CS350 Lecture 3-2 Requirements Engineering

Doo-Hwan Bae

bae@se.kaist.ac.kr

So far,

- Software/System modeling
  - Use-case diagram: Functional view from user's point of view
  - Class diagram: static structure
  - Sequence diagram: dynamic behavior
- Q: Why do we need software modeling?
- Those models will eventually map into programming constructs
- What are major programming constructs?
  - Most of programming languages have them(similar to domain objects!)

Programming Constructs to Modeling Constructs

- Why modeling?
  - Managing C_____?
- Programming constructs
  - Sequence
  - Selection
  - Iteration
- Modeling constructs
  - Flow chart
  - Many ‘Boxes’ & ‘Arrow’ notations

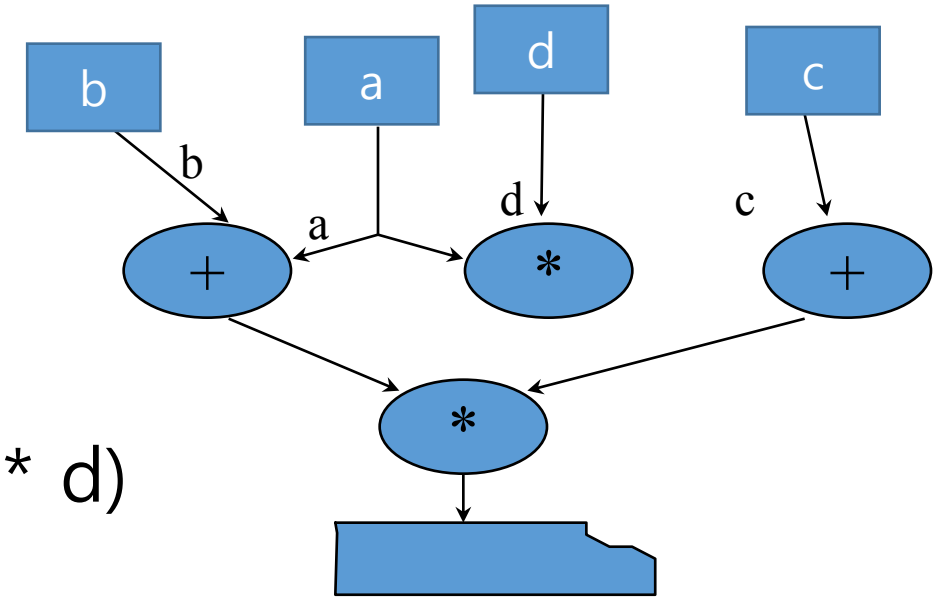
Structured Analysis

- Consists of
  - -- Data Flow Diagram
  - -- Data Dictionary
  - -- Process Specification

Data Flow Diagram

- For business data processing, where control aspects may not play significant role.
- Components
  - Function, or action:
  - Data store:
  - External entity:
  - Data flow:
- Conceptual Example:

(a + b) * (c + a * d)



Data Flow Diagram(Cont.)

- Example:

  Automatic Teller Machine(ATM)

Data Flow Diagram (Cont.)

- Drawback:
  - Cannot give a precise and detailed definition.
  - Cannot simulate the system with DFDs.
  - Extensions
    - Integrate with different description, i.e., STD
    - Augment DFD with control flow, Ward&Mellor
    - Revise DFD, to make it fully formal.

Data Dictionary(1/2)

- Contains formal definitions of all the data items shown in DFD
- Provides precise detail concerning the data entities
- Format

  -- Alphabetically ordered list of entries

  -- Consists of formal definitions and verbal descriptions
- Notations:

  data-element-name = expression

  + : concatenation

  | : alternatives

  ' ' : literals

  [ ] : options

  { } : repetitions

Data Dictionary(2/2)

- Example

Payment = US-dollars | Korean-Wons

US-dollars = '$' + dollar-amt + '.' + cent-amt

* Payment is the income received from subscribers. It is entered as dollars and cents.

Korean-Wons = 'w' + ...

dollar-amt = {digit}

cent-amt = digit + digit

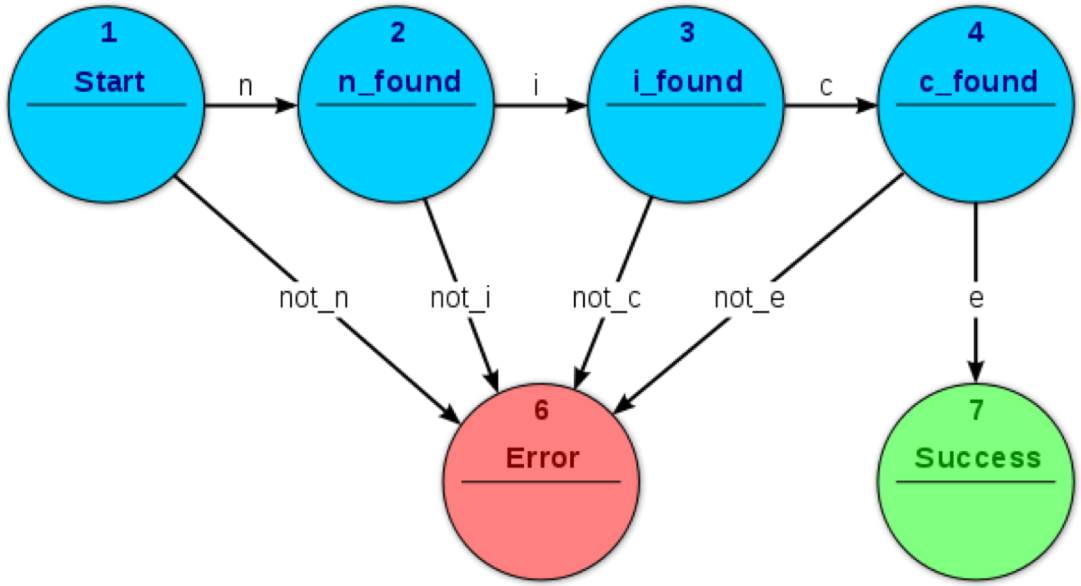
digit = 0|1|2|.....|9

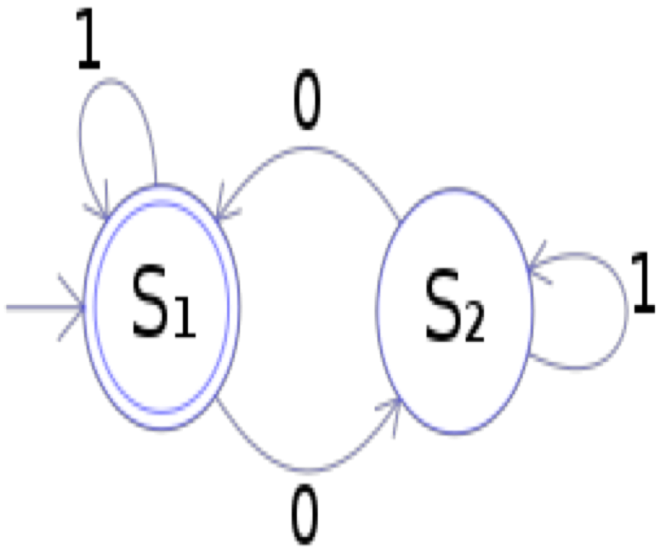
Finite State Machine (FSM) (1/4)

- Emphasize control aspects of the application.
- Components
  - A finite set of states
  - A finite set of input
  - A transition function

Finite State Machine (FSM) (2/4)

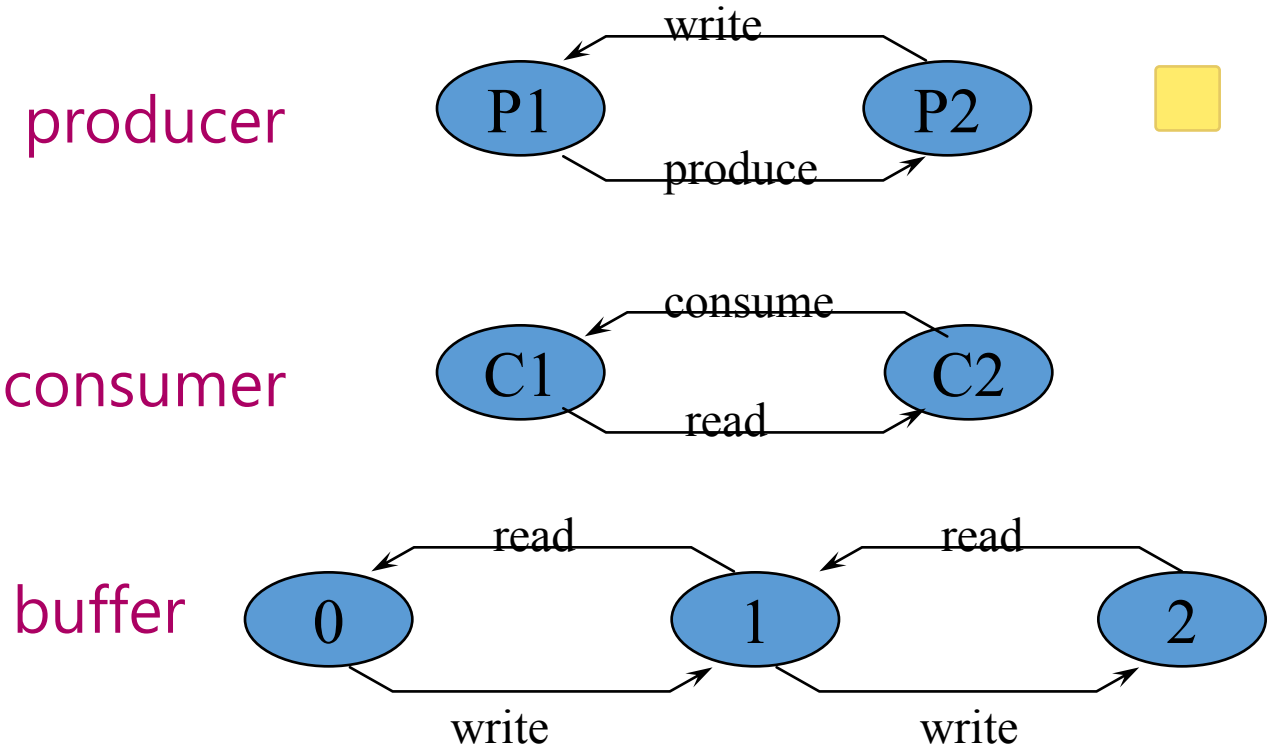
- Some examples: What it does?





FSM (3/4)

- Example: Producer/Consumer problem.



FSM (4/4)

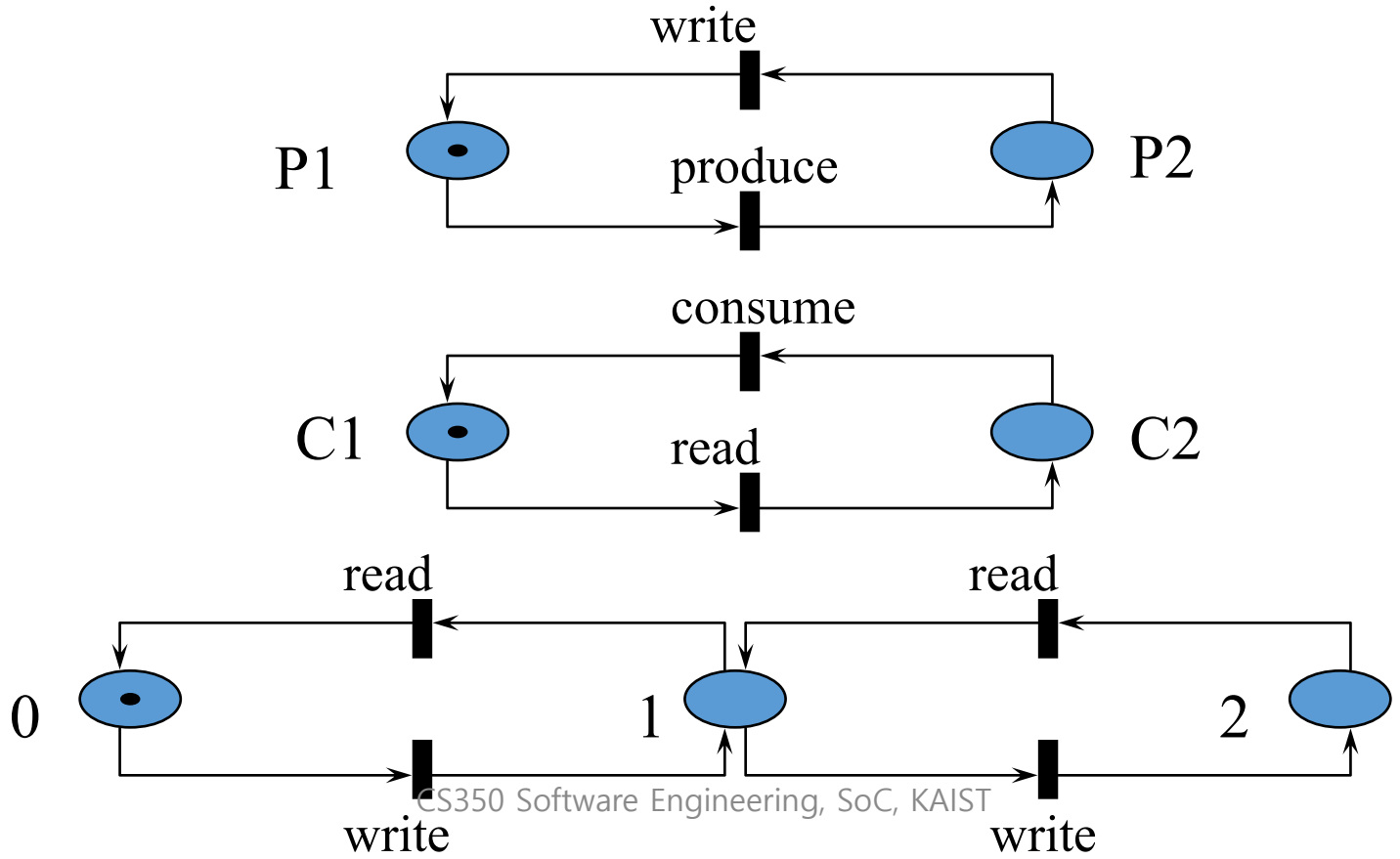
- Drawback;
  - Computation power is limited.
  - (Not easy to specify data info.)
  - The cardinality of the state space growth
  - Difficult to specify asynchronous activities.

Petri Net (1/5)

- Specify asynchronous systems
- Components
  - A finite set of places:
  - A finite set of transitions:
  - A finite set of arrows:
    - from places to transitions,
    - from transitions to places.
- Tokens

Petri Net (2/5)

- Example: Producer/Consumer problem.



Petri Net (3/5)

- Try to model concepts of the following:
  - 3-version program and 2 out of 3 policy
  - Resource allocation
  - Deadlock
  - Starvation

Petri Net (4/5)

- Modeling examples
  - Finite state machines.
  - Parallel activities.
  - Data flow computations.
  - Communication protocols.
  - Synchronous control.
  - Producer-consumer problem with priority.
  - Formal languages.

Petri Net (5/5)

- Drawback:
  - Too simple to specify complex systems.
  - Not possible to specify a selection policy.
  - Timing issues for real-time systems.
- Extensions:
  - Assigning values to tokens.
  - Specifying scheduling policies.
  - Incorporate timing constraints.

Nonfunctional Requirements

- User interface & human factors
- Documentation
- Hardware consideration
- Performance characterization
- System interfacing
- Quality issues
- System modification
- Physical environments
- Security issues
- Resources and management issues

Summary

- Remember that in the real world, no one will give you the exact, complete, consistent, and unambiguous requirements for software development.
- Requirements changes are inevitable
- In addition to functional requirements, nonfunctional requirements are also important for value-added software development.

Exercise

- Think about Petri-net modeling of the following:

* Draw a Dining philosophers' problem with four philosophers and four forks. A philosopher needs two forks to eat.