<td>9.8</td>
</tr>
<tr>
<td>#scmds</td>
<td>3</td>
<td>58</td>
<td>73</td>
<td>82</td>
<td>65</td>
<td>121</td>
<td>74</td>
<td>95</td>
</tr>
<tr>
<td>#sdlog</td>
<td>0</td>
<td>20</td>
<td>20</td>
<td>40</td>
<td>10</td>
<td>28</td>
<td>19.7</td>
<td>13.9</td>
</tr>
</tbody>
</table>

| Tcmds     | 108| 127| 143 | 167| 151| 173| 144.8| 24.5|
| Tdlogs    | 69 | 64 | 40  | 67 | 50 | 49 | 56.5 | 11.8|
| Cmds/M    | 13.3| 10.5| 13.9| 13.9| 14.1| 9.3 | 12.5 | 2.1 |
| Cmds/SM   | 3.0 | 3.9 | 2.7 | 3.2 | 5.0 | 5.5 | 3.9  | 1.1 |

| #palettes | 9  | 13 | 5   | 11 | 6  | 6  | 8.3  | 3.2 |
| #tools    | 125| 106| 54  | 77 | 68 | 60 | 81.7 | 28.0|
| #prefs    | 113| 76 | 11  | 51 | 82 | 27 | 60.0 | 37.7|
| #options  | yes| yes| no  | yes| no | yes| yes  | 4.2 |
| macros    | yes| yes| no  | yes| no | yes| yes  | 4.2 |

### Analysis of WIMP applications

- **#menus**: Menus in menu bar
- **#cmds**: Commands in menus
- **#dlogo**: Commands that lead to a dialog box
- **#smenu**: Sub-menus
- **#scmds**: Commands in sub-menus
- **#sdlogs**: Commands in sub-menus that lead to a dialog box

#### Tcmds

- Total commands: #cmds - #smenus + #scmds

#### Tdlogs

- Total dialog boxes: #dlogs + #sdlogs

#### Cmds/M

- Mean commands per menu: #cmds / #menus

#### Cmds/SM

- Mean commands per sub-menu: #scmds / #smenus

#### #palettes

- Palettes and toolbars

#### #tools

- Widgets in palettes and toolbars

#### #prefs

- Preference pages

#### #options

- Options in preference pages

#### macros

- Whether macros can be defined
Successive versions

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Excel 4-&gt;5</th>
<th>Word 5-&gt;6</th>
<th>Photoshop 2.5-&gt;4</th>
</tr>
</thead>
<tbody>
<tr>
<td>#menus</td>
<td>E4 8 8 0%</td>
<td>W5 8 8 0%</td>
<td>P2 7 8 +14%</td>
</tr>
<tr>
<td>#cmd   ds</td>
<td>#dlog 60 44 -27%</td>
<td>#smenu 0 15 + +</td>
<td>#scmds 0 58 + +</td>
</tr>
<tr>
<td>#dlog</td>
<td>107 106 -1%</td>
<td>0 1 +</td>
<td>0 3 + +</td>
</tr>
<tr>
<td>#smenu</td>
<td>0 1 + +</td>
<td>0 0 +</td>
<td>56 82 +46%</td>
</tr>
<tr>
<td>#scmds</td>
<td>0 20 + +</td>
<td>0 0 +</td>
<td>39 40 +3%</td>
</tr>
<tr>
<td>Tcmds</td>
<td>93 127 +37%</td>
<td>107 108 +1%</td>
<td>115 167 +45%</td>
</tr>
<tr>
<td>Tdlogs</td>
<td>60 64 +7%</td>
<td>55 69 +25%</td>
<td>60 67 +12%</td>
</tr>
<tr>
<td>Cmnds/M</td>
<td>11.6 10.5 -10%</td>
<td>13.4 13.3 -1%</td>
<td>11.1 13.9 +25%</td>
</tr>
<tr>
<td>#palettes</td>
<td>8 13 +63%</td>
<td>3 9 +200%</td>
<td>6 11 +83%</td>
</tr>
<tr>
<td>#tools</td>
<td>108 106 -2%</td>
<td>63 125 +98%</td>
<td>49 77 +57%</td>
</tr>
<tr>
<td>#prefs</td>
<td>0 10 + +</td>
<td>0 12 +20%</td>
<td>9 8 -11%</td>
</tr>
<tr>
<td>#options</td>
<td>0 76 + +</td>
<td>52 113 +117%</td>
<td>58 51 -12%</td>
</tr>
<tr>
<td>macros</td>
<td>yes yes</td>
<td>no yes</td>
<td>no yes</td>
</tr>
</tbody>
</table>

Power vs. Simplicity

Simple things should be simple
Complex things should be possible
How to combine power & simplicity?

More is less: the illusion of power

Bloatware
Too many functions
More functions with each new version
### Marketing software: increased power?

Add features
- More menu items - Each is harder to find
- More commands - Each is harder to learn
- More dialog boxes - More steps to the goal

Add programming
- Macros
- Scripting languages
- Require users to understand programming concepts

### Marketing software: increased simplicity?

Add wizards
- Hard to understand: What did the wizard do?
- Lose control: Wizard may do the wrong thing
- Waste time: Must fix the wizard’s mistakes

Add Customization:
- Preferences menus
  - Hard to navigate
  - Hard to translate into user’s terms
  - Hard to choose relevant settings
  - Rarely sharable
  - Most users don’t bother

### Costs vs. benefits

Simple things are harder
Complex things are not used

Cost of learning
- Learned skills made obsolete
- No path from novice to expert

Cost of making choices
- Cognitive: more decisions
- Sensory-motor: more steps

### A better approach

Specializing software
Example: Apple Macintosh

- FinalCut Pro
- Aperture
- iMovie
- iPhoto

---

(c) 2011, Michel Beaudouin-Lafon, mbl@lri.fr
Another approach

- Shifting the curve

Complexity can be simple

Going beyond WIMP

- No menus,
- No windows,
- No dialog boxes

Graphical design
Interaction design
Layered approach

### Comparison: Bryce vs WIMP

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Avg</th>
<th>Bryce2</th>
<th>% of Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>#menus</td>
<td>7.7</td>
<td>3</td>
<td>38.9%</td>
</tr>
<tr>
<td>#cmds</td>
<td>95.2</td>
<td>45</td>
<td>47.3%</td>
</tr>
<tr>
<td>#dlog</td>
<td>36.8</td>
<td>18</td>
<td>48.9%</td>
</tr>
<tr>
<td>#smenu</td>
<td>17.3</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>#scmds</td>
<td>67.0</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>#sdlog</td>
<td>19.7</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>Tcmds</td>
<td>144.8</td>
<td>45</td>
<td>31.1%</td>
</tr>
<tr>
<td>Tdlogs</td>
<td>56.5</td>
<td>18</td>
<td>31.8%</td>
</tr>
<tr>
<td>Cmds/M</td>
<td>12.5</td>
<td>15.0</td>
<td>120.0%</td>
</tr>
<tr>
<td>Cmds/SM</td>
<td>3.9</td>
<td>0.0</td>
<td>0.0%</td>
</tr>
<tr>
<td>#palettes</td>
<td>8.3</td>
<td>9</td>
<td>108.4%</td>
</tr>
<tr>
<td>#tools</td>
<td>81.7</td>
<td>71</td>
<td>86.9%</td>
</tr>
<tr>
<td>#prefs</td>
<td>7.8</td>
<td>1</td>
<td>12.8%</td>
</tr>
<tr>
<td>#options</td>
<td>60.0</td>
<td>5</td>
<td>8.3%</td>
</tr>
</tbody>
</table>
Case study: CPN 2000 Project

Redesign of Design/CPN
Current use world-wide: 600+ organizations

Purpose:
Edit and simulate coloured Petri Nets

Opportunity:
Explore research questions with a real-world application

Two key design decisions

Support two-handed input
Dominant and non-dominant hands

Integrate four interaction techniques:
Toolglasses Floating palettes
Contextual menus Bi-manual interaction

Why these techniques?
User studies show context affects tool preference
Palettes: focus on command
Marking menus: focus on object
Toolglasses: mixed focus

Less is more: the power of simplicity

CPN2000 case study
New version has more power but
no menu bar
no title bars
no scrollbars
no dialog boxes
no selection

This required
Participatory design process
Interaction model
Implementation from scratch
Interaction model

**Definition**
Set of principles, rules and properties that guide the design of an interactive system
Helps combine interaction techniques in a consistent way

**Properties**
- Descriptive: describes a range of existing interactive systems
- Evaluative: helps evaluate interactive systems
- Generative: helps create new interaction techniques

Need for a new interaction model

Direct manipulation... is often too indirect
Support more direct forms of interaction

Instrumental interaction

**Inspiration**
Interaction with our environment is mediated by tools and instruments

Two categories of objects
- Domain objects
- Interaction instruments

Interaction instruments

**Conceptual model**

Two levels of interaction: mediation
Instruments and modes

An instrument turns a mode into an object

Activating a mode = activating an instrument

Spatial mode: pointing

Temporal mode: selection

Cost of activation

Describing current WIMP interfaces

WIMP interfaces are based on widgets

Instruments of (in)direct manipulation

Handles, Title bars

Menus, Toolbars

Scrollbars

Dialog and Property boxes

Describing novel interaction techniques

Dynamic Queries

Dropable Tools

Toolglasses

Describing novel interaction techniques

Tangible interfaces

More input devices and therefore more instruments

Augmented/Mixed reality

Augmenting physical objects with computational capabilities

(c) 2011, Michel Beaudouin-Lafon, mbl@lri.fr
Evaluation : Properties of an instrument

Degree of indirection
- Spatial offset
- Temporal offset

Degree of integration
- How to use the degrees of freedom of the physical device
- Integrality & separability of input devices (Jacob et al., 94)

Degree of conformance
- Similarity between physical action and effect on object

Generative power : Three design principles
- Reification
  - Extends the notion of what constitutes an object
- Polymorphism
  - Extends the power of commands with respect to these objects
- Reuse
  - Provides a way of capturing and reusing interaction patterns
Example: text search instrument

Classic search:
- Sequential
- Modal

Search instrument:
- Show all occurrences
- Allow replacing occurrences in any order
- Augmented scrollbar

Reification

Turns concepts into (interface) objects

Interaction instrument
- Reification of a command into an interface widget

Example:
- Scrolling a document -> scrollbar

Examples
- Guidelines: reification of alignment
- Layers: reification of mode

Polymorphism

Extends commands to multiple object types

Common examples:
- Cut, paste, delete, move

Context-dependent commands

Homogenous groups
- If applicable to one object, then applicable to a group of same-type objects

Heterogeneous groups
- Applicable to a heterogeneous group if it has meaning for individual object types

Reuse

Captures interaction patterns for later reuse

Output reuse
- Reuse previously created objects
  - Example: duplicate, copy/paste

Input reuse
- Reuse previous commands
  - Example: redo, history, macros
Three dimensions of an interface

Objects  Interaction & Visualization  Commands

Reification  Objects  Interaction & Visualization  Reuse

Polymorphism

Magnetic guidelines

Reification of the alignment command

Power and simplicity
- Align command vs Align object:
  - Align (now) vs Align (and keep aligned)

Multiple shapes
- Horizontal, vertical, diagonal, circular, rectangular
- Distribute objects

Decomposition
- Create / Move / Add object / Remove object

Layers

A mode defines:
- Which objects are visible
- Which commands are available

Layer = reification of mode
- Turn layer on/off
- Guidelines, simulation, annotations...

Increased power
- Combine layers

Example in CPN2000: debug mode, simulation mode
Groups

Reification + Polymorphism
Group = reification of a selection
Polymorphism:
  Apply a command to a group = apply it to each object in the group
  Generic commands: Open, Edit, Cut-Copy-Paste

Examples in CPN2000
  Folders = Groups of pages
  Index = Hierarchy of documents and palettes
  Magnetic guidelines = Groups of layout-constrained objects
  Styles = Objects that share graphical attributes

Styles

Reification + Output reuse
Style object
  Reification of a collection of attributes
  Objects that share a style = group
  Editing style affects all objects in group

Style picker
  Copies any object's current attributes

Style dropper
  Applies style to any object

Macros

Input reuse + Reification + Polymorphism
Reuse
  Record a sequence of commands as a macro
Polymorphism:
  Apply macro as a command in new contexts
Reification:
  Edit macro as first class object

Integrating the principles

Reification and polymorphism
  More objects and fewer commands
Reification facilitates output reuse
  More first-class objects can be reused
Polymorphism facilitates input reuse
  Increases the scope of commands
Design principles

Increase simplicity
- Reification: direct instruments not indirect commands
- Polymorphism: fewer commands
- Reuse: copy/redo rather than re-create from scratch

Increase power
- Reification: commands as first-class objects
- Polymorphism: same command works in multiple contexts
- Reuse: path to programming/scripting

Conclusion

Instrumental Interaction makes explicit the artifacts involved in the mediation between user and objects of interest

Descriptive, evaluative and generative model

Design principles help combine power and simplicity