Software Engineering

Specifying Systems - Intro

何明昕 HE Mingxin, Max

Send your email to c.max@yeah.net with a subject like: *SE345-Andy: On What …*

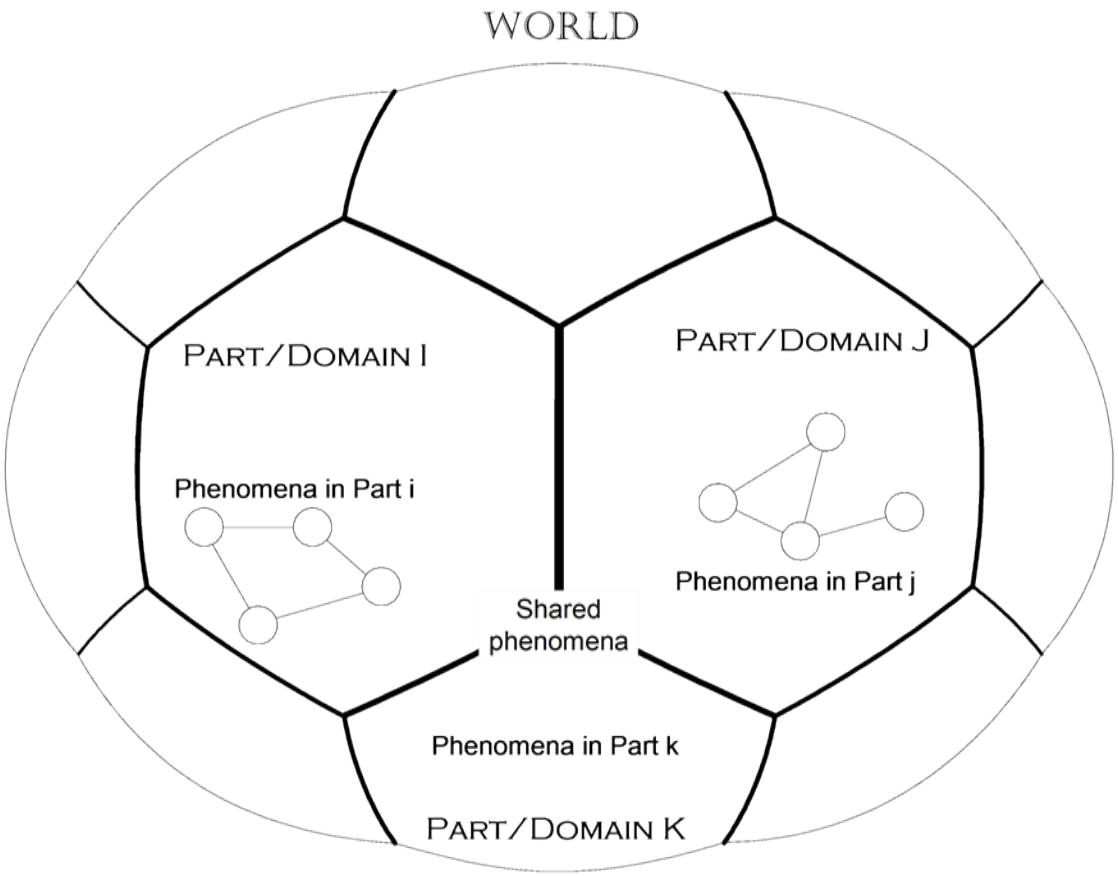
Download from c.program@yeah.net

/文件中心/网盘/SoftwareEngineering24S

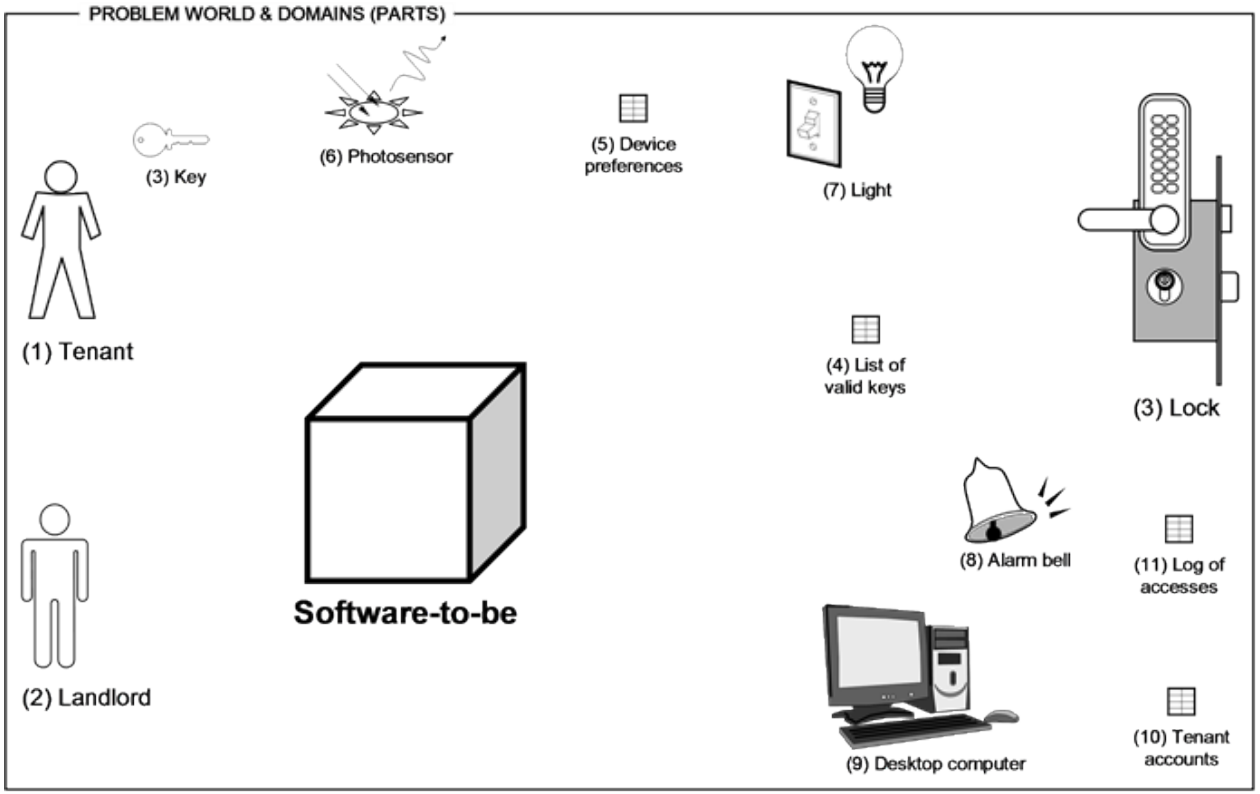
Topics

- Domains, Phenomena
- States, Events
- Context Diagrams
- Systems and System Descriptions
- Basic Formalisms for Specifications
  - Boolean Logic
  - Finite State Machines

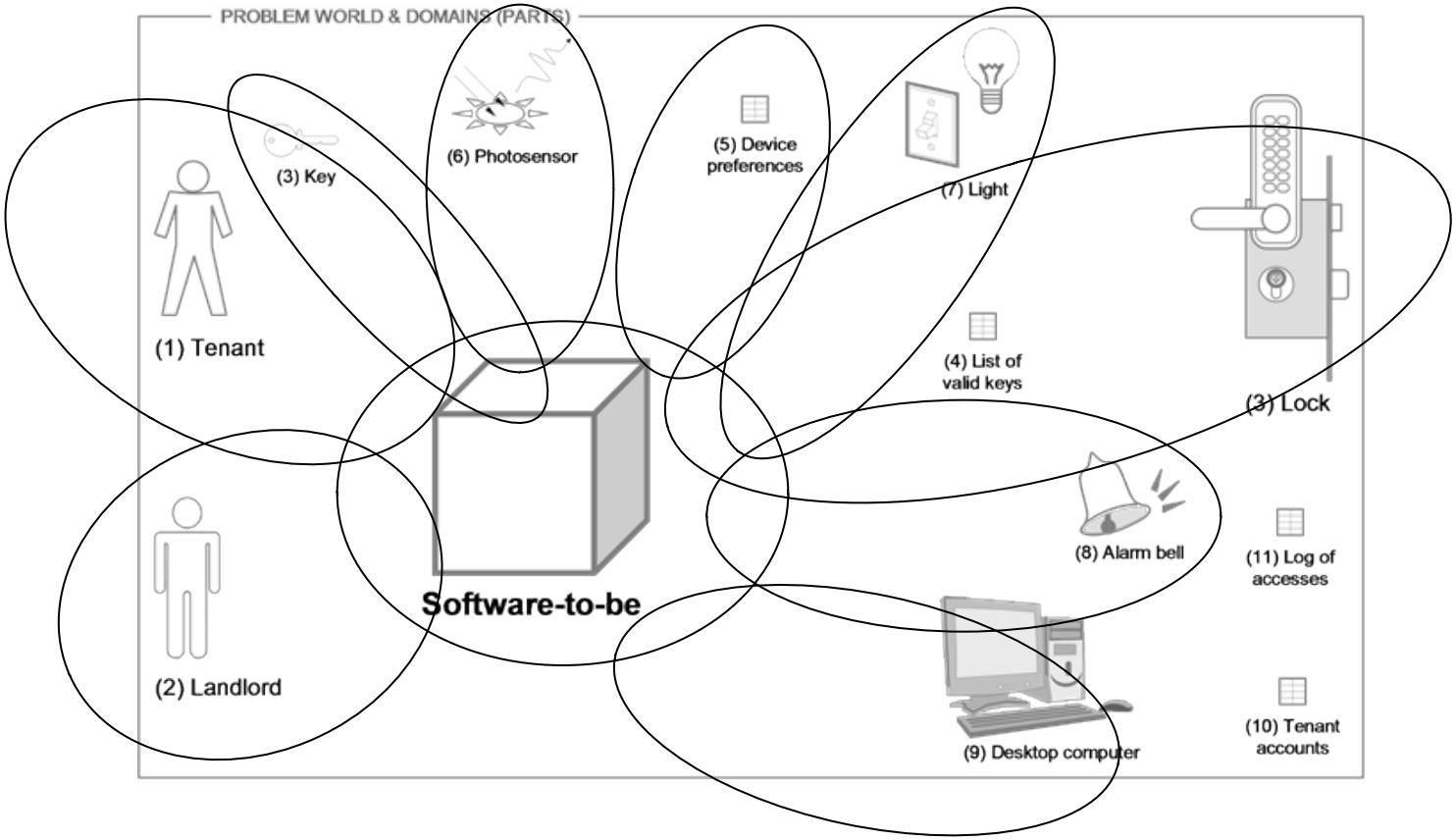
World, Parts, Phenomena



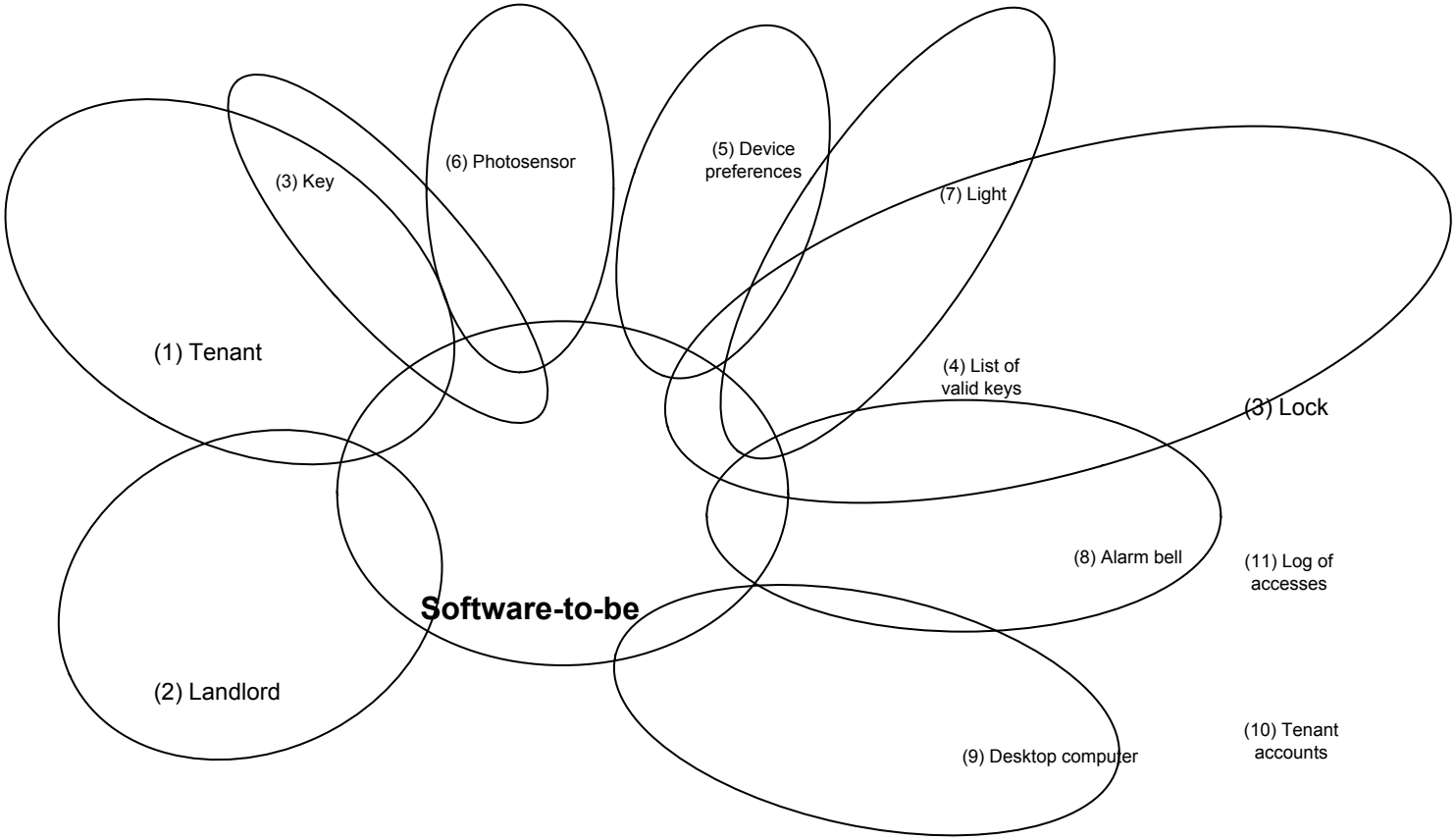
Example of a Problem Domains



Example of Problem Domains



Example of Problem Domains

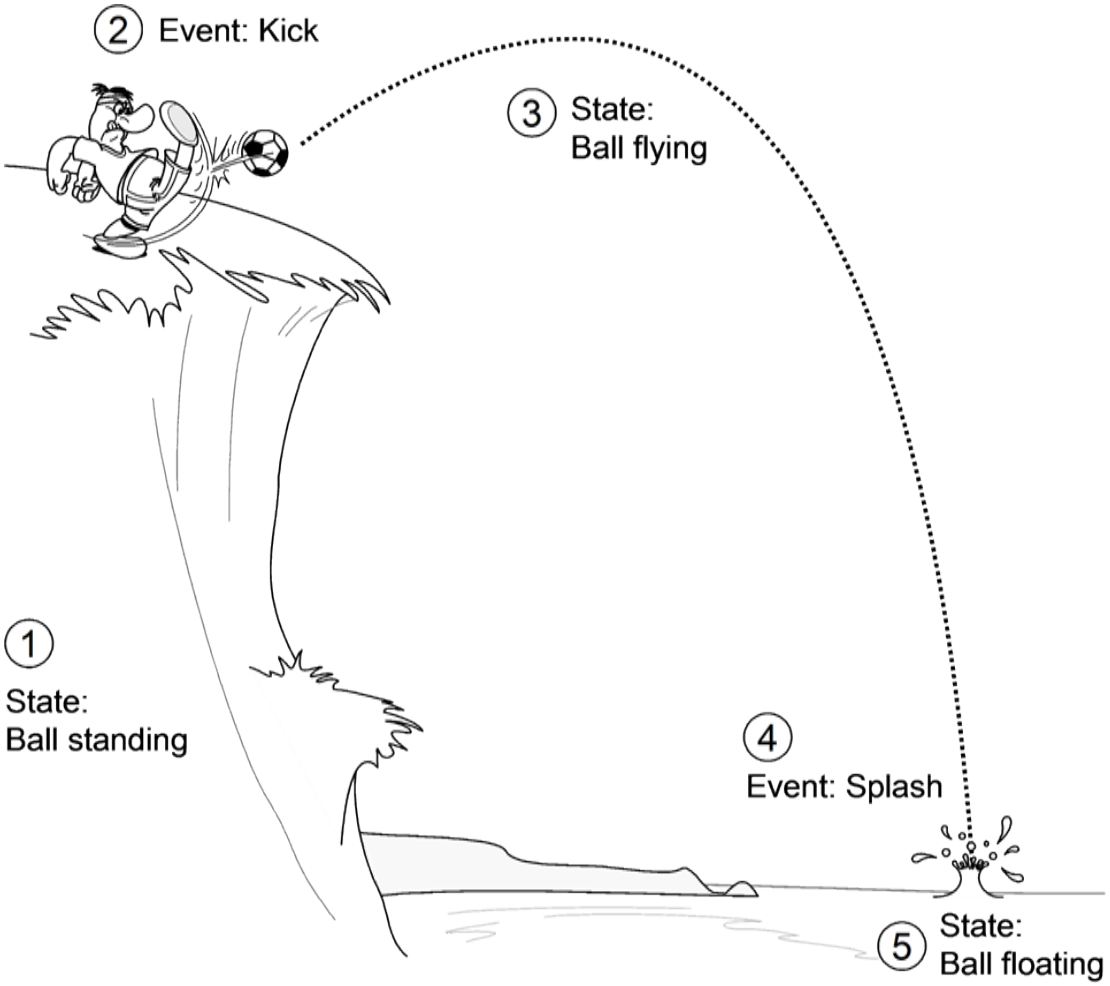


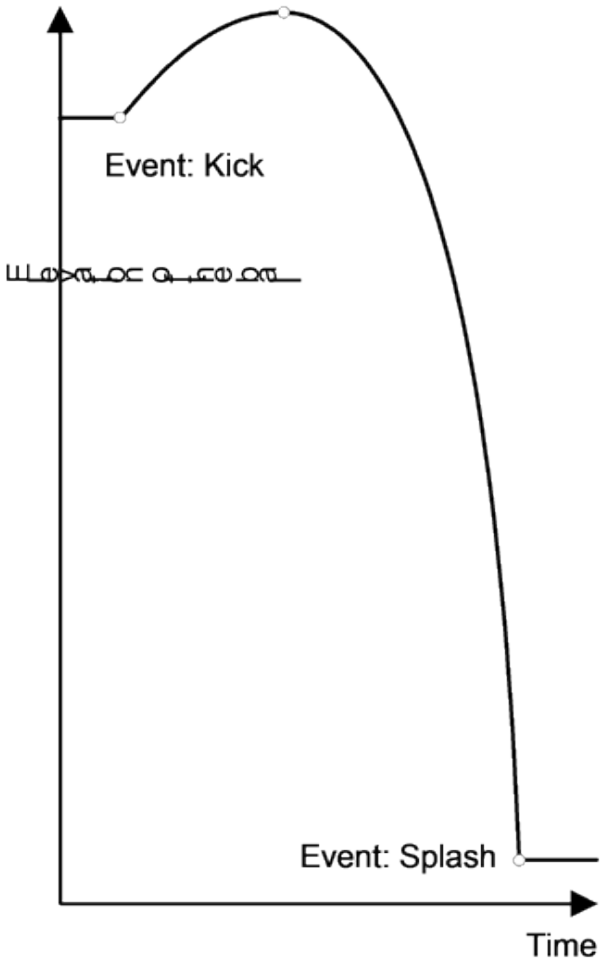
Definitions …

- A **phenomenon** is a fact, or object, or occurrence that appears or is perceived to exist
- An **event** is an individual happening, occurring at a particular point in time
  - Events are *indivisible* and *instantaneous*
- A **state** is a relation among individual entities and values, which can change over time
- Individuals are in **relation** if they share a certain characteristic
  - *RelationName*(*Individual*$_1$, …, *Individual*$_n$)

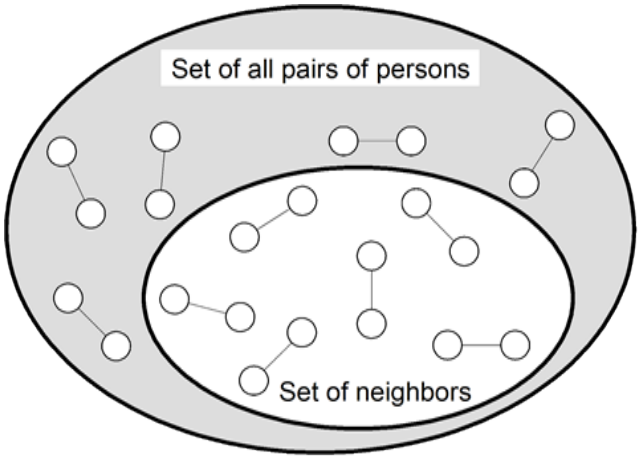
Events

Events take place at transitions between the states

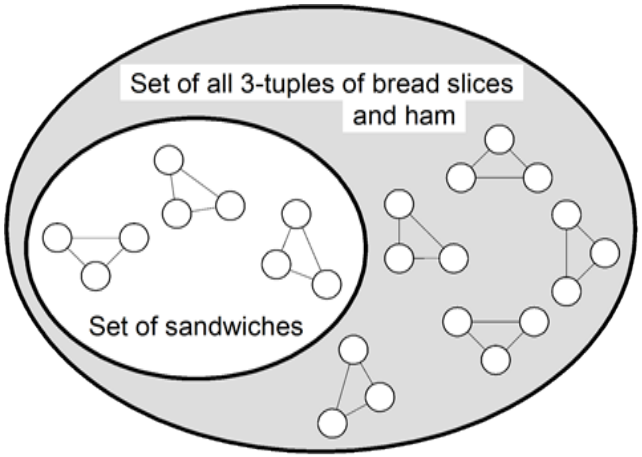




Relations: Examples



Relation: ***Neighbors*** (Person_i, Person_j)



Relation:
Sandwich (Bread-slice, Ham-slice, Bread-slice)

Example: States of a DVD Player

State 1: NotPowered (the player is not powered up)

State 2: Powered (the player is powered up)

State 3: Loaded (a disc is in the tray)

State 4: Playing

State 1: NotPoweredEmpty (the player is not powered up *and* contains no disc)

State 2: NotPoweredLoaded (the player is not powered up *and* a disc is in the tray)

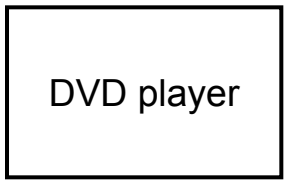
State 3: PoweredEmpty (the player is powered up *and* contains no disc)

State 4: PoweredLoaded (the player is powered up *and* a disc is in the tray)

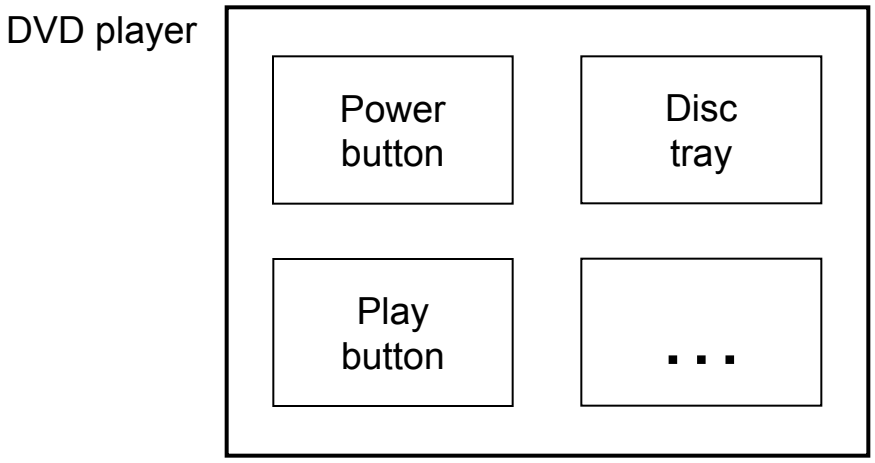
State 5: Playing

Different Abstractions

(Level of detail)



Atomic object



Object composed of parts

Example: States of a DVD Player

System Part (Object)	State Relations
Power button	{Off, On}
Disc tray	{Empty, Loaded}
Play button	{Off, On}
…	…

State Variables

State variable = a physical *part* or an *attribute* of an object

State Variable	State Relations
Door lock	{Disarmed, Armed}
Light bulb	{Unlit, Lit}
Counter of failed attempts	{0, 1, …, maxNumOfAttempts}
Auto-lock timer	{0, 1, …, autoLockInterval}

Hidden States

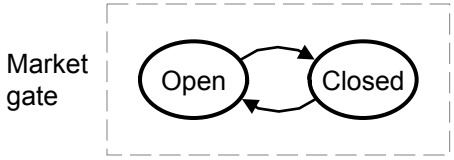
Observable state:
apple's appearance

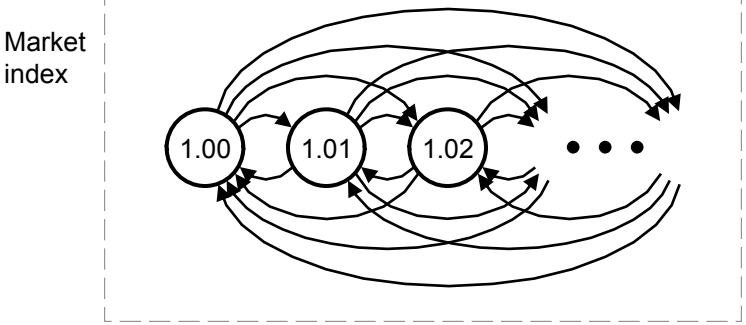
Hidden state:
contains a worm

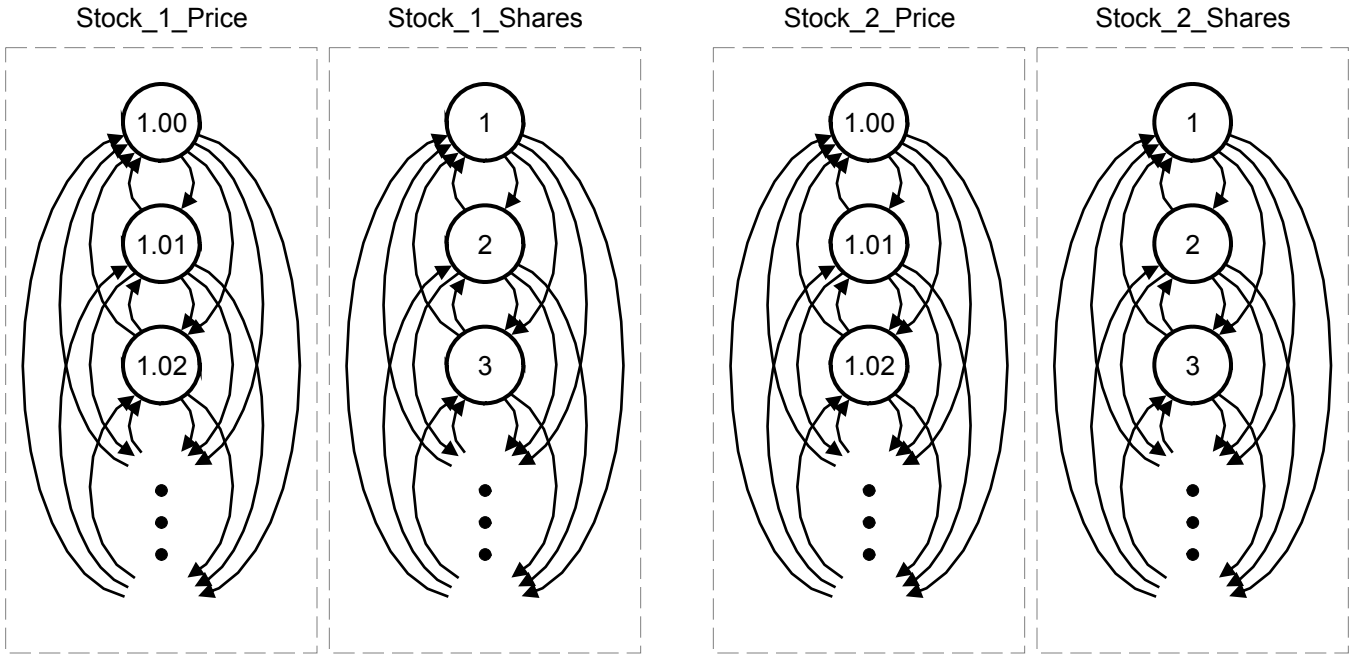
Goal:

Find the likelihood of different hidden states, for given observable states

States Example: Stock Market







(*prices & number of shares for all listed stocks*)

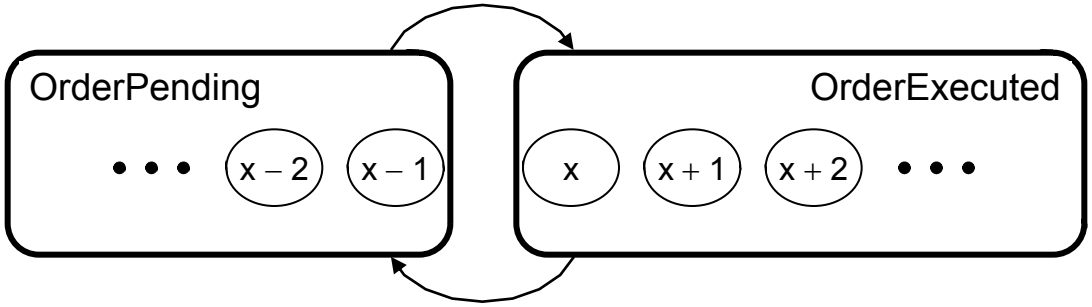
Defining States

- *CountingDown*(Timer) $\triangleq$
  The relation *Equals*(Timer, τ) holds true for τ decreasing with time

- *Idle*(Timer) $\triangleq$
  The relation *Equals*(Timer, τ) holds true for τ remaining constant with time

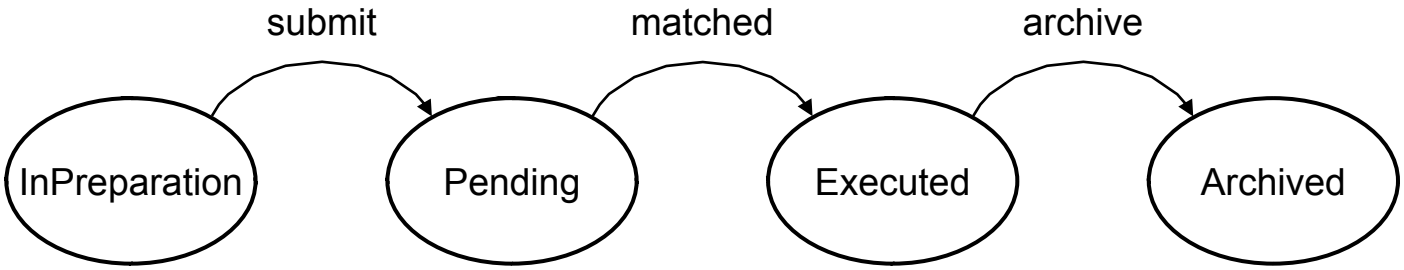
Microstates and Macrostates

Microstates representing the number of offered shares are aggregated:



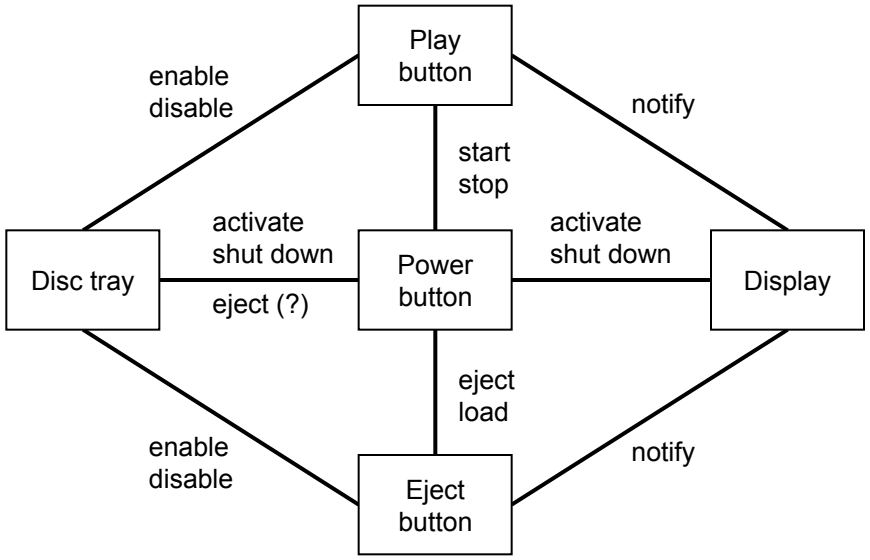
Events

Events marking transitions between the states of a trading order:

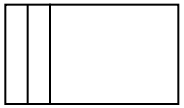


Event	Description
trade	Causes transition between stock states *Buy*, *Sell*, or *Hold*
submit	Causes transition between trading-order states *InPreparation* → *OrderPending*
matched	Causes transition between trading-order states *OrderPending* → *OrderExecuted*
…	…
…	…

Context Diagram: DVD Player



Context diagram symbols:

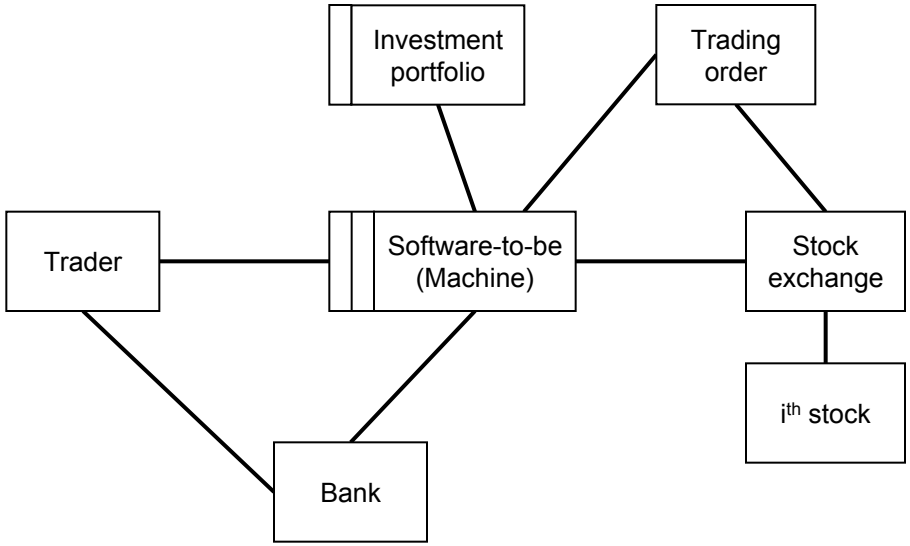


A box with a double stripe is a *software-to-be domain* (or, machine domain)

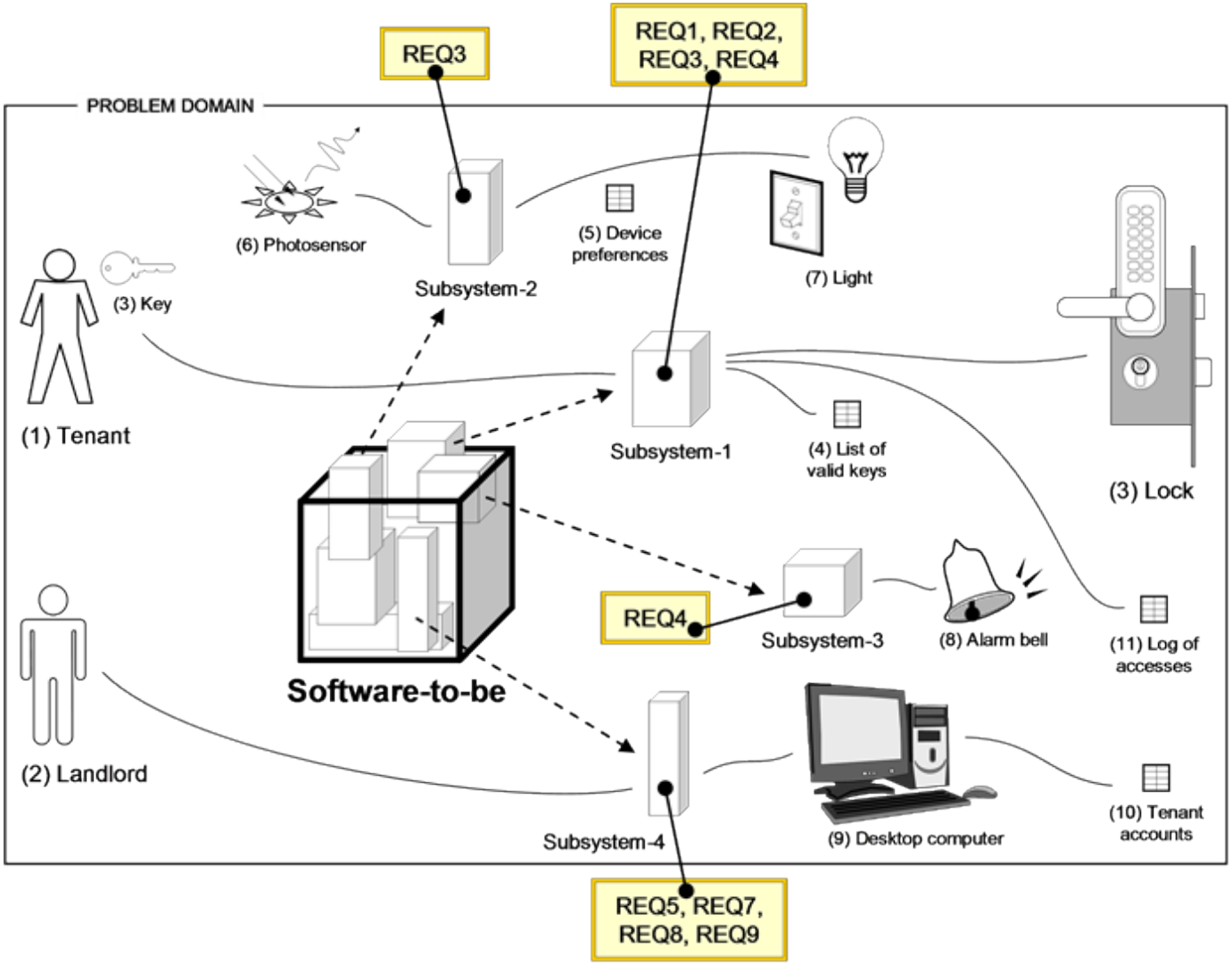
A box with a single stripe is a *designed domain*

A box with no stripe is a *given domain*

Context Diagram: Stock Trading

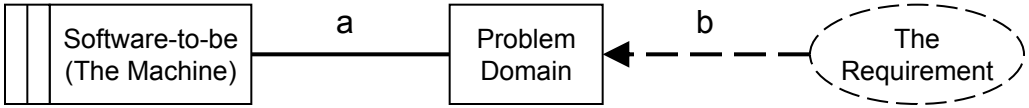


Problem Decomposition

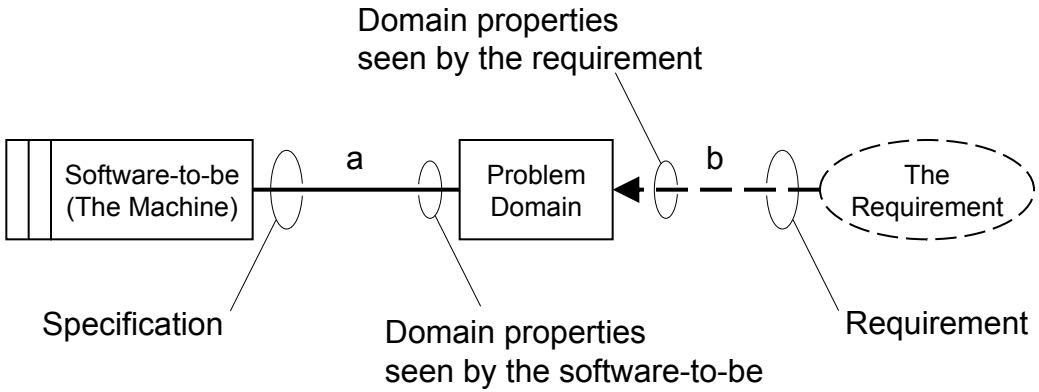


Machine and Problem Domain

(a)



(b)



a: specification interface phenomena
b: requirement interface phenomena

Boolean Logic

Propositional Logic			
$\wedge$	conjunction (p and q)	$\Rightarrow$	implication (if p then q)
$\vee$	disjunction (p or q)	$\Leftrightarrow$	biconditional (p if and only if q)
$\neg$	negation (not p)	$\equiv$	equivalence (p is equivalent to q)
Predicate Logic (extends propositional logic with two quantifiers)			
$\forall$	universal quantification (for all x, $P(x)$)		
$\exists$	existential quantification (there exists x, $P(x)$)		

Example: From Req't to Propositions

Label	Declarative sentence (not necessarily a proposition!)
a	The investor can register with the system
b	The email address entered by the investor exists in real world
c	The email address entered by the investor is external to our website
d	The login ID entered by the investor is unique
e	The password entered by the investor conforms to the guidelines
f	The investor enters his/her first and last name, and other demographic info
g	Registration is successful
h	Account with zero balance is set up for the investor

REQ1 represented as a set of propositions
a
$(\forall\ email)(\forall\ id)(\forall\ pwd)\ [B(email) \wedge C(email) \wedge D(id) \wedge E(pwd) \Rightarrow g]$
f
$g \Rightarrow h$

Example: From Req't to Propositions

Label	Propositions (partial list)
m	The action specified by the investor is "buy"
n	The investor specified the upper bound of the "buy" price
o	The investor specified the lower bound of the "sell" price

Label	Propositions (they complete the above list)
p	The investor requests to place a market order
q	The investor is shown a blank ticket where the trade can be specified (action, symbol, etc.)
r	The most recently retrieved indicative price is shown in the currently open order ticket
s	The symbol SYM specified by the investor is a valid ticker symbol
t	The current indicative price that is obtained from the exchange
u	The system executes the trade
v	The system calculates the player's account new balance
w	The system issues a confirmation about the outcome of the transaction
x	The system archives the transaction

REQ2 represented as a set of propositions

$p \Rightarrow q \wedge r$

s

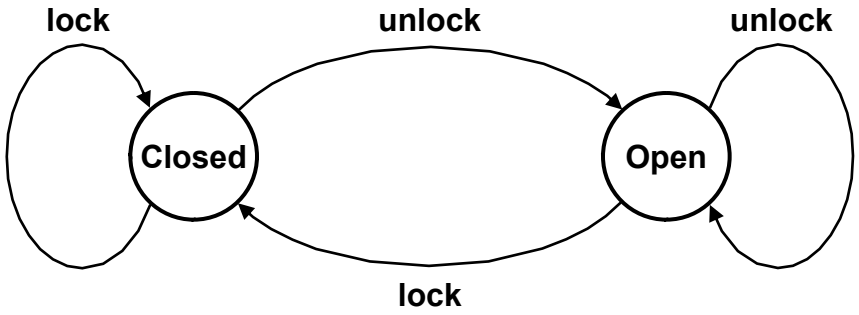
$\boldsymbol{y} = v \wedge \{\neg(n \vee o) \vee [(o \wedge p \vee \neg o \wedge q) \wedge (\exists\ \mathrm{IP})(\mathrm{LB} \leq \mathrm{IP} \leq \mathrm{UB})]\}$

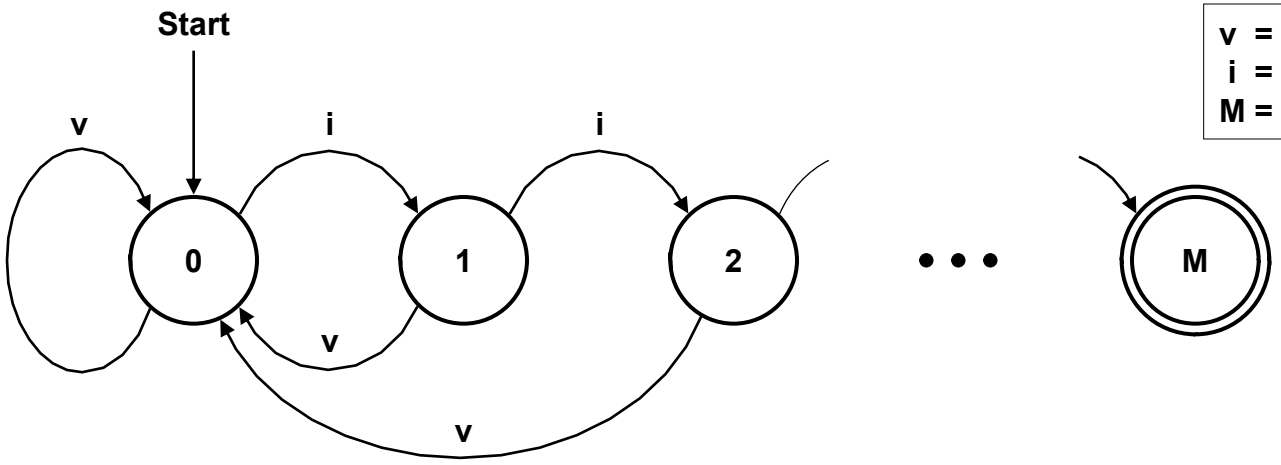
$\boldsymbol{z} = \neg m \vee \{[\neg n \wedge (\mathrm{VOL} \times \mathrm{IP} \leq \mathrm{BAL})] \vee [n \wedge (\mathrm{VOL} \times \mathrm{UB} \leq \mathrm{BAL})]\}$

$y \wedge z \Rightarrow u$

$u \Rightarrow v \wedge w \wedge x$

FSM State Transition Diagram

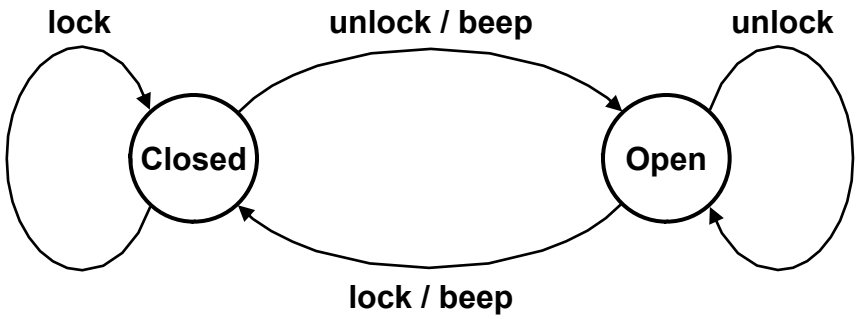




v = (input-key ∈ Valid-keys)
i = (input-key ∉ Valid-keys)
M = maxNumOfAttempts

FSMs with Outputs

(a)



(b)