NUR System architectures, Formal descriptions/models of user interfaces

System architecture: (H)MVC, PAC
UI models: Seeheim, ARC, CTT, Flow chart, Petri nets, STN, JSD
Designing Interactive System

USER NEEDS & BEHAVIOR
- Interview transcriptions
- Scenarios & Use-cases
- Storyboards
- User models
- HTA

IDEAS & CONCEPTS
- Sketching
- Design studio

PROTOTYPING
- Lo-Fi prototyping
- Hi-Fi prototyping
- Information architecture

MODELING
- STN, CTT, PN

Source: Buxton 2007
Software system models

(H)MVC, PAC
Architecture: monolithic vs. components

- Seeheim has big components

- often easier to use smaller ones
  - esp. if using object-oriented toolkits

- Smalltalk used MVC – model–view–controller
  - model – internal logical state of component
  - view – how it is rendered on screen
  - controller – processes user input
MVC model - view - controller
MVC

- Three parts: model, views and controllers
- The model is independent of the views and the controllers
- There can be several views and controllers linked to one model
MVC issues

- MVC is largely pipeline model:
  \[ \text{input} \rightarrow \text{control} \rightarrow \text{model} \rightarrow \text{view} \rightarrow \text{output} \]

- but in graphical interface
  - input only has meaning in relation to output
  - e.g. mouse click
    - need to know *what* was clicked
    - controller has to decide what to do with click
    - but view knows what is shown where!

- in practice controller ‘talks’ to view
  - separation not complete
MVC Dynamics

1. User input event routed by Window System to appropriate Controller

2. Controller may require View to “pick” object of focus for event
MVC Dynamics

3. Controller requests method of Model to change its state

4. Model changes its internal state
MVC Dynamics

- 5. Model notifies all dependent Views that data has changed
- 6. View requests from Model current data values
MVC Dynamics

- 7. Model notifies all dependent Controllers that data has changed.
- 8. Controller requests from Model current data values.
MVC Dynamics

- 9. Controller informs View if elements are disabled.
- 10. View requests redraw
Hierarchical MVC
PAC – Presentation – Abstraction – Controller

Agent 1

Agent 2

Agent 3

Agent 4

Agent 5

Presentation

Abstraction

Control

NUR – System architecture (/H/MVC, PAC), formal description of UI (Seeheim, STN, CTT, PN)
Dialog and its structure
Dialog modeling

- Set of states with transitions between them

- Transition between states can be dependent on conditions

- It is possible to assign description of actions to individual transitions
What is dialog?

- Most of dialogs are not structured (structure = structure of a sentence => not sufficient)
- Examples of structured conversation: movie scenario, wedding ceremony,…
- Real dialog with computer is usually structured and limited (not like in Star Trek)
Structured dialogue – wedding ceremony

Minister: do you <man’s name> take this woman …
Man: I do
Minister: do you <woman’s name> take this man …
Woman: I do
Man: With this ring I thee wed
   (places ring on woman's finger)
Woman: With this ring I thee wed (places ring ..)
Minister: I now pronounce you man and wife
Typical features of a dialog

- **Wedding ceremony**
  - Given scenario for 3 participants
  - Sequence of “actions” is given
  - Some parts are fixed „I do“
  - Some parts are variable— “do you man’s name …”
  - What to do with the ring (with words “with this ring …”)

- **When telling these words – are we married?**
  - Only on the right place with the license (minister)
  - Syntax only – not semantics
  - What if some other answer will be said?
Do we understand what a dialog is?

- **Syntactic level of communication between human and computer**

- **Notation for dialogue description**
  - diagrammatic
  - textual

- **Dialog is linked-up with**
  - semantics (usually linked up with application)
  - presentation (to the user)
UI implementation issues
Interface layers / logical components

- Linguistic: lexical/syntactic/semantic
- Seeheim
- Arch/Slinky
Monolithic Reference Architecture

Presentation
Dialogue
Application
Problem with monolithic architecture

- Difficult to modify/maintain
- Every single change is propagated to other parts of the system
- When this problem became an urgent one?

Solution:
- separation of UI part from application
- Existence of several models (no universal model exists)
Feedbacks

- Different kinds of feedback:
  - lexical – movement of mouse
  - syntactic – menu highlights
  - semantic – sum of numbers changes

- Semantic feedback often slower
  - use rapid lexical/syntactic feedback

- But may need rapid semantic feedback
  - freehand drawing
  - highlight trash can or folder when file dragged
Seeheim model

Lexical
Presentation

Syntactic
Dialogue Control

Semantic
Application Interface Model

User → Presentation ← Dialogue Control → Application Interface Model → Application

Lexical
Syntactic
Semantic

?
Seeheim model: the bypass/switch

User \rightarrow Presentation \leftarrow Dialogue Control \rightarrow Application Interface Model \rightarrow Application

Lexical \rightarrow Syntactic \rightarrow Semantic

- User \rightarrow Presentation
- Presentation \rightarrow Dialogue Control
- Dialogue Control \rightarrow Application Interface Model

rapid semantic feedback

? but regulated by dialogue control

direct communication between application and presentation
Conceptual vs. implementation

Seeheim

– arose out of implementation experience
– but principal contribution is conceptual
– concepts part of ‘normal’ UI language

… because of Seeheim …

… we think differently!

e.g. the lower box, the switch

• needed for implementation
• but not conceptual
Seeheim problem

- Low granularity

- Result: problem with software modification
More layers

- dialogue
- lexical
- functional core
- adaptor
- functional core
- lexical
- physical

More layers
Arch model

- Domain Adaptor
  - Domain Objects
- Domain Specific
- Presentation
  - Presentation Objects
- Interaction
  - Interaction Objects
- Toolkit
Dialog - formal description
Diagrammatic notation

- Frequently used (picture gives us a nice overview)
- Dialog structure – at the first glance
- What to do with large and complex dialogues
- Typical diagrams used
  - STN - State transition networks (STD)
  - Petri nets
  - Flowcharts
  - JSD diagrams
Textual description

- Non-formal description (in common language)

- Grammars

- Some other theoretically based descriptions (production rules ...)

NUR – System architecture (/H/MVC, PAC), formal description of UI (Seeheim, STN, CTT, PN)
Why to describe dialog?

- The purpose
  - Communication with other designers
  - Tool in early phase of design (brainstorming - ideas)

- How to embed semantics?
  - The users can take an active part in discussions
  - The users can suggest extension of functionality
  - We complete the dialog description by intended meaning (semantics) of a new action
Dialog model: State diagrams
OUR DESIGN: VIDEO EXAMPLE
OUR DESIGN: GUIDELINES

- information architecture
- interaction design
- visual design
OUR DESIGN: NAVIGATION SCHEME

- 6-key navigation
- BACK as a shortcut
- no interference between basic and advanced functions
- simple views
OUR DESIGN: PROTOTYPE

Search

Watching

On demand

TV

Radio

Apps

Settings

FEATURE CHOICE

CATEGORY CHOICE

Watch TV

TV Guide

Mosaic

Reminders

Recording plans

Actions

Recording

Shortcuts

Seek

Playback

Detail

ACTION CHOICE

CATEGORY CHOICE

NUR – System architecture (H/MVC, PAC), formal description of UI (Seeheim, STN, CTT, PN)
OUR DESIGN: FINAL DESIGN

- HTML + JavaScript
How to work with system states

- A lot of formalisms exists
- State diagrams
  - transition diagrams
  - transition networks
  - Principle: INPUT -> transition from the current state into new one

- Examples from everyday life?
Example state: switch

- state: off
- state: on

![Diagram of switch state transitions](image-url)
State diagram - example

Alarm control

[Diagram of state transitions: Green, Yellow, Red]

Alan J. Dix
Why state diagrams?

- Formal description of UI behavior
- Dialog is represented as a set of states with transitions between them
- The course of dialog is linked-up with the current state
- Transition between states can be conditional
- Manifestation of transitions can be added
  - change of screen
State diagrams - some rules

- States cannot overlap or intersect
- Have exactly as many arrows from a state as there are possible actions from that state
- Each arrow is labeled with its action; when action name matches
  - consequent state might omit, e.g. ‘off’ action leads to ‘off’ state
- It may be convenient to merge arrows that go to the same state.
Dialog model are good for

- Experiments with model (see in next slides)
  - distinguish between control inputs and application inputs
  - change of screen state

- Discovery of all possible paths in the model
  - check if all paths end up in proper states – not in a state that is not a final one and has no output etc.)
  - automatic check (graph algorithm?)

- Possibility of (semi-)automatic UI creation
State transition networks (STN)

- circles - states
- edges - action/events
State diagrams = models

- What brings us the use of these models?
- Simulation of interaction
- Check the user ability to cope with UI
- Check the functionality
State transition networks - events

- Transitions are hard to read and interpret:
  - Notation includes a lot of states ('state heavy')
  - Events require too many details
State transition networks - states

- Circle notations are rather unintuitive
  - States are hard to name
  - They can be drawn easily
State transition network - transitions

- Select 'circle'
- Highlight 'circle'
- Click on centre
- Rubber band
- Draw circle
- Finish

Start → Menu

Start → Menu

- Click on first point
- Rubber band
- Double click
- Draw last line

Line1

Line2

- Click on point
- Draw line and
- rubber band from new point

Arc from each state back to menu
Becomes messy!

Press escape key
Hierarchical STN

- Complex dialogs can be described
- Naming sub-dialogs

![Diagram of hierarchical STN showing main menu with branches for graphics, text, and paint sub-menus]

Select 'graphics'
Select 'text'
Select 'paint'
Hierarchical state transition network
Augmented transition networks (ATNs)

- Form of STN
  - We assume existence of several "registers" that are set before transition (and tested afterwards)
    - if condition is true and event occurs, follow arc

- Example: How many times wrong PIN was used
  - three times – either three inputs or semantic approach: register that is tested each time – till the number 3 has been achieved

- Example: How many times we clicked when drawing the line
More complex examples

More dialogs in parallel
Concurrent dialogs - individual STNs

- Click on ‘bold’
- Click on ‘italic’
- Click on ‘underline’
Concurrent dialogs – bold and italic combined

- **NO style**
  - click on ‘bold’
  - click on ‘italic’
  - bold only
  - italic only

- **bold only**
  - click on ‘italic’
  - click on ‘bold’
  - **italic**
  - **bold italic**

- **italic only**
  - click on ‘bold’
  - click on ‘italic’

Text Style:
- bold
- italic
- underline
Concurrent dialogs - all together - combinatorial explosion

Text Style
- bold
- italic
- underline

Example

NUR – System architecture (/H/MVC, PAC), formal description of UI (Seeheim, STN, CTT, PN)
State properties

- Availability
  - Can we get from anywhere to anywhere?
  - How easy is it?

- Reversibility
  - Can we get to the previous state?
  - Not an UNDO

- Dangerous states
  - We do not want to get there
Home work: what interaction technique is this?

- **Input events**
  - B: mouse button click
  - Mv: cursor movement

- **Application operation**
  - A1: store start point
  - A2: store end point

- **Output**
  - O1: track cursor
  - O2: draw line from start to end point
Another homework

What is it? How to convert it into STN?

- Grammars
  - BNF (Backus-Naur Form)
    - Dialog syntactic level
    - Used widely to specify the syntax of computer programming languages.
  - Example: line-drawing function

\[
\begin{align*}
\text{draw-line} &::= \text{select-line} + \text{choose-point} + \text{last-point} \\
\text{select-line} &::= \text{position-mouse} + \text{CLICK-MOUSE} \\
\text{choose-point} &::= \text{choose-one} \\
\text{choice} &::= \text{choose-one} + \text{choose-point} \\
\text{Last-point} &::= \text{position-mouse} + \text{DOUBLE-CLICK-MOUSE} \\
\text{position-mouse} &::= \text{empty} | \text{MOVE-MOUSE} + \text{position-mouse}
\end{align*}
\]
STN = graph => applying graph theory

- Shortest route around graph that includes every arc
- Check that every action works as specified
- Determine length of tour
  - a measure of how hard the device will be to test, document, understand or explore
STN = graph => applying graph theory

- Connectivity:
  - For most systems, a user should be able to get from any state to any other, i.e., the STN should be \textit{strongly connected}.

Cyclic graph – just one button. We go through all states.
Automating usability checks

- A state transition network is a finite state machine

- We can describe the device in a computer program:
  - List of states
  - List of actions

- Matrix of actions x states describing transitions
  - Can automatically generate the transition diagram
  - Can automatically find shortest paths

- Provide user instructions; generate the help manual
  - Can check if some path lengths are unreasonably long
  - Can make frequently used actions easier (e.g. larger buttons)
Testing STD

- Test to find errors in the design and the implementation
  - The state transitions should be made visible during testing

- Check
  - the action carefully
  - the state carefully
  - dead states

- Ensure events that are not supposed to be possible, really cannot happen
Weather forecast HTA

1. Manage settings
   1.1 set favorite places
   1.2 set units

2. Select place
   2.1 select country
   2.2 select region
   2.3 select town

3. Select date range

4. View weather forecast
   4.1 chose wind forecast
   4.2 chose temp. forecast
   4.3 chose precip. forecast

Plan 1: 2.1 – 2.2 – 3 – 4.2
Plan 2: 2.3 – 3 – 1.1 – {4.1, 4.2, 4.3}
Weather forecast STN

1. Start
2. Tap on WF icon to view Weather Forecast for 1st 7 days
3. Swipe left/right to view Weather Forecast for 1st 7 days
4. Tap on "i" icon to get more information
5. Swipe left/right to view Weather Forecast for next 7 days
6. Tap on "+" to add new place
7. Tap on "back" to go back
8. Add new place
Concurrent Task Tree (CTT)
Symbols used in CTT
## CTT – operators used

<table>
<thead>
<tr>
<th>Category</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabling</td>
<td>T1 &gt;&gt; T2 or T1 [] &gt;&gt; T2</td>
</tr>
<tr>
<td>Disabling</td>
<td>T1 [&gt; T2</td>
</tr>
<tr>
<td>Interruption</td>
<td>T1</td>
</tr>
<tr>
<td>Choice</td>
<td>T1 [] T2</td>
</tr>
<tr>
<td>Iteration</td>
<td>T1* or T1{n}</td>
</tr>
<tr>
<td>Concurrency</td>
<td>T1</td>
</tr>
<tr>
<td>Optionality</td>
<td>[T]</td>
</tr>
</tbody>
</table>
CTT example
Another CTT example
CTT example – what task is it?

Diagram:
- Handle request
- Specify request
- Send request
- Provide result
- Specify departure
- Specify arrival
- Specify type of seat
- Specify smoking seat
- Present result
- Present error msg
CTT – “KOS – like” example

NUR – System architecture (/H/MVC, PAC), formal description of UI (Seeheim, STN, CTT, PN)
Other models
Flowcharts

- **Blocks**
  - processes/events
  - stateless

- **Dialog description**
  - not the algorithm description!!!
JSD diagrams

Jackson System Development/Design

For tree structures

- Poor expressiveness
- Good visual appearance

![Diagram of Personnel Record System]

- login
- transaction
- logout
  - add employee record
  - change employee record
  - display employee record
  - delete employee record
Dialog description - Summary

- **Diagrammatic**
  - STN, Flow charts, JSD

- **Textual**
  - grammars, production rules, event algebras

- **Issues**
  - Based on events and transitions between states
  - Powerful description and easy to "read" and interpret
  - model vs. description
  - Sequential vs. parallel
Dialog model: Petri nets (PN)
What are Petri nets (PN)?

- Similar to finite state automata
- Transition between states is made by “token shift”
- Event can trigger the transition only in the case when tokens are on all inputs
- Result of a transition: tokens are removed from inputs and they are placed on outputs (synchronization)
- PN are applied mostly in HW applications (synchronization)
PN: basic programming constructs

- Sequencing

- Synchronization

- Concurrency
PN: basic programming constructs

- Selection

- Control Branching

- Looping
PN example: vending machine

- finite-state machine
- accepts either nickels or dimes
- sells 15c or 20c candy bars
- vending machine can hold up to 20c
- it does not return coins
PN example: coffee maker

- money added
- start cleaning
- cleaning done
- cleaning off
- start water heater
- temp. sensor
- heater off
- 3 sec. delay
- stop grinder
- start grinder
- drop cup
- 10 sec delay
- filter coffee
- pour coffee
- cup removed
Homework – analyze the following examples
An Example of a State Transition Diagram
Microwave oven model

- Full power
- Half power

**Full power**
- do: set power = 600

**Waiting**
- do: display time

**Half power**
- do: set power = 300

**Set time**
- do: get number
- exit: set time

**Operation**
- do: operate oven

**Enabled**
- do: display 'Ready'

**Disabled**
- do: display 'Waiting'

**Number**
- Full power
- Timer

**Operation**
- Cancel

**Door closed**
- Start

**Door open**
- Waiting
- do: display time
Example question for examination

- Transform given menu structure to STN.
- What are the standard constructs of PN?
- What are the key features of STN?
- What are the operators of CTT?
Thank you for attention