Discrete Interaction Design

Specification

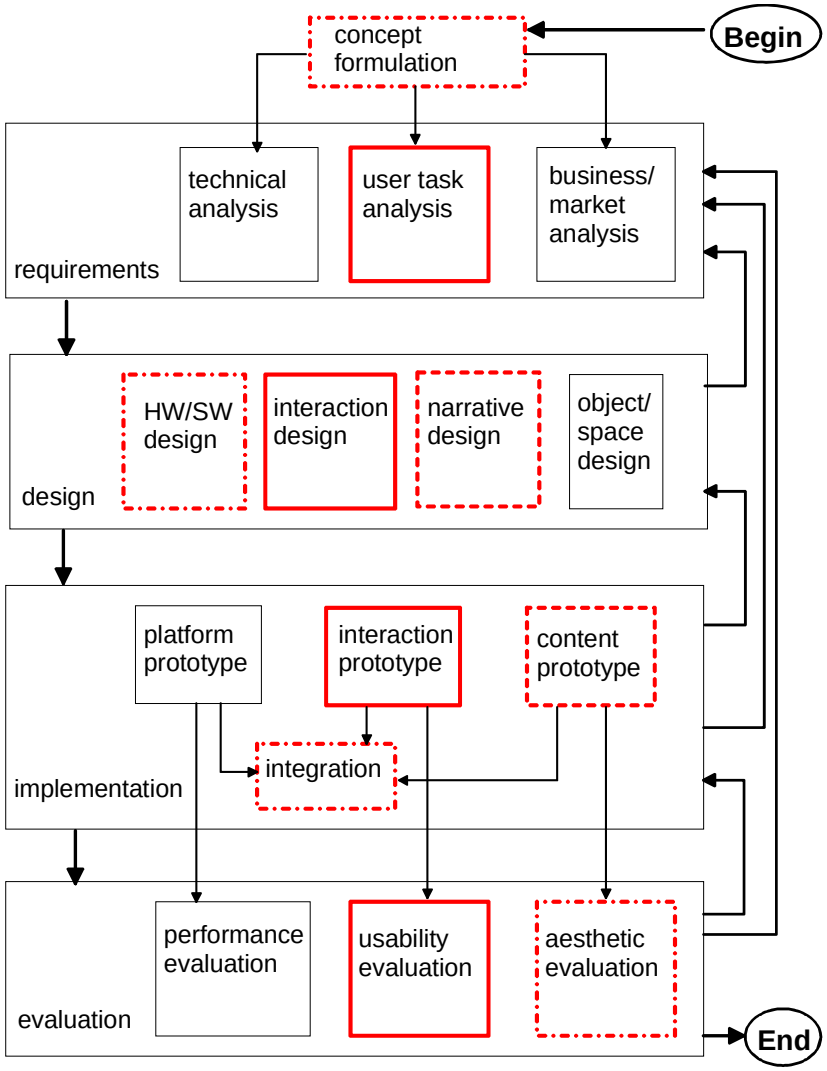
Prof. Dr. Matthias Rauterberg
Faculty Industrial Design
Technical University of Eindhoven
g.w.m.rauterberg@tue.nl

6-April-2009

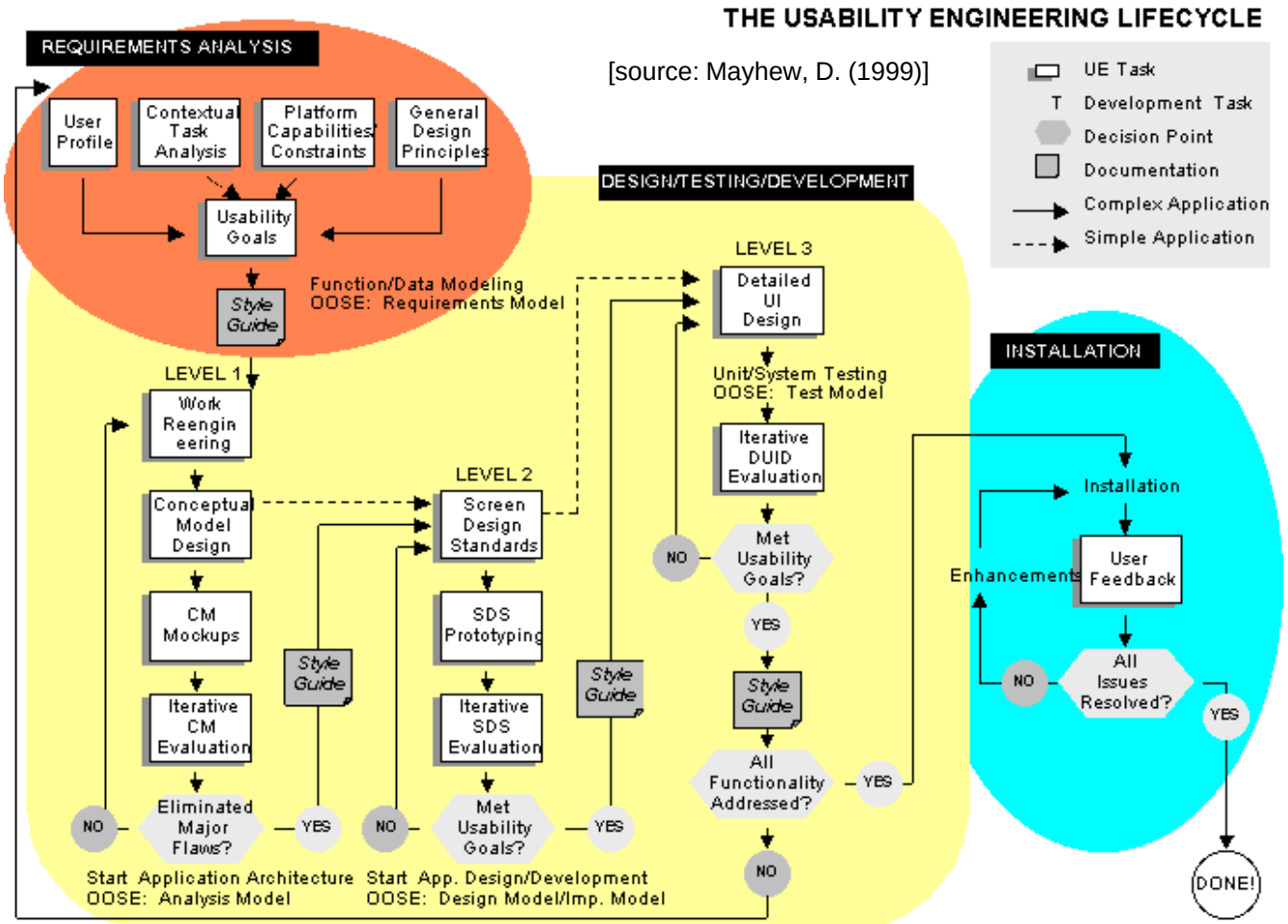
Key references/literature:

- 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]
- Horrocks, I. (1999), Constructingthe user-interface with statecharts. Addison-Wesley. [ISBN: 0201342782]
- PN:
  Reisig, W. (1992), *A Primer in Petri Net Design*. Springer. [ISBN: 0387520449]

The Lifecycle Model



The Usability Engineering Lifecycle



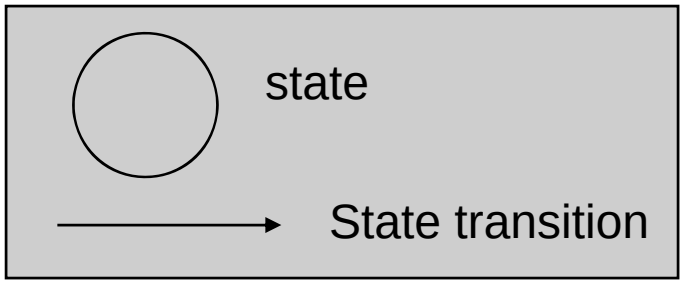
User Interaction Specification

- Many approaches/notations to specifying interaction
  - State-Transition-Diagrams (STD)
  - **Petri Nets** (PN)

- 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



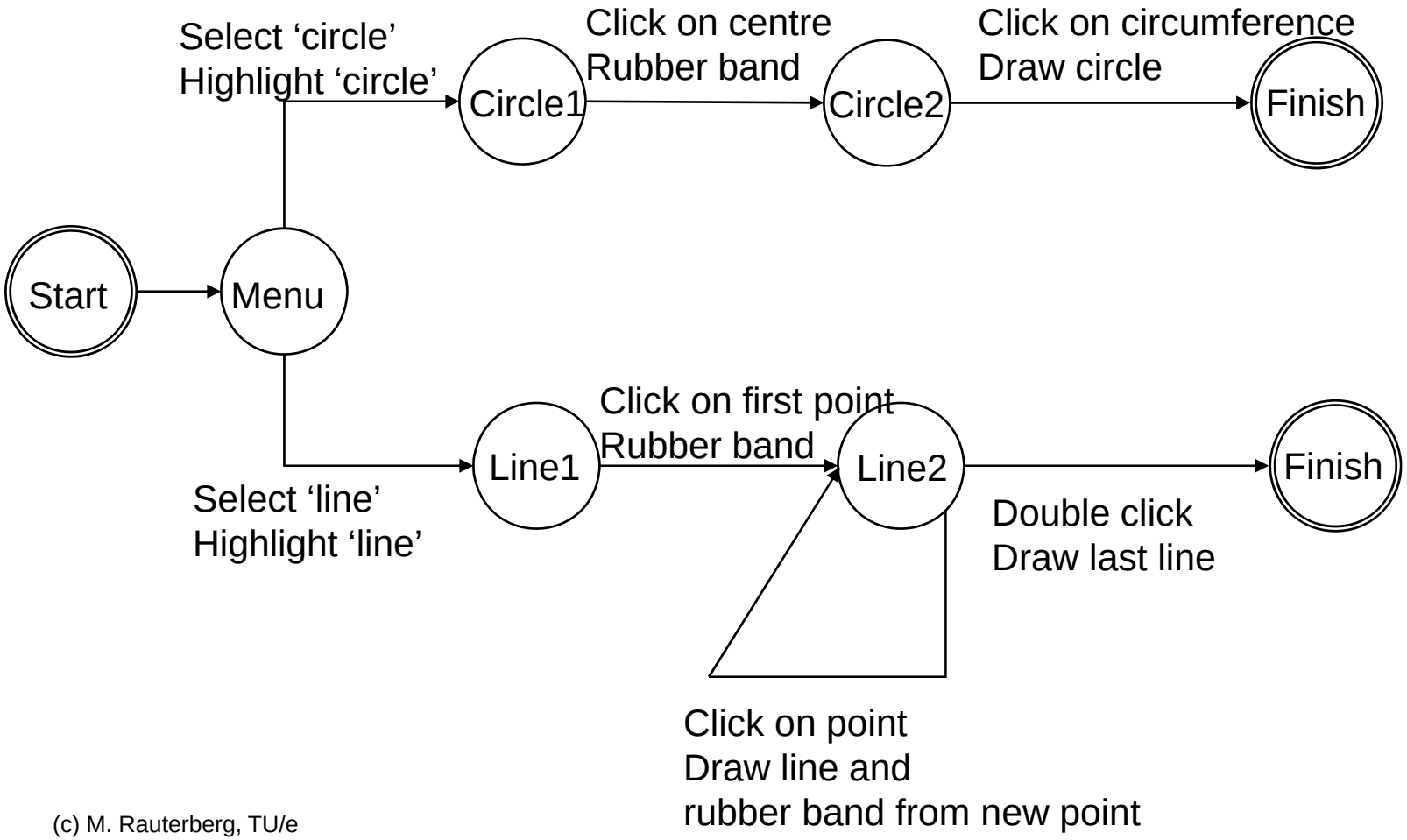
State
= set of values that describe an object (its condition/situation) at a specific moment in time

{State is determined based on the attribute values}

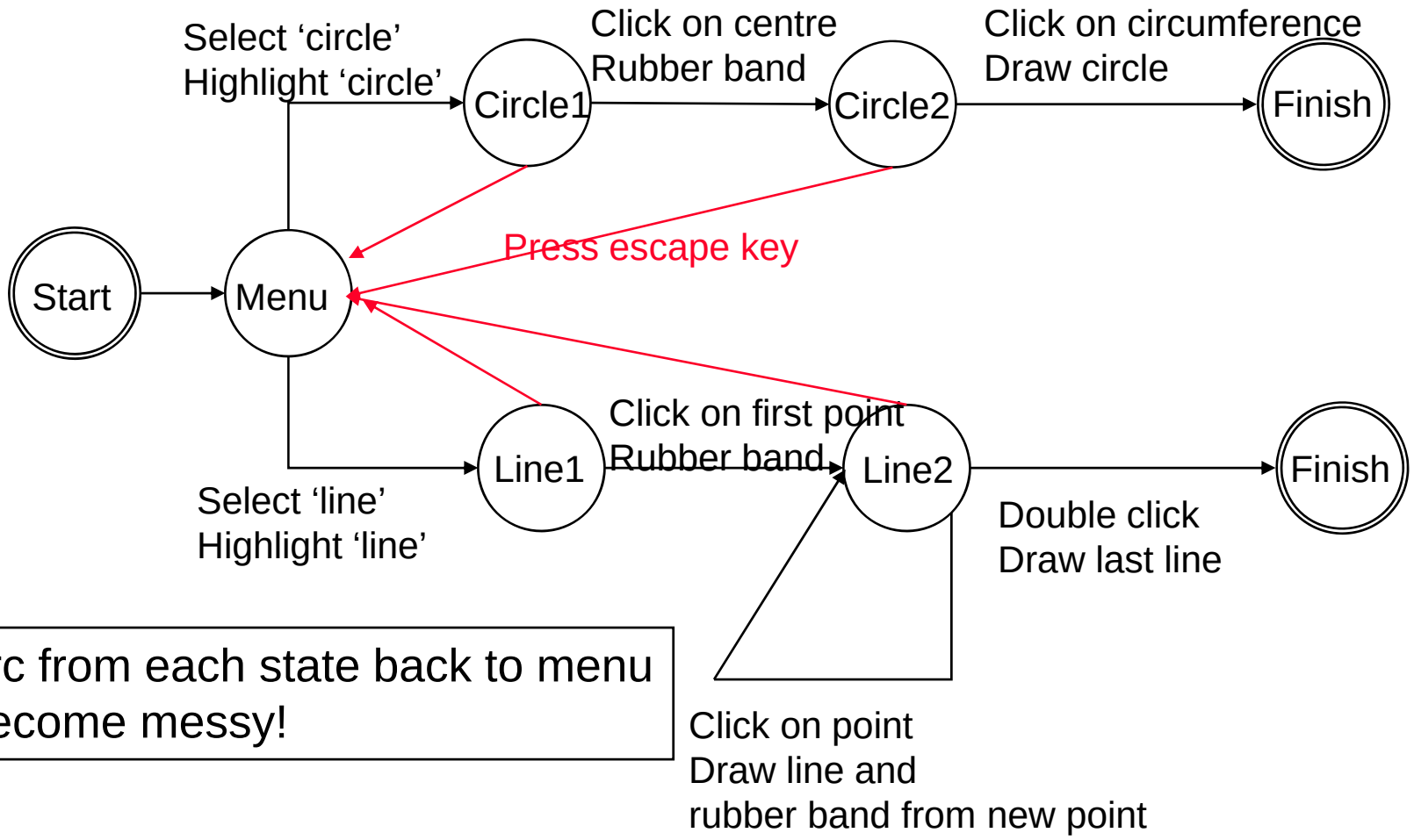
State transition
= relationship indicating a state change

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

STD Example 1: Draw Circle



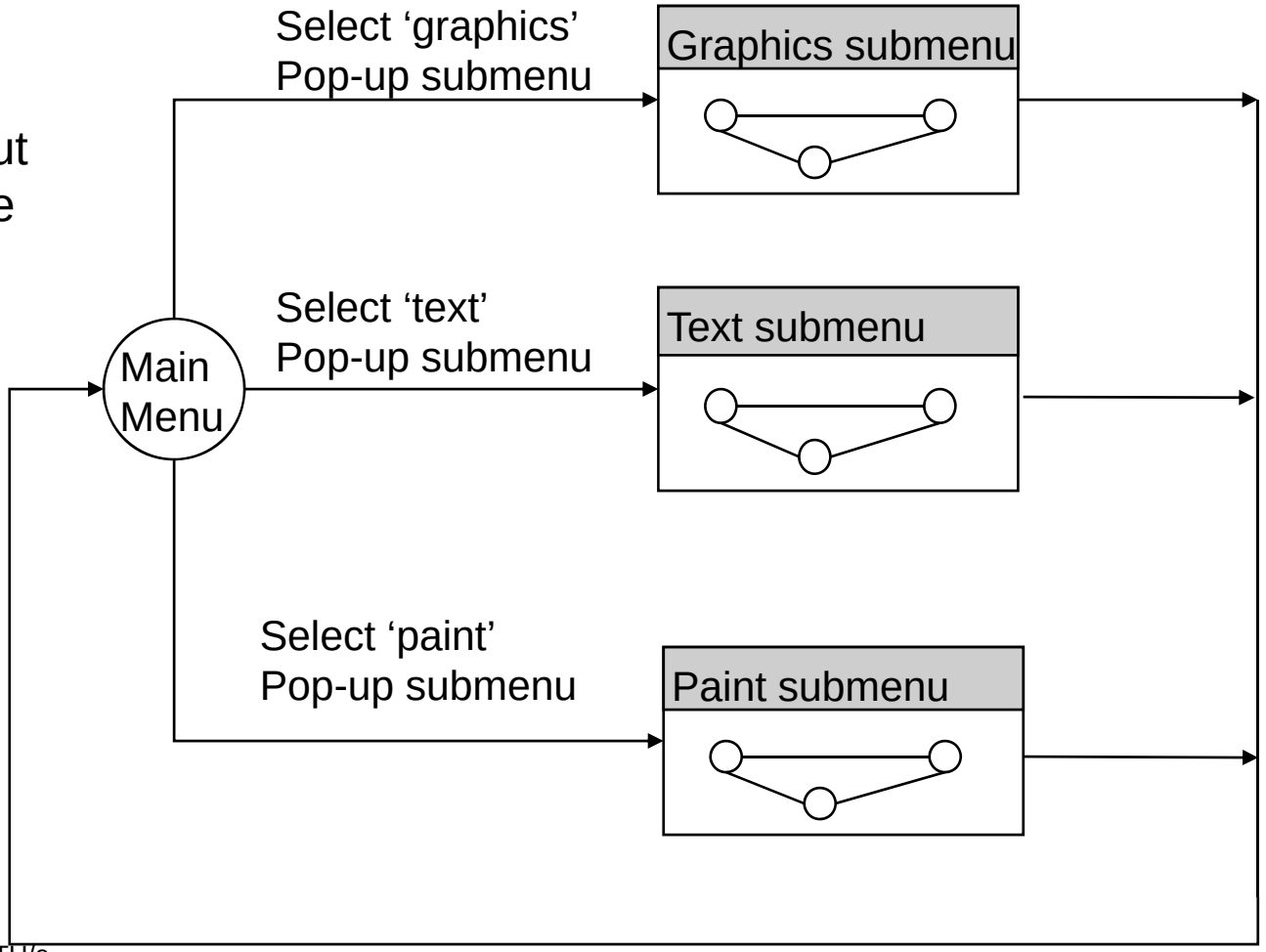
STD Example 1 (cont'd)



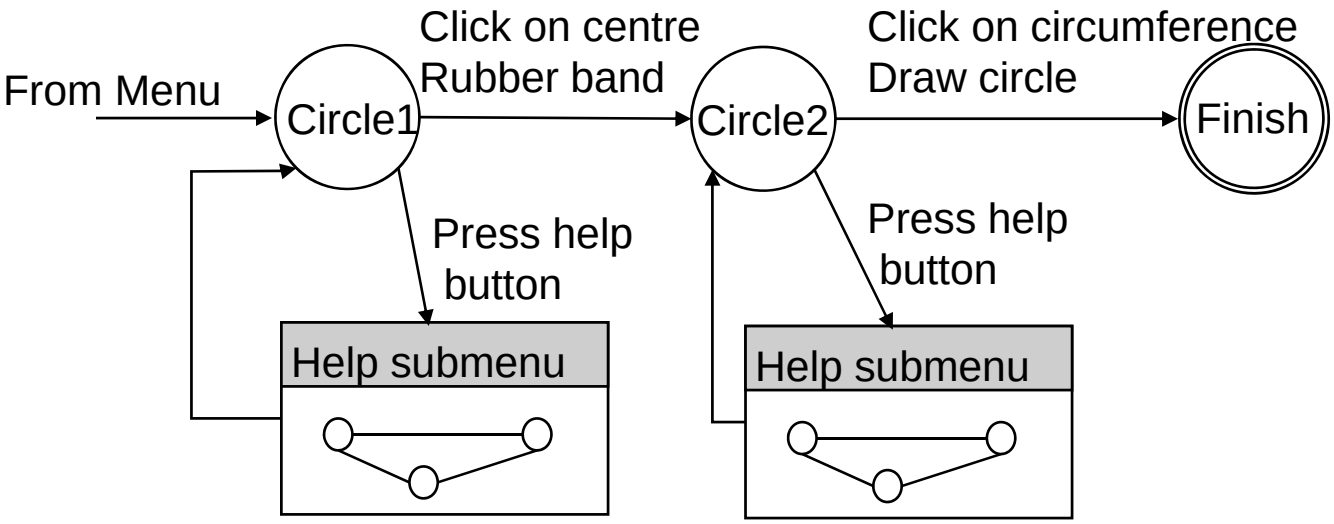
Arc from each state back to menu
Become messy!

Hierarchical STD Example 2

Not more powerful, but more simple and flexible



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 = $\{t_1, t_2, ..., t_n\}$ is a finite set of transitions

IN: (PxT)$\rightarrow$S

OUT: (TxP)$\rightarrow$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

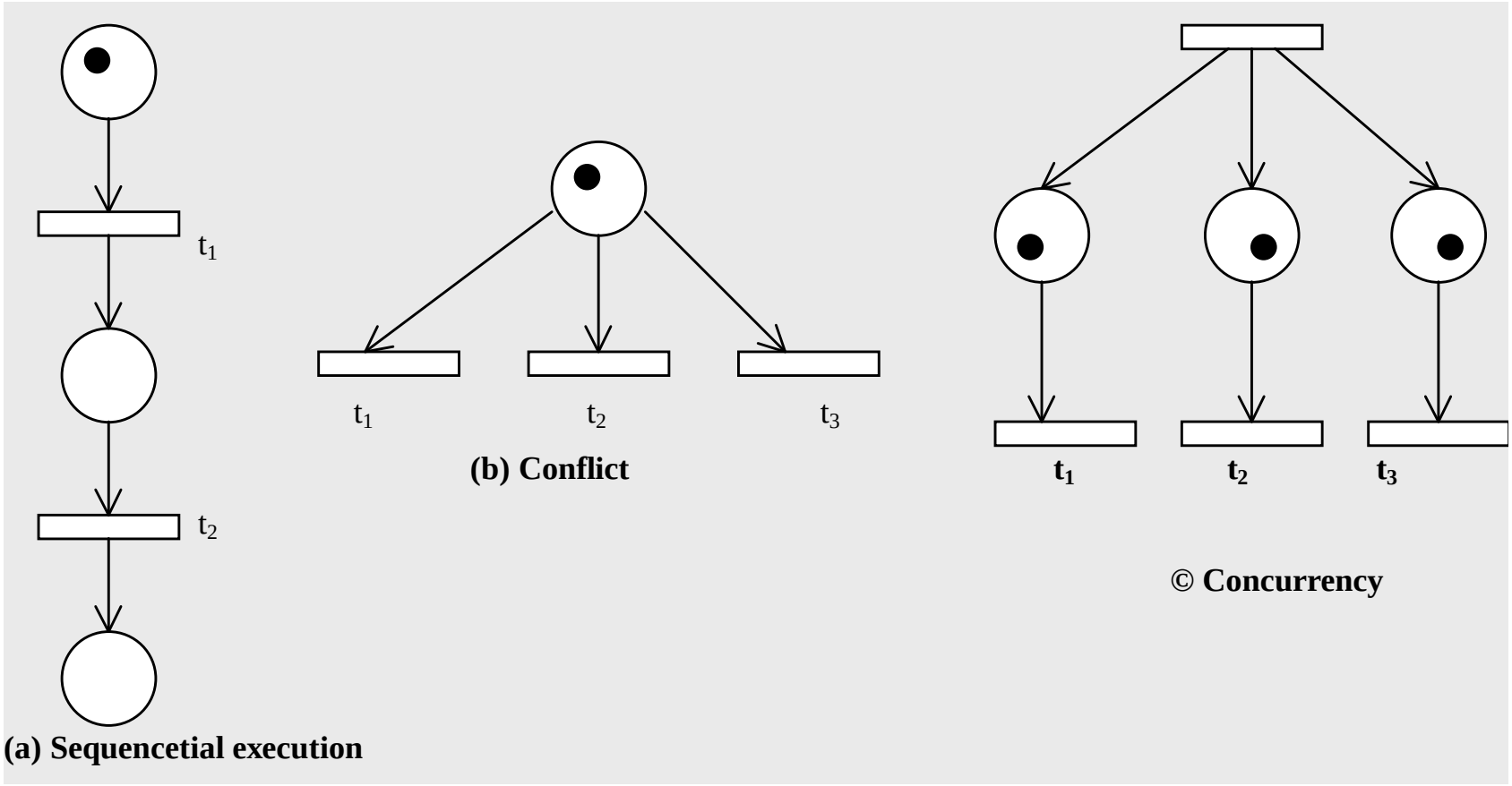
PN: Change of States (1)

- 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.

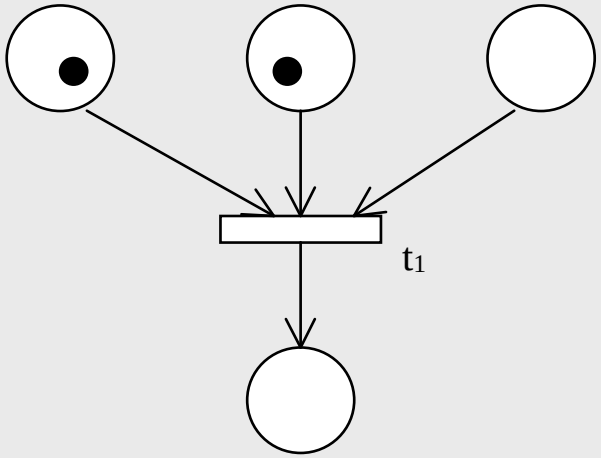
PN: Change of States (2)

- 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.

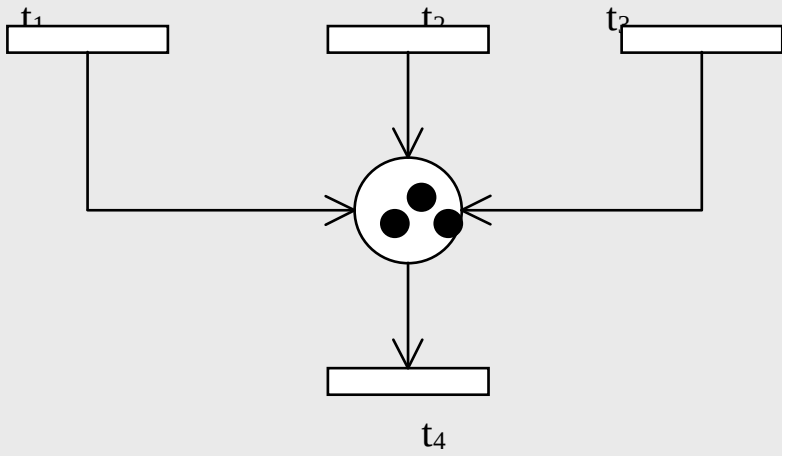
PN: basic modeling (1)



PN: basic modeling (2)

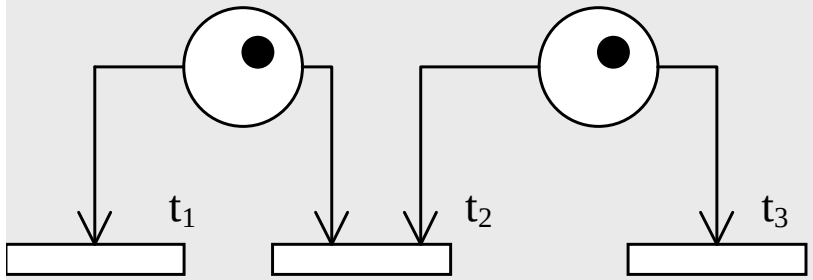


(d) Synchronation

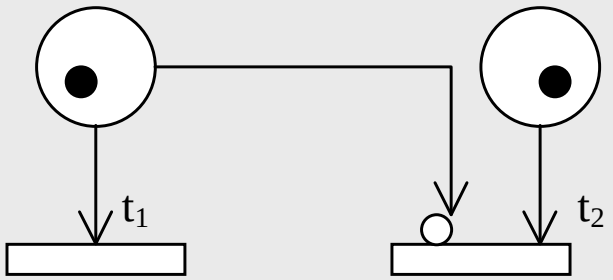


(e) Merging

PN: basic modeling (3)

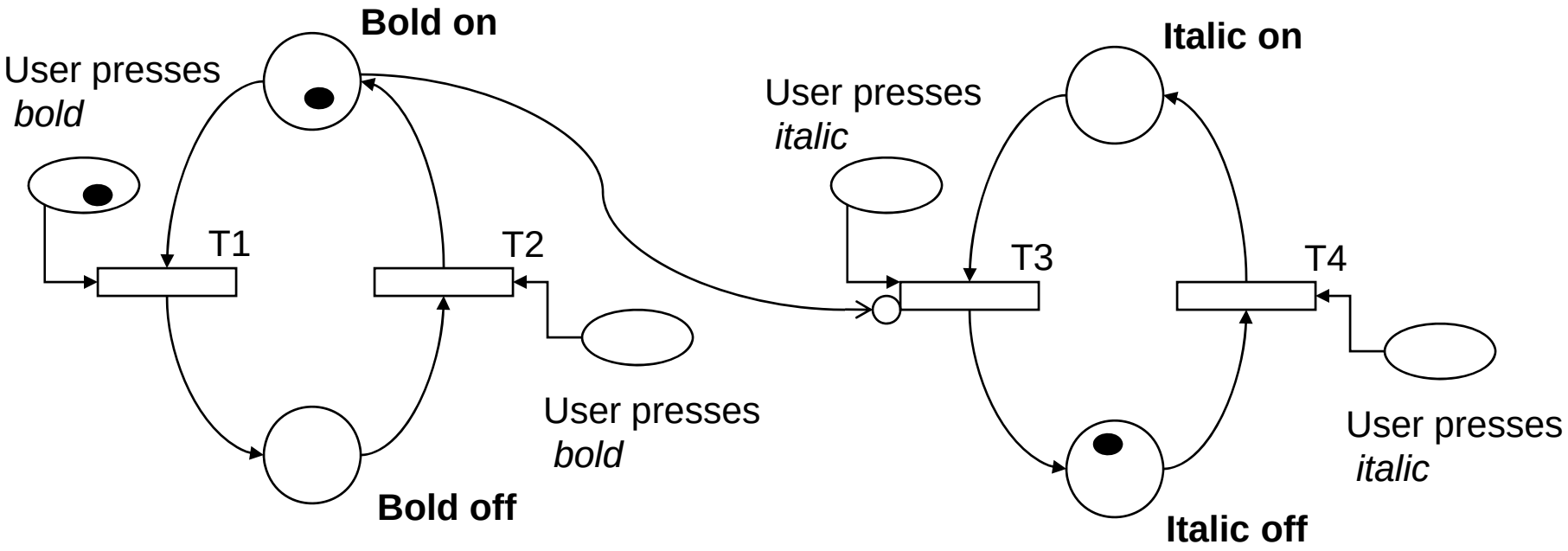


(f) Confusion



(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)