

## Model-Based and Demonstrational Techniques

SMD158  
Interactive Systems  
Spring 2005



---

---

---

---

---

---

---

---

## Overview

- Model-based user interface development
  - Motivation and review
  - Example system, HUMANOID
- Demonstrational user interface development



---

---

---

---

---

---

---

---

## Model-Based User Interface Development



---

---

---

---

---

---

---

---

## Shortcomings of Existing Interface Builders

- Concentrate on static appearance
- Limited to buttons and boxes
- Cannot handle
  - Complex structure (graphics)
  - Complex data types
    - + Example, multimedia, video and sound
  - Runtime determined data
    - + Variable amounts
    - + Time varying

Feb-28-05

© 2001-2005 by David A. Carr

4

L

---

---

---

---

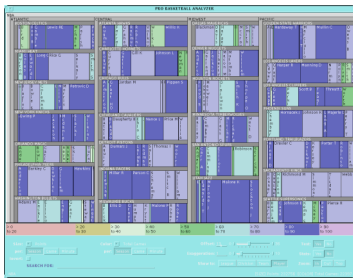
---

---

---

---

## An Example Application, Tree Map of the NBA



from, <http://www.cs.umd.edu/hcil/treemaps/>  
© 2001-2005 by David A. Carr

Feb-28-05

5

L

---

---

---

---

---

---

---

---

## What Can Be Done with an Interface Builder



Feb-28-05

© 2001-2005 by David A. Carr

6

L

---

---

---

---

---

---

---

---

## Goals for Model-Based Approaches

- Increase the percentage of the application that is not directly coded
  - Simplifies and speeds up development
  - Increases exploration of alternative designs
  - Reduces errors
  - Allows non-programmer designers to develop interfaces
- Provides a representation that can be analyzed by tools such as:
  - Design critics
  - Suitability checkers
  - Help systems

Feb-28-05

© 2001-2005 by David A. Carr

7



---

---

---

---

---

---

---

---

## User Interface Model Types

- Task model
  - What the user does and how
- Presentation model
  - What the system displays and how the user interacts
- Application model
  - How the user's actions change the functional core
- User model
  - What the user knows, can do, ...

**Review**

Feb-28-05

© 2001-2005 by David A. Carr

8



---

---

---

---

---

---

---

---

## Example System, HUMANOID

- Concentrates on presentation aspects
- Provides 5 models
  - Application → Application Model
  - Presentation → Presentation Model
  - Behavior → Presentation Model
  - Dialog sequencing → Dialog Model
  - Action side-effects → Dialog Model
- Provides a forms-based interface to the designer

Feb-28-05

© 2001-2005 by David A. Carr

9



---

---

---

---

---

---

---

---

## HUMANIOD, Applications Model

- Data in the Functional Core
- Object oriented
- Defines
  - Objects
  - Global variables
  - Data flow among them (constraints)
  - Defined in an inheritance hierarchy
- Advantages over interface builder
  - Used to generate a default (clunky) interface
  - Makes all design aspects visible
  - Usable by design critics and help generators

Feb-28-05 © 2001-2005 by David A. Carr 10

L

---

---

---

---

---

---

---

---

## Types of Objects

- Commands
  - call-back, pre-conditions, inputs
- Inputs:
  - type, predicate (validates),
  - min+max values or alternatives,
  - parser+unparser (from/to string)
- Application objects
  - Can have input objects

Feb-28-05 © 2001-2005 by David A. Carr 11

L

---

---

---

---

---

---

---

---

## Design Operations on Objects

- Sub-class and modify
- Add data-flow constraints
- Group into composites

Feb-28-05 © 2001-2005 by David A. Carr 12

L

---

---

---

---

---

---

---

---

## HUMANIOD, Presentation Model

- Defines:
  - Containment hierarchy
  - Layout
- Template based
  - Is-A ⇒ parent object in type hierarchy
  - Input data ⇒ display type
  - Widget ⇒ layout primitive (graphical, toolkit, or layout)
  - Applicability conditions ⇒ information about use
  - Parts ⇒ contained object
  - Behaviors ⇒ what you can do with/to it

Feb-28-05

© 2001-2005 by David A. Carr

13



---

---

---

---

---

---

---

---

## HUMANIOD, Behavior Model

- Based on Myers' interactors
- Seven classes of interactor
  - menu
  - move-grow
  - new-point
  - angle
  - text
  - trace
  - gesture
- Slots are filled with code fragments
- Complex operations
  - Iteration
  - Conditionals
- Developer specifies values for 10-20 parameters
  - Start, stop, & abort events
  - Operating & final feedback
  - Continuous/one-shot
  - Start & continue criteria (e.g., mouse position)
  - ...

Feb-28-05

© 2001-2005 by David A. Carr

14



---

---

---

---

---

---

---

---

## Menu Interactor

- General purpose selector
- Widgets include: menu item, button, radio button, checkbox
- Example, tree map object
  - pointed at: details to status line

Feb-28-05

© 2001-2005 by David A. Carr

15



---

---

---

---

---

---

---

---

## HUMANIOD, Dialog Sequencing Model

- Applies constraints to command objects
- Properties of commands
  - Idle/active/running
  - Enabled/disabled
  - Ready/not ready
- Conditional guards in terms of the interface state are used to specify the last two.
- Methods attached to transitions are invoked when states change
- Command groups react to changes in their children

Feb-28-05

© 2001-2005 by David A. Carr

16



---

---

---

---

---

---

---

---

## HUMANIOD, Action Side-Effects Model

- Attached to dialog-sequence methods
- Uses
  - Select a newly created object
  - Error messages
  - Show dialog box

Feb-28-05

© 2001-2005 by David A. Carr

17



---

---

---

---

---

---

---

---

## Summary

- Model-based systems aim to:
  - Provide a more natural level of abstraction
  - Allow designers to create UIs without programming
  - Provide a basis for automatic design aids
- HUMANIOD
  - Presentation-oriented research system
  - Written in LISP
  - Notable for concise specification of presentation & hooks to a programming language
  - Does not consider task or user models

Feb-28-05

© 2001-2005 by David A. Carr

18



---

---

---

---

---

---

---

---

## Demonstrational User Interface Development

Feb-28-05

© 2001-2005 by David A. Carr

19



---

---

---

---

---

---

---

---

## What Is Demonstrational Programming?

- The system observes and records user actions
- The system generalizes these actions by selecting the appropriate objects as variables
- The system stores the augmented record as a program
- Examples:
  - Word's macro recorder
  - Triggers



Feb-28-05

© 2001-2005 by David A. Carr

20



---

---

---

---

---

---

---

---

## Why Demonstrational Programming?

- Provide end-users with the ability to automate repetitive tasks
- Provide macros for visual shells
- Lower the knowledge required to create macros
- Teach a programming language

Feb-28-05

© 2001-2005 by David A. Carr

21



---

---

---

---

---

---

---

---

**Challenges**

- Computational generality
- Data access
- Usability
- Managing risk

Feb-28-05 © 2001-2005 by David A. Carr 22 **L**

---

---

---

---

---

---

---

---

**Computational Generality**

- Variables
  - Must represent objects of interest in the system
  - Must map in a reasonable and unsurprising manner
- Operators
  - Must be useful within system context
  - Must general for the variables
- Conditionals
  - Variable based
  - User input based
- Loops

Feb-28-05 © 2001-2005 by David A. Carr 23 **L**

---

---

---

---

---

---

---

---

**Data Access**

- Objects of interest must be available to the system
- Difficult as objects are:
  - Bound to at least one application
  - Publishing is not a design requirement for most
  - Vary widely in type and complexity
- Data access often limits the applicability of the system

Feb-28-05 © 2001-2005 by David A. Carr 24 **L**

---

---

---

---

---

---

---

---

## Usability

- Program generation
  - User or system initiated
  - Feedback for progress and result
- Program representation
  - Program-like  $\Rightarrow$  user need to be a programmer
  - None
  - Text
  - Pictorial rewrite rules (cartoon frames)
- Editing programs
  - Improves usefulness
  - Users must understand representation
  - Editor must also be usable

Feb-28-05

© 2001-2005 by David A. Carr

25



---

---

---

---

---

---

---

---

## Levels of Inferencing

- None
  - Most user control
  - Least assistance
- Simple, rule-based
  - Understandable for a few rules
  - Local applicability
  - Users often overestimate system abilities
- Complex AI algorithms
  - Least comprehensible
  - Refine programs over time

Feb-28-05

© 2001-2005 by David A. Carr

26



---

---

---

---

---

---

---

---

## Managing Risk

- All systems will be wrong sometimes
- The user must be able to
  - Recognize the errors
  - Correct them
  - Recover from the actions performed
- The more severe the possible consequences the less likely the user is to trust the system

Feb-28-05

© 2001-2005 by David A. Carr

27



---

---

---

---







---

---

---

---

## Other Uses, Widgets by Example

- A simple pushbutton 
  - Mouse over ⇒ 
  - Mouse down ⇒ 
  - Mouse outside ⇒ 
  - Mouse up ⇒ 
  - Mouse outside after up ⇒ 

Feb-28-05

© 2001-2005 by David A. Carr

28



---

---

---

---

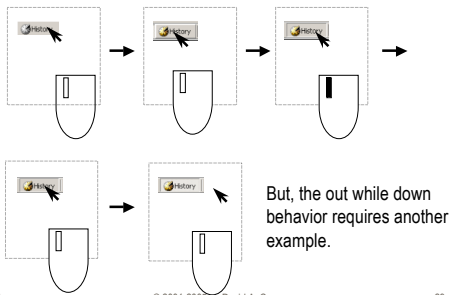
---

---

---

---

## By Demonstration



Feb-28-05

© 2001-2005 by David A. Carr

29



---

---

---

---

---

---

---

---

## Summary

- Intended to reduce the burden for end-user programming
- Can use varying levels of inferencing
  - none, simple rules, complex AI
- Need to meet 4 challenges
  - Computational generality, data access, usability, risk management

Feb-28-05

© 2001-2005 by David A. Carr

30



---

---

---

---

---

---

---

---

Questions?



Feb-28-05

© 2001-2005 by David A. Carr

31

L

---

---

---

---

---

---

---

---