Interaction Design
Specification

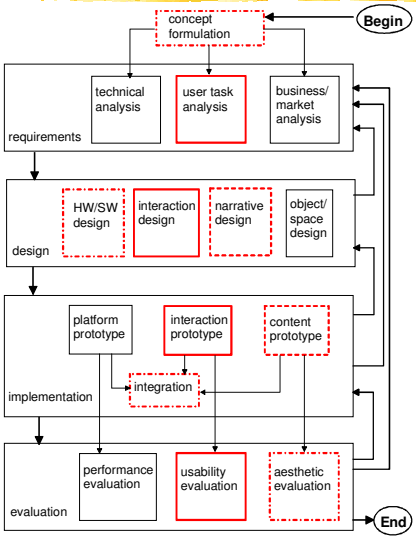
Prof. Dr. Matthias Rauterberg
Faculty Industrial Design
Technical University of Eindhoven
g.w.m.rauterberg@tue.nl

21-February-2008

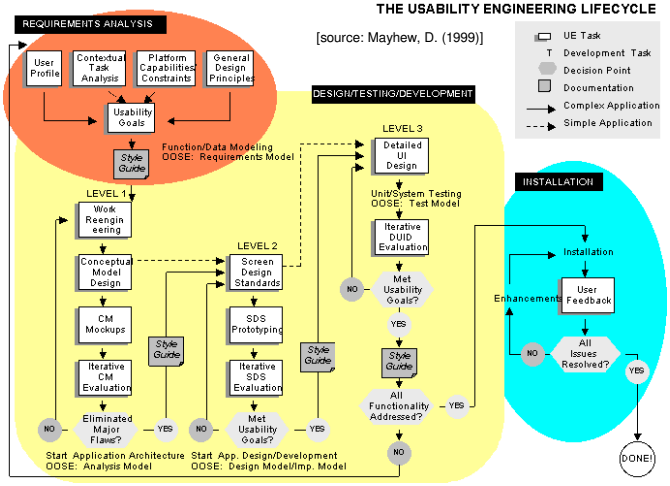
Key references/literature:

- Lifecycle Model:
  Mayhew, D. (1999), The usability engineering lifecycle. Morgan Kaufmann. [ISBN 1-55860-661-4]
- STD:
  Gersting, J.L. (1999), *Mathematical Structures for Computer Science*. 4th Edition, Freeman. [ISBN: 0716783061]
- PN:
  Reisig, W. (1992), *A Primer in Petri Net Design*. Springer. [ISBN: 0387520449]
- GOMS:
  Card S, Moran T, & Newell A (1983), *The psychology of human computer interaction*, Lawrence Erlbaum Assoc, Hillsdale, NJ
- UAN:
  Hartson, H. R., Siochi, A.C., & Hix, D. (1990), "The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs", *ACM Transactions of Information Systems*, 8(3), pp. 181-203.
- Hix, D. & Hartson, H. R., (1993) "Developing User Interfaces", Wiley. (Chapter 6 and 7)
- Hartson, H.R. & Gray P. D. (1992), "Temporal Aspects of Tasks in the User Action Notation", *Human Computer Interaction*, 7(92), pp. 1-45.

The Lifecycle Model



The Usability Engineering Lifecycle



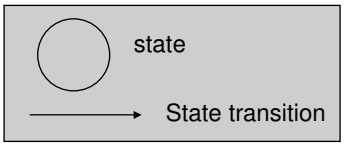
User Interaction Specification

- Many approaches/notations to specifying interaction
  - State-Transition-Diagrams (STD)
  - Petri Nets (PN)
  - Goal-Operation-Methods-Selections (GOMS)
  - User Action Notation (UAN)

- Aim to provide more detailed descriptions of interaction between user and system
- Refinement of task model in terms closer to system
- Provides medium of discussion and review between human factors designers and systems developers

State Transition Diagram (STD)

Basic Elements



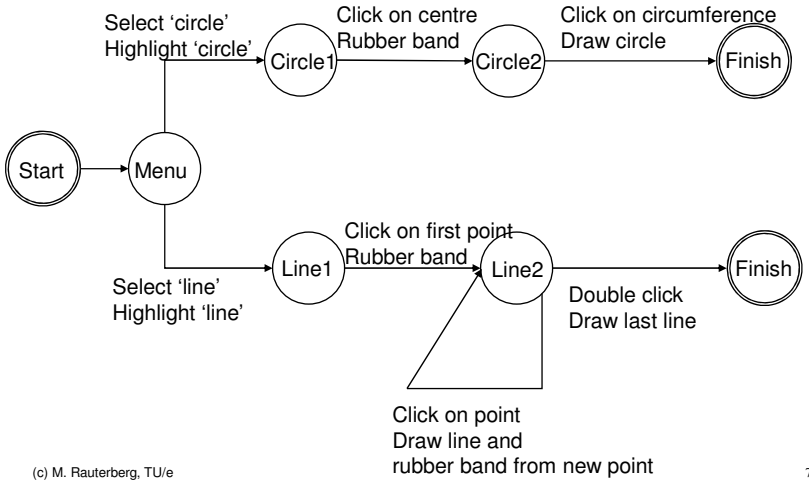
State
= set of values that describe an object (its condition/situation) at a specific moment in time

{State is determined based on the attribute values}

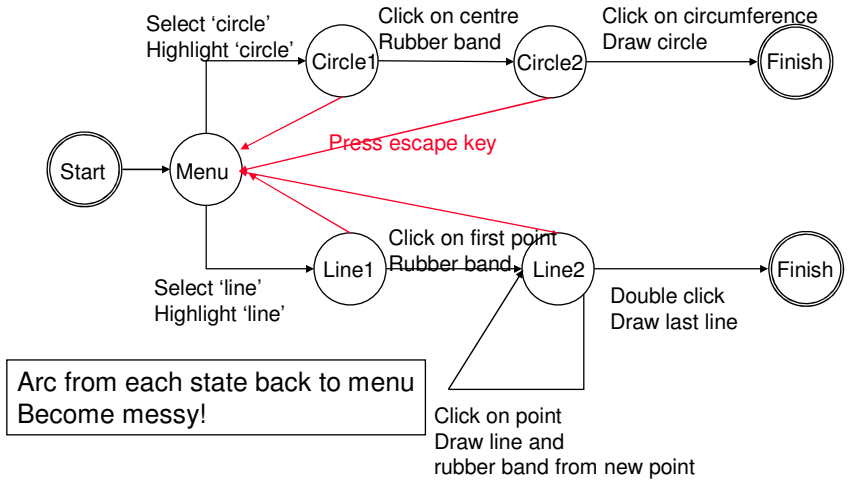
State transition
= relationship indicating a state change

{atomic (i.e. non-interruptible)}

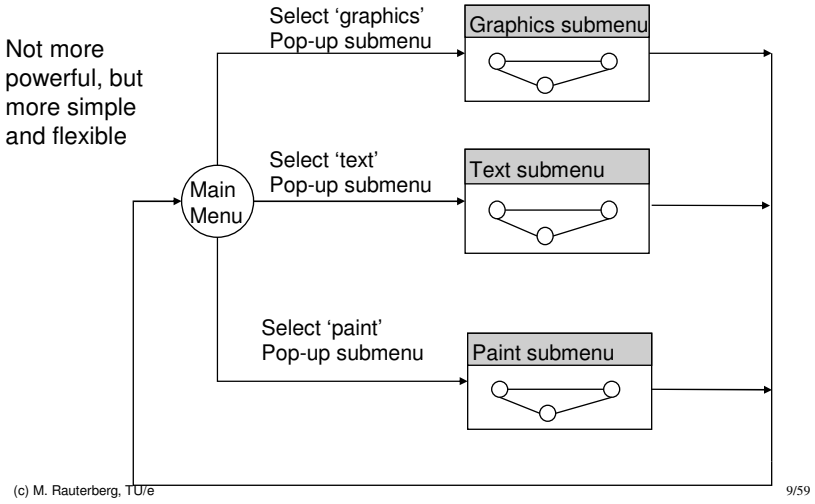
STD Example 1: Draw Circle



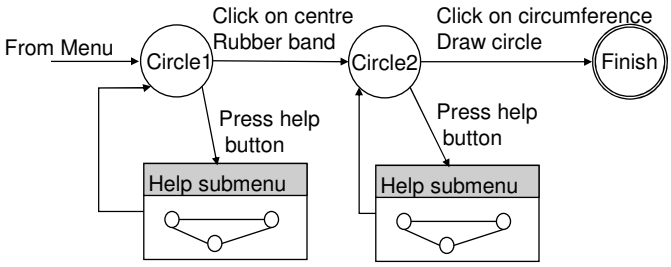
STD Example 1 (cont'd)



Hierarchical STD Example 2



Hierarchical STD Example 2 (cont'd)

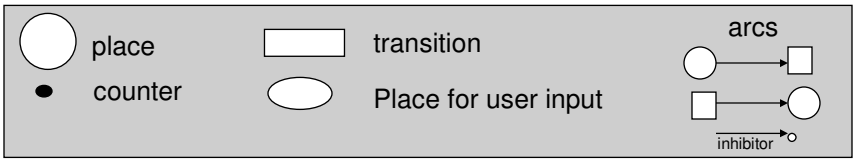


Petri Nets (PN)

- First introduced by Carl Adam Petri in 1962.
- A diagrammatic tool to model concurrency and synchronization in distributed systems.
- Very similar to State Transition Diagrams.
- Used as a visual communication aid to model the system behaviour.
- Based on strong mathematical foundation.

PN: Building Blocks

Basic Elements



- PN consists of three types of components: *places* (circles), *transitions* (rectangles) and *arcs* (arrows):
  - **Places** represent possible states of the system;
  - **Transitions** are events or actions which cause the change of state; And
  - *Every* **arc** simply connects a place with a transition **or** a transition with a place.

PN: Formal Definition

A Petri net (PN) is a 5 tuple
PN (P,T,IN,OUT,M)
where:

P = {p1,p2,....,p~} is a finite set of places,
T = $\{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions
IN: (PxT)→S
OUT: (TxP)→S
M: Marking vector

PN: Formal Definition (cont'd)

IN are input functions defining directed arcs from places to transitions
OUT are output functions defining directed arcs from transitions to places
S is a set of all nonnegative integers k such that:

- If k = 1 a directed arc is drawn without a label
- If k > 1 a directed arc is drawn with label k.
- If k = 0 no arc is drawn.

PN: Firing Rules for Transitions

- A specific transition t_i is said to be **enabled** if each input place p_i is marked with at least $w(p_i,t_i)$ tokens where $w(p_i,t_i)$ is the weight of the arc from p_i to t_i.
- An enabled transition may or may not fire depending on whether or not the event actually takes place .
- The firing of an enabled transition t_i removes $w(p_i,t_i)$ tokens from each input place p_i of t_i, and adds $w(p_j,t_i)$ tokens to each output place p_j of t_i where $w(p_i,t_i)$ is the weight of the arc from input place p_i to t_i, and $w(p_j,t_i)$ is the weight of the arc from t_i to output place p_j

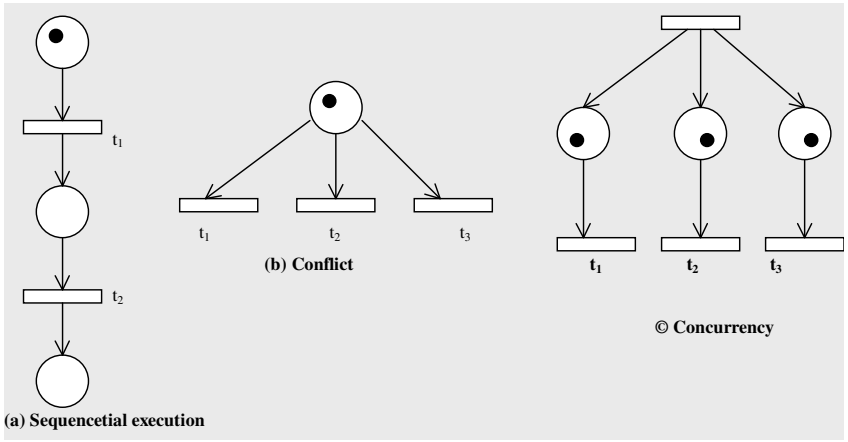
PN: Change of States (1)

- is denoted by a movement of *token(s)* (black dots) from place(s) to place(s); and is caused by the *firing* of a transition.
- The firing represents an occurrence of the event or an action taken.
- The firing is subject to the input conditions, denoted by token availability.

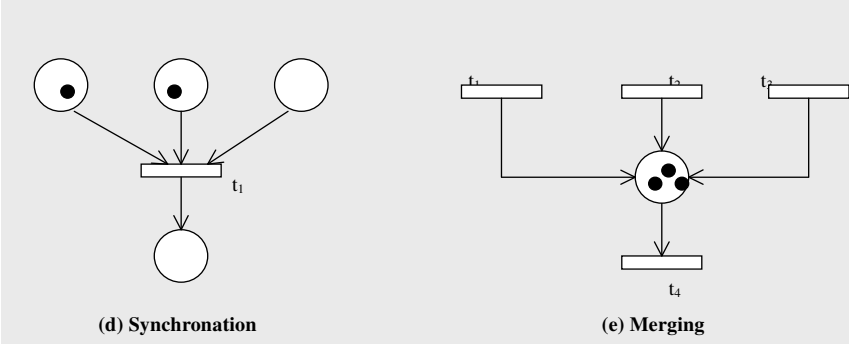
PN: Change of States (2)

- A transition is *firable* or *enabled* when there are sufficient tokens in its input places.
- After firing, tokens will be transferred from the input places (old state) to the output places, denoting the new state.
- Note that the examples are Petri nets representation of a finite state machine (FSM). PNs are much more powerful to model systems beyond FSMs.

PN: basic modeling (1)



PN: basic modeling (2)

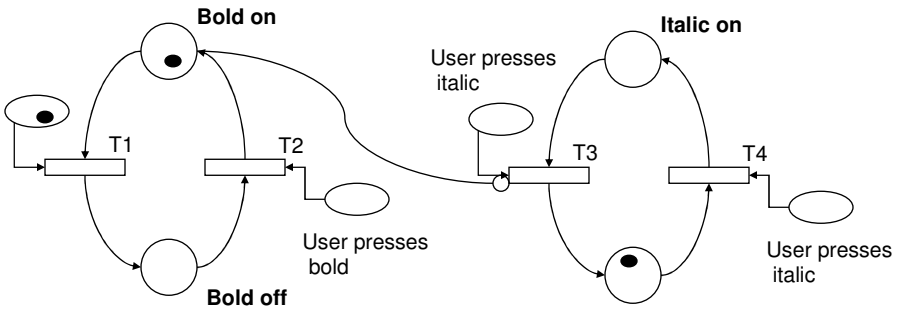


(d) Synchronation (e) Merging

PN: basic modeling (3)

(f) Confusion (g) Priorites

PN Example: Font Selection



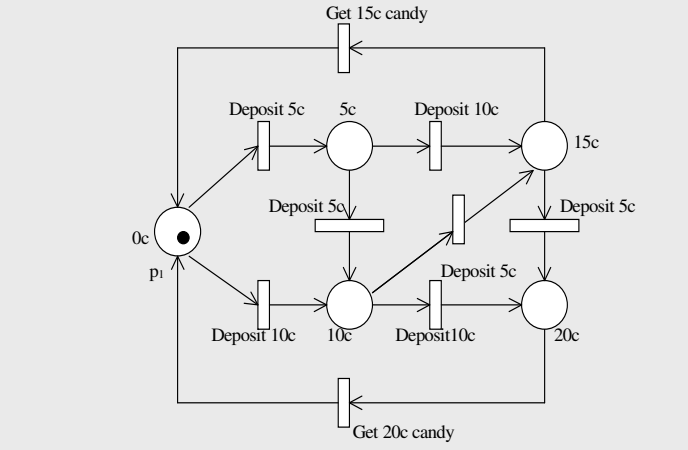
PN Example: a finite-state machine (1)

Consider a vending machine

- It accepts either nickels or dimes
- Sells 15c or 20c candy bars
- The vending machine can hold up to 20c
- Coin return transitions are omitted

the next slides are the state diagram of this vending machine which represented by the Petri net Any finite-state machine (or its state diagram) can be modeled with a state machine.

PN Example: a finite-state machine (2)



Question: What is missing in this specification?

What is GOMS?

- A family of user interface modeling techniques
- Goals, Operators, Methods, and Selection rules
- Input: detailed description of UI and task(s)
- Output: various qualitative and quantitative measures
- Usefully approximations possible
- Based on Model Human Processor

Members of GOMS Family

- **K**eystroke-**L**evel **M**odel (KLM) - [see Card, Moran, Newell (1983)]
- Natural GOMS Language (NGOMSL) - [see Kieras (1988+)]
- **C**ritical **P**ath **M**ethod or **C**ognitive, **P**erceptual, and **M**otor GOMS (CPM-GOMS) [see John (1990+)]

What GOMS can model

- Task must be goal-directed
  - Some activities are more goal-directed than others
  - Even creative activities contain goal-directed tasks
- Task must a routine cognitive skill - as opposed to problem solving as in Cognitive Walkthrough
- Serial and parallel tasks

GOMS Output

- Functionality coverage and consistency
  - Does User-Interface contain needed functions?
  - Are similar tasks performed similarly? (NGOMSL only?)
- Operator sequence
  - In what order are individual operations done?
  - Abstraction of operations may vary among models

GOMS Output (cont'd)

- Execution time
  - By expert
  - Very good rank ordering
  - Absolute accuracy ~10-20%
- Procedure learning time (NGOMSL only)
  - Accurate for relative comparison only
  - Does not include time for learning domain knowledge
- Error recovery

Applications of GOMS

- Compare User-Interface designs
- Profiling
- Sensitivity and parametric analysis
- Building a help system
  - GOMS modeling makes user tasks and goals explicit
  - Can suggest questions users will ask and the answers

Other GOMS techniques

- NGOMSL
  - Regularized level of detail
  - Formal syntax, so computer interpretable
  - Gives learning times
- CPM-GOMS
  - Closer to level of Model Human Processor
  - Much more time consuming to generate
  - Can model parallel activities

GOMS Approach

Goals are what the user wants to achieve

Operators are basic actions user performs

Methods: decomposition of a goal into subgoals/operators

Selection means of choosing between competing methods

GOMS notation

Basic Elements

GOMS is a textual notation formalism.
[to make it as clear as possible, from now on all pre-specified GOMS terms are in **red**!
All symbols in **blue** are use as a meta-syntax defined by Backus-Naur-form (BNF)]

Goal:
followed by name of the [sub]goal (e.g. goal: read-text)

Op[eration]:
followed by name for the operations (e.g. op: write-text
or operation: write-text)

Method:
followed by name for the defined sequence of operations plus all these operations

Selection[rule]:
followed by name for the rule plus conditions for method selection

Concept: Goals

- Something the user wants to achieve
- Examples:
  - goal: go-to-airport
  - goal: delete-File
  - goal: create-directory
- Hierarchical structure
  - may require many subgoals

Concept: Methods

- Sequence of steps to accomplish a goal
  - goal decomposition
  - can include other goals
- Assumes method is learned & routine
- Example

  method: drag-file-to-trash

  op: select-an-object
  op: click-on-object
  op: drag-object-to-trash

Concept: Operators

- Specific actions (small scale or atomic)
- Lowest level of analysis
  - can associate with times
- Examples
  op: Locate-icon-for-item-on-screen
  op: Move-cursor-to-item
  op: Hold-mouse-button-down
  op: Locate-destination-icon
  op: User-reads-the-dialog-box

Concept: Selection Rules

- If [more than one method to accomplish a goal] (Selection rule)
  Then (method or operation to use)
- Examples

```
IF (condition) THEN (accomplish GOAL)
IF (car has automatic transmission) THEN
   (select drive)
IF (car has manual transmission) THEN (find car
  with automatic transmission)
```

Operators vs. Methods

- Operator: the most primitive action
- Method: requires several Operators or subgoal invocations to accomplish
- Level of detail determined by
  - KLM level - key press, mouse press
  - Higher level - select-Close-from-File-menu
  - Different parts of model can be at different levels of detail

Description 'How to Use GOMS'

```
Goal: Generate-task-description
   Op: pick high-level user goal
   Op: write Method for accomplishing Goal
   remark: may invoke subgoals
   Op: write Methods for subgoals, this is recursive
   if (Operator level is reached) then (stop)
Goal: Check-quality-of-task-description
   Op: Evaluate description of task
Goal: Validate-task-description
   Op: Apply results to User-Interface
Goal: Improve-quality-of-task-description
   Op: Iterate
```

GOMS Example 1: PDA Text Entry

```
goal: enter-text-PDA
    op: move-pen-to-text-start
    goal: enter-word-PDA repeat until no-more-words
        op: write-letter repeat until no-more-letters
        if (goal: correct-misrecognized-word)
          then (method: correct-word)

Method: correct-word
   goal: correction-of-misrecognized-word
    op: move-pen-to-incorrect-word
    op: delete-incorrect-word
    op: write-correct-word
```


GOMS Example 2: Iconise Window

```
GOAL: ICONISE-WINDOW
method: USE-CLOSE-METHOD
       GOAL: Close-window-with-mouse
            op: MOVE-MOUSE-TO-WINDOW-HEADER
            op: POP-UP-MENU
            op: CLICK-OVER-CLOSE-OPTION
method: USE-F7-METHOD
       GOAL: Close-window-with-key
            op: PRESS-F7-KEY
Selection:
 If (application is GAME) then (method: USE-F7-METHOD)
 If (application is NOT GAME) then (method: USE-CLOSE-
METHOD)
```

GOMS & KLM Example 2 (cont'd)

Six execution phase operators

- Physical motor
  - K - key stroking
  - P - pointing
  - H - homing
  - B - button pressing
- Mental
  - M - mental preparation
- System
  - R - response

Times are empirically determined (T=Task).

T_execute = T_K + T_P + T_H + T_B + T_M + T_R

GOMS & KLM Example 2 (cont'd)

assume hand starts on mouse

USE-F7-METHOD		**USE-CLOSE-METHOD**	
H[to keyboard]	0.40	P[to menu]	1.1
M	1.35	B[LEFT down]	0.1
K[F7 key]	0.28	M	1.35
		P[to option]	1.1
		B[LEFT up]	0.1
Total	**2.03 secs**	**Total**	**3.75 secs**

GOMS Example 3: Graph Drawer

```
goal: draw-graph
    goal: draw-node repeat until no more nodes
        goal: draw-circle
            op: draw-circle-gesture
            goal: verify-circle-gesture
                if (misrecognized or drawn incorrectly)
                  then (goal: correct-gesture)
            goal: connect-node repeat until no more
              connections
                op: draw-line-gesture
                op: move-pen-to-node-just-drawn
            goal: name-node
                op: make-naming-gesture
                goal: enter-text
```


GOMS Example 3 (cont'd)

```
goal: correct-gesture
    op: move-pen-to-undo-button
    op: tap-undo-button
goal: copy-node
    op: move-pen-to-node
    op: draw-copy-gesture
    op: drag-pen-to-destination
```

GOMS Example 4: DOS File Delete

Goal: Delete-a-File
Method: deleting-a-file-in-DOS
- **op:** retrieve from Long term memory that command verb is "del"
- **op:** think of directory name & file name and make it the first listed parameter
- **op:** accomplish goal of entering & executing command
- **op:** return with goal accomplished

GOMS Example 5: Mac File Delete

Method: deleting-a-file-in-MAC-OS
- **op:** find file icon
- **op:** accomplish goal of dragging file to trash
- **op:** return with goal accomplished

Selection:
If (application is DOS)
  then (method: deleting a file in DOS)
If (application is MAC)
  then (method: deleting a file in MAC-OS)

Advantages of GOMS

- Gives qualitative & quantitative measures
- Model explains the results
- Less work than user study – no users!
- Easy to modify when UI is revised
- Research: tools to aid modeling process since it can still be tedious

Disadvantages of GOMS

- Not as easy as HE, guidelines, etc.
- Takes lots of time, skill, & effort
- Only works for goal-directed tasks
- Assumes tasks performed by **experts** without **error**
- Does not address several UI issues,
  - readability, memorizability of icons, commands

User Action Notation (UAN)

- Developed by Hix and Hartson
- Further refined by Hartson and Gray
- Is neutral about the user and interface technology
- Aims to show how tasks match computer devices
- Elements
  - Symbols and operators
  - Conditions and options
  - Tables of user action, feedback and system action/state
  - Temporal relations and constraints

UAN: Symbols and Operators

- Existing UAN uses special characters for mouse movement and button actions
  [remark: in this introduction **text notation** will be used]
- Example operators in UAN

  move_mouse(x,y)*

  release_button(x',y')

  highlight(icon)

  de_highlight(icon)

  file = select()

UAN: Conditions and Options

while (condition) TASK

if (condition) then TASK

iteration A* or A+

waiting can be an operation on a task

UAN: Tables

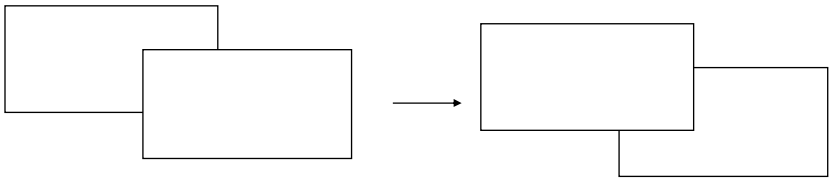
- Three columned table

USER ACTION	FEEDBACK	SYSTEM STATE
clicking mouse entering text moving mouse	highlighting object echo characters show icon moving	selecting file setting string NULL

UAN: Temporal relations

- strict sequence A,B
  - B follows completion of A
- Order independence A&B
  - A and B can be done in any order
- Concurrence with A|| B
  - A and B are done simultaneously
- Interruptible by A->B
  - A can interrupt B
- Interleavable A<|>B
  - Swapping between A and B

Move object from front to back



UAN for moving object to back

select_object, choose_bring_to_back

USER-ACTION	FEEDBACK	SYSTEM-STATE
click(x,y)		if intersect_object
	then	
	highlight_object	object=selected
send_to_back_item	move_to_back	
		move_object_in_list

send_to_back_item is a separate UAN task for standard menu item selection

Note: object is still highlighted and selected at end of task

UAN: Example 1

- drag and drop a file in the recycling bin

USER-ACTION	FEEDBACK	SYSTEM-STATE
mouse_down(x,y)		if intersect(icon,x,y) icon = selected
	then highlight(icon)	
drag_icon(x,y)*	show_outline(icon)	
		if intersect(bin,x,y)
	then hightlight(bin)	
mouse_up(x',y')		if intersect(bin,x',y')
	then hide(icon) show_bin_full()	

UAN: Example 1a

- That was only a partial example:
- Amend it to show what happens if the mouse is released without the icon over the bin
- Generalize the example to drag an icon over any object, e.g.
  - Bin
  - Folder
  - Application

(Hint: will need to use conditions)

UAN: Example 1b

USER ACTION	FEEDBACK	SYSTEM STATE
mouse_down(x,y)		if intersect(object,x,y) object = selected
drag_icon(x,y)*	then highlight(object) show_outline(object)	
	then highlight(target)	if intersect(target,x,y)
mouse_up(x',y')	then hide(object)	if intersect(target ,x',y')
	else draw(object)	intersect_action(target)

Generic drag and drop interaction: action depends on target

UAN: Example 2

- Clicking on a URL with temporal

USER ACTION	FEEDBACK	SYSTEM STATE
mouse_down(x,y)		if intersect(URL,x,y)
	then colour(link)	
mouse_up(x,y)		if intersect(URL,x,y)
	fetch(URL)	URL = visited

What if after mouse_up() I click on another URL?

Two choices

A, B Must complete the first selection before making another

or

A <- B Can interrupt the first task by the second