投票　最多可选1项

设置

Have you read the paper list of ICSE2021?

A　Yes

B　No

提交

Software Engineering Review

Ning Li

April, 2022

1. Overall Content

Part 0 Introduction

Ch1:The Nature of Software *

Ch2: Software Engineering *

Part1 Process

Ch3: Software Process Structure **

Ch4: Process Models ****

Ch5: Agile Development ***

Part2 Modeling

Ch8: Understanding Requirements *

Ch9: Requirements Modeling: Scenario-based ****

Ch10: Requirements Modeling: Class-based ****

Ch11: Requirements Modeling: Behavior, Patterns, and WebApps **

Ch12：Design Concepts **

Ch13：Architectural Design ***

Ch14：Component-Level Design ****

Ch15：User Interface Design ****

Ch15' Writing code (supplement) ***

1. Overall Content

Part 3 Quality Management

Ch19：Quality Concepts **

Ch21：Software Quality Assurance ***

Ch22：Software Testing Strategies ***

Ch23：Testing Conventional Applications ****

Ch24：Testing Object-Oriented Applications ***

Ch29：Software Configuration Management **

Part 4: Managing Software Projects

Ch31: Project Management Concepts **

Ch32: Process and Project Metrics ***

Ch33: Estimation for Software Projects ***

Ch34: Project Scheduling ***

Ch36: Maintenance and Reengineering *

2. Content - part0 Introduction

- Ch1:The Nature of Software (*)
  1. Terminology： fault, error, failure
  2. Software vs. Program?
  3. Wear vs. Deterioration (Software and Hardware)
  4. Nature: Changing (traditional software, web, mobile app, cloud , product line)

- Ch2: Software Engineering (*)
  1. Definition of SE： (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).
  2. Layered Technology: tools, methods, process model, a “quality” focus.
  3. Process Framework：
     - Activities: Communication, Planning, Modeling, Construction, Deployment
     - Umbrella Activities:
       - Software project tracking and control, risk management
       - Software quality assurance, technical review, measurement
       - Software configuration management
       - Reusability management, work product preparation and production

2. Content - part1 Process

- Ch3: Software Process Structure (**)
    1. generic process model: activity -> action -> task
    2. Process flow : linear process flow; iterative process flow;
       evolutionary process flow; parallel process flow.

- Ch4: Process Models (****)
    1. Waterfall Models (V models)
    2. Incremental Process Models
    3. Evolutionary Process Models (Prototyping and Spiral models)
    4. Concurrent Models
    5. The Unified Process (UP)
       - Inception, Elaboration, Construction, Transition, Production

- Ch5: Agile Development (***)
    1. manifesto for agile
    2. typical agile development: XP(user stories, KIS priciple, pair programming)
    3. Scrum, Kanban

2. Content - part2 Modeling - Requirment (1)

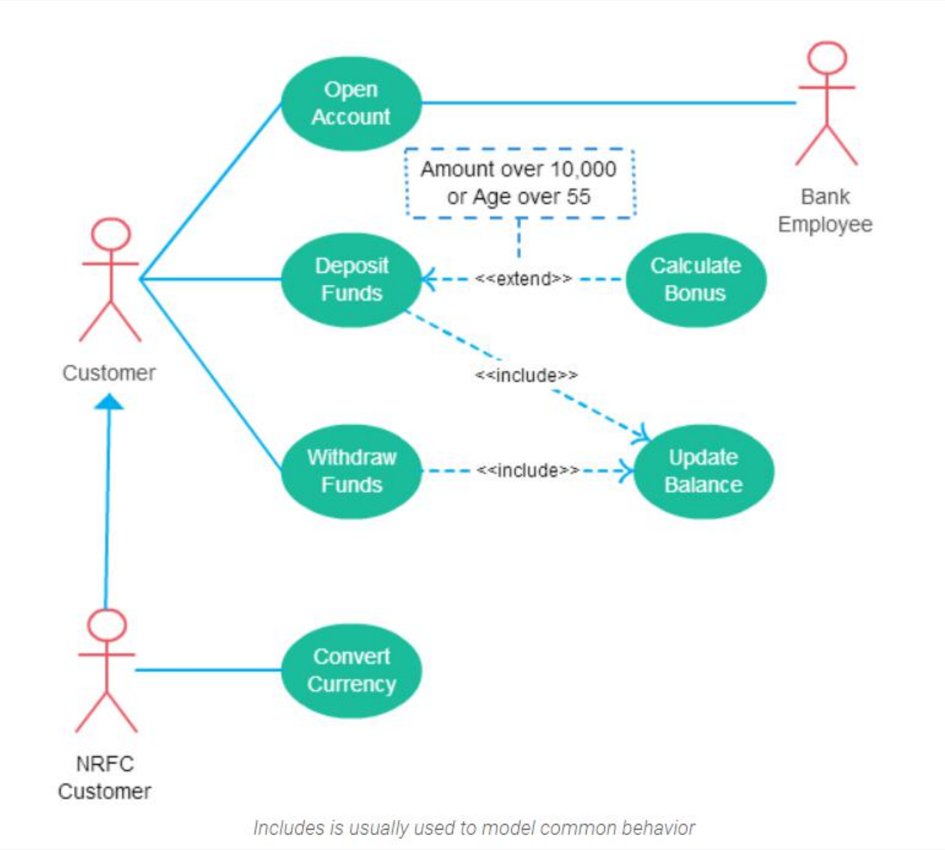
- Ch8: Understanding Requirements (*)
  1. Requirements Engineering：
     - Inception: establishing the groundwork
     - Elicitation: elicit requirements from all stakeholders (including Non-Functional Requirment)
     - Elaboration: create an refined model (data, function and behavioral)
     - Negotiation: agree on a deliverable system (developers and customers)
     - Specification: written document, models, user scenarios, prototype
     - Validation: a review mechanism
     - Requirements management

- Ch9: Requirements Modeling: Scenario-based (****)
  1. Requirements analysis: => analysis model (use-case diagram, data flow, functional activities)
     - operational characteristics
     - interface with other system elements
     - constraints
  2. use-case diagram (Actors, Use cases, Associations, System boundary boxes)
  3. activity diagram (including swimlane diagram)

Use-Case Diagram

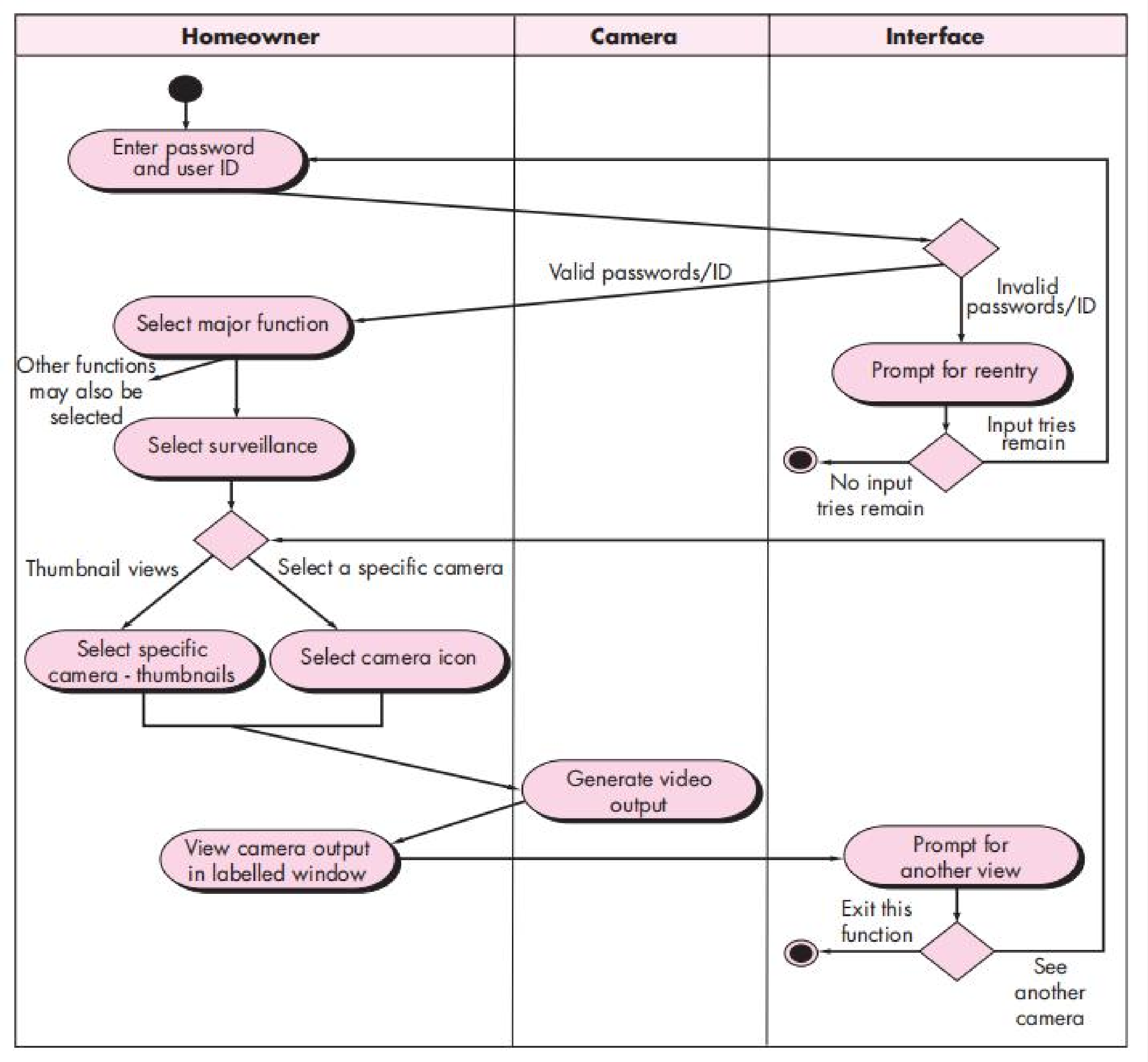
Relationship in use-case diagram

Type	Description
Association	between actor and use case
Generalization (Inheritance)	between actor or between use case
Include (Descompositon)	between use case
Extend	between use case



Includes is usually used to model common behavior

Swimlane Diagrams

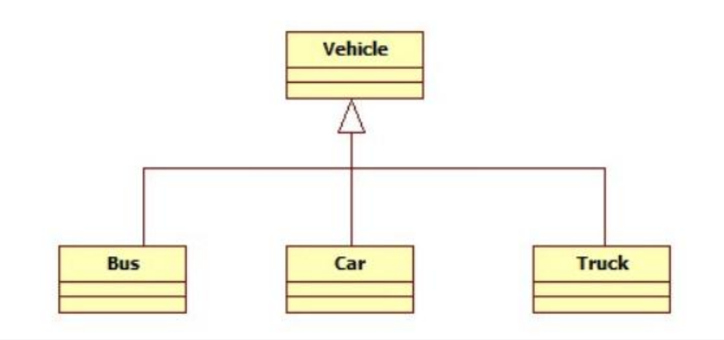


2. Content - part2 Modeling - Requirment (2)

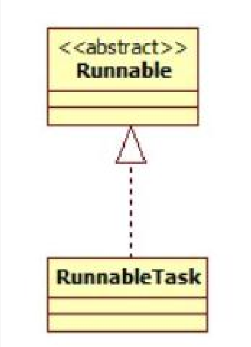
- Ch10: Requirements Modeling: Class-based (****)
  1. structured analysis(data and process) and object-oriented analysis (attributes, operation, class)
  2. Defining potential classes (extracting the nouns)
  3. Specifying attributes (extracting the nouns)
  4. Defining operations (extracting the verbs)
  5. Relation between class (including multiplicity)
     - Generalization (is-a): Inheritance
     - Realization (Interface)
     - Dependency (call)
     - Association (common[reference], aggregation[has-a],composition [contains-a])

- Ch11: Requirements Modeling: Behavior, Patterns, and WebApps (**)
  1. Identifying events and allocate events to the objects.
  2. state diagram
  3. sequence diagram

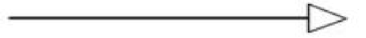
UML Class diagram



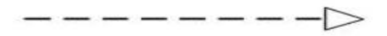
- **Relation between class**



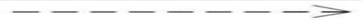
  - □ Generalization (is-a): Inheritance

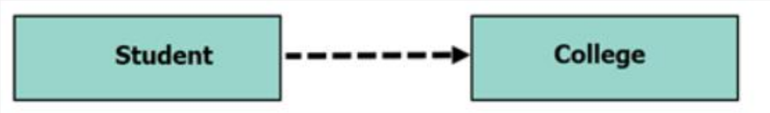


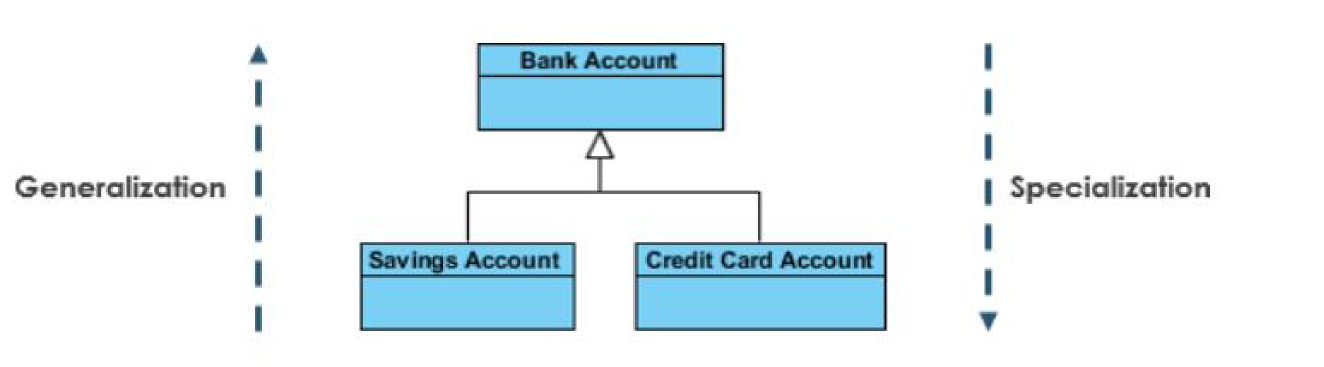
  - □ Realization (Interface)



  - □ Dependency (call)



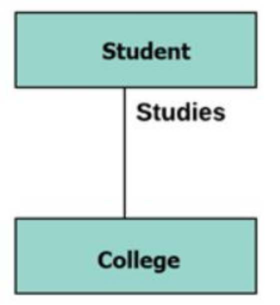




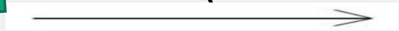
UML Class diagram

- **Relation between class**

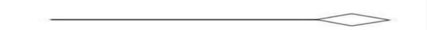
□ Association



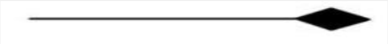
- Common (reference)

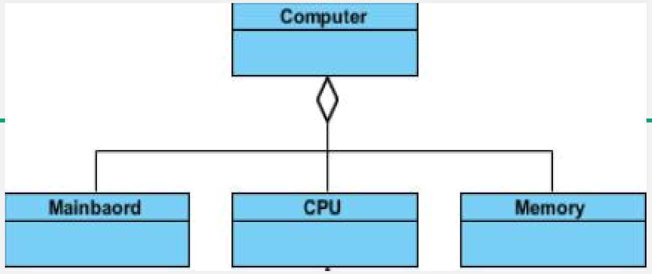


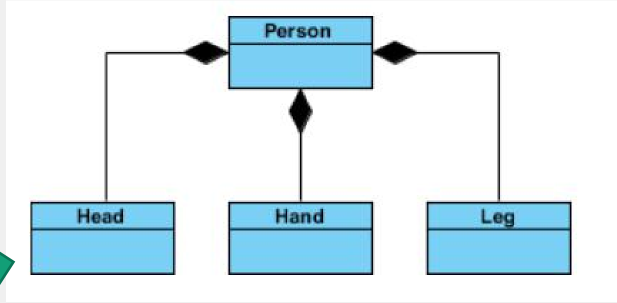
- Aggregation (has-a)



- Composition (contains-a, special case of aggregation)

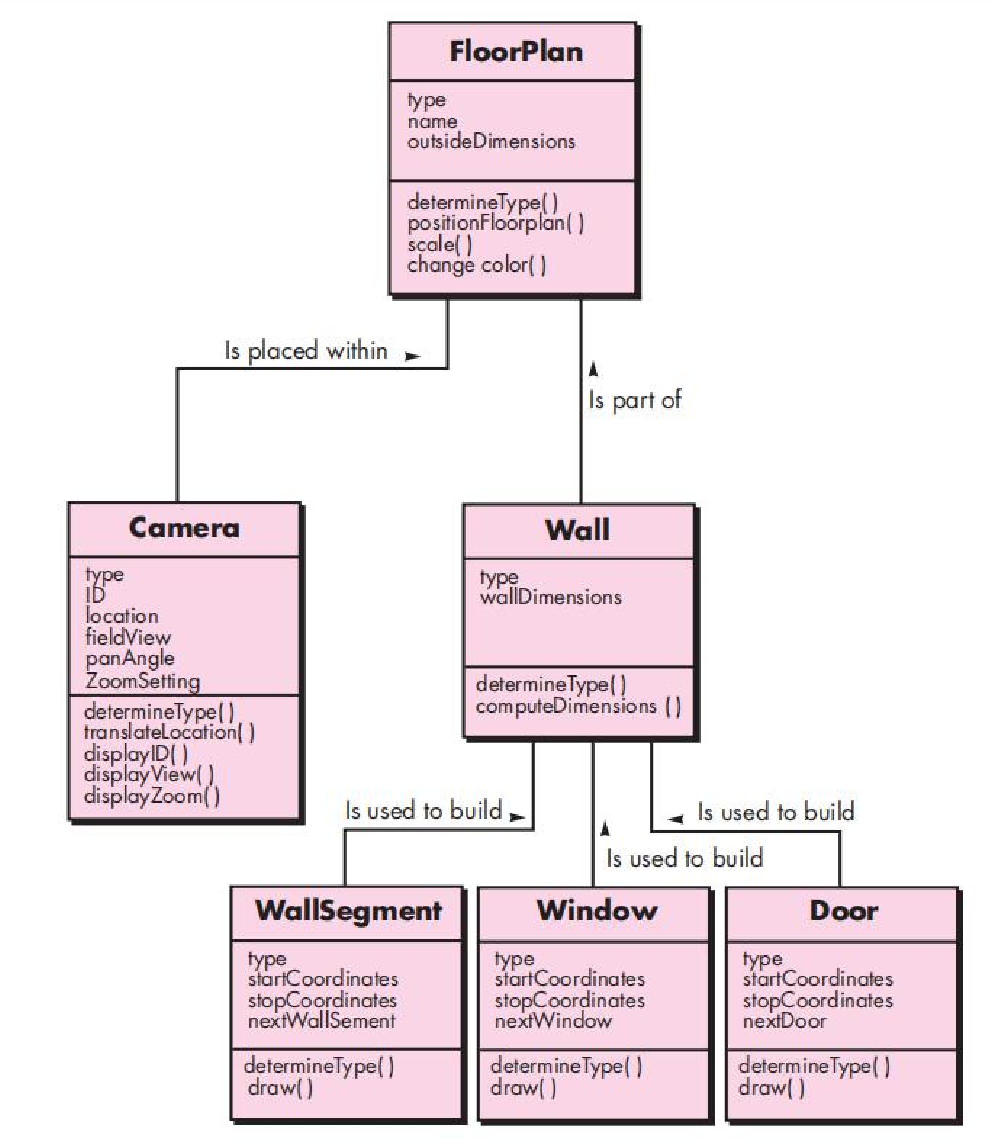


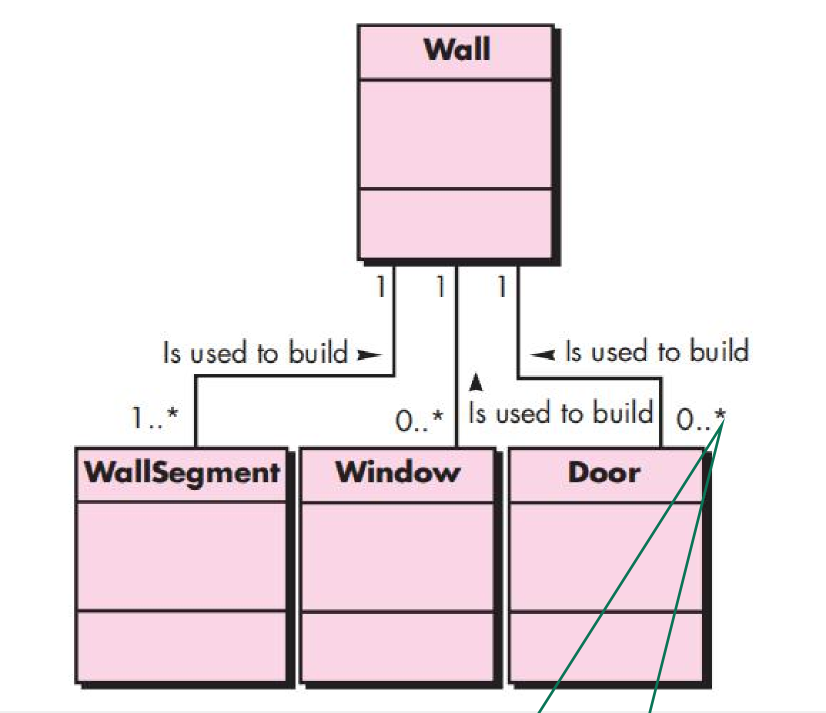




- **Association:** two classes in a model need to communicate with each other.
- **Aggregation** implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the students still exist.
- **Composition** implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate to a House.

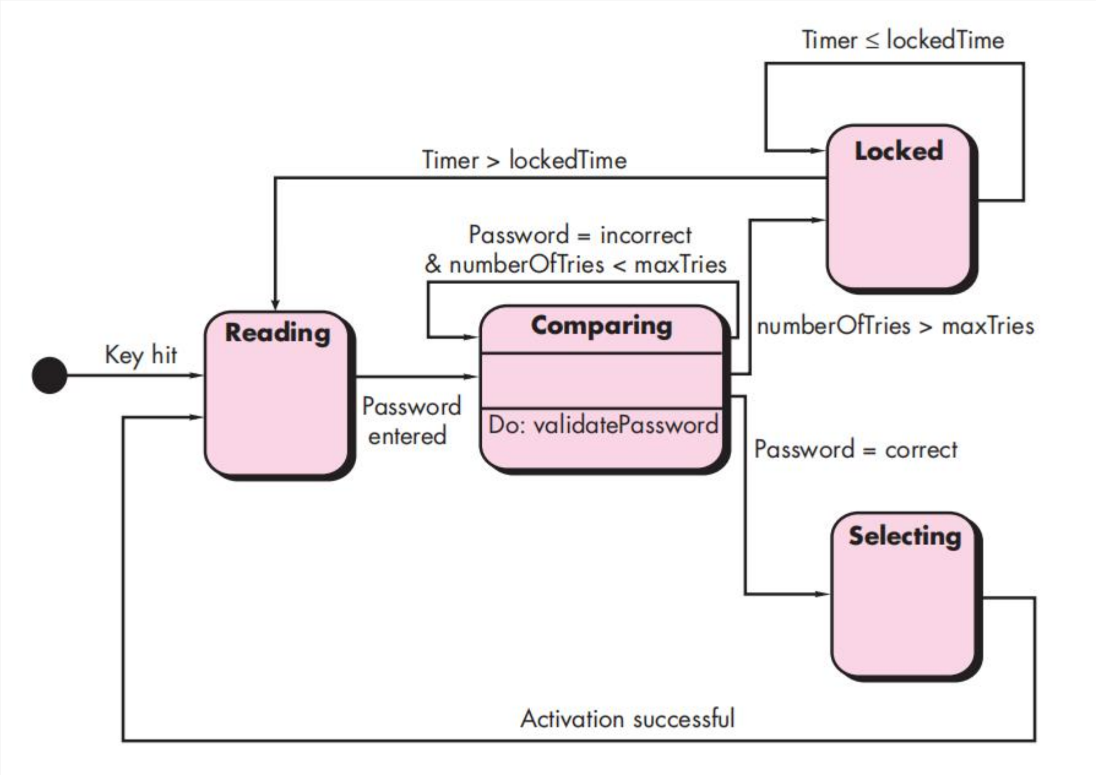
Multiplicity





*: an unlimited upper bound on the range.

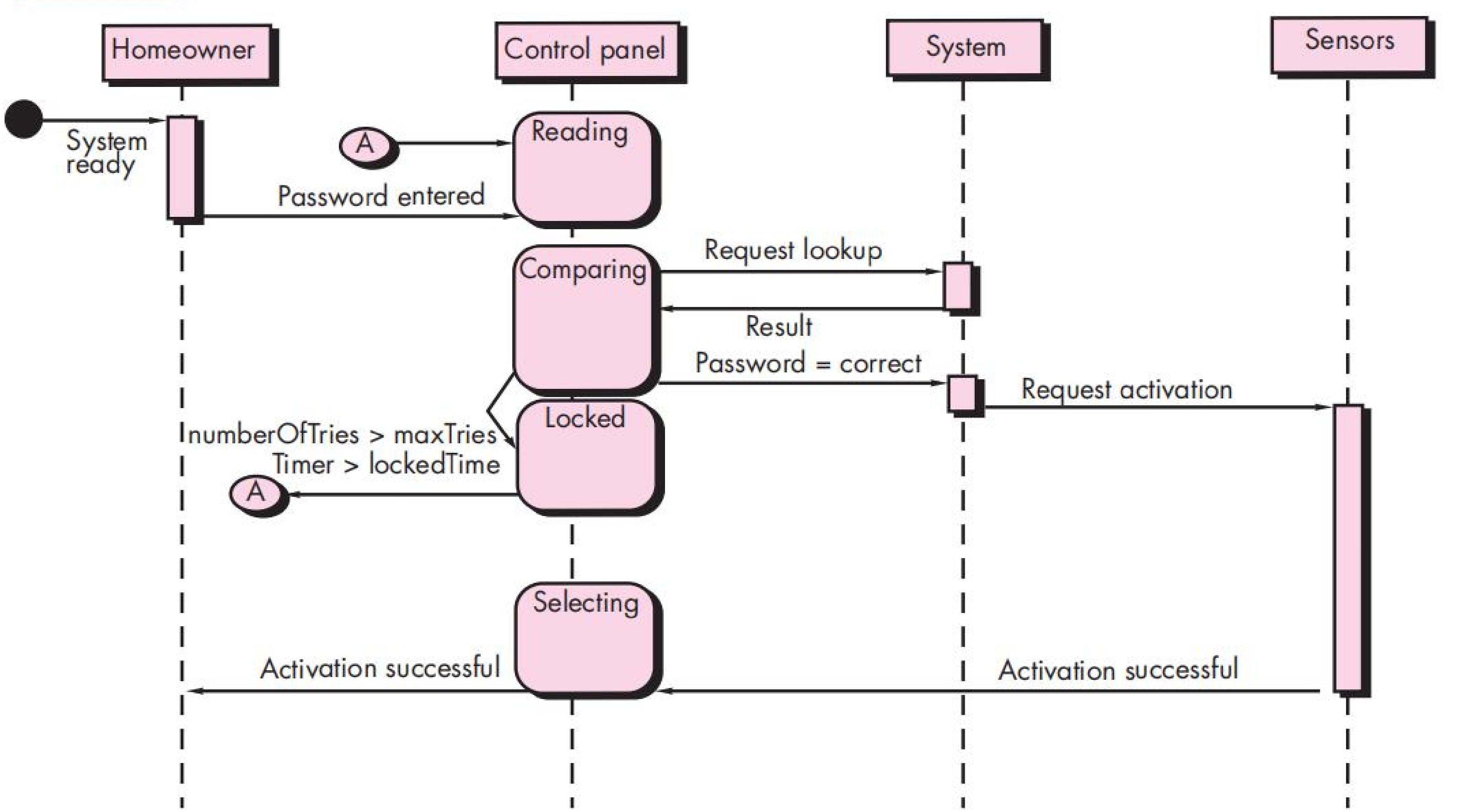
state diagram



how an individual class changes state based on external events

sequence diagram

FIGURE 7.7 Sequence diagram (partial) for the *SafeHome* security function

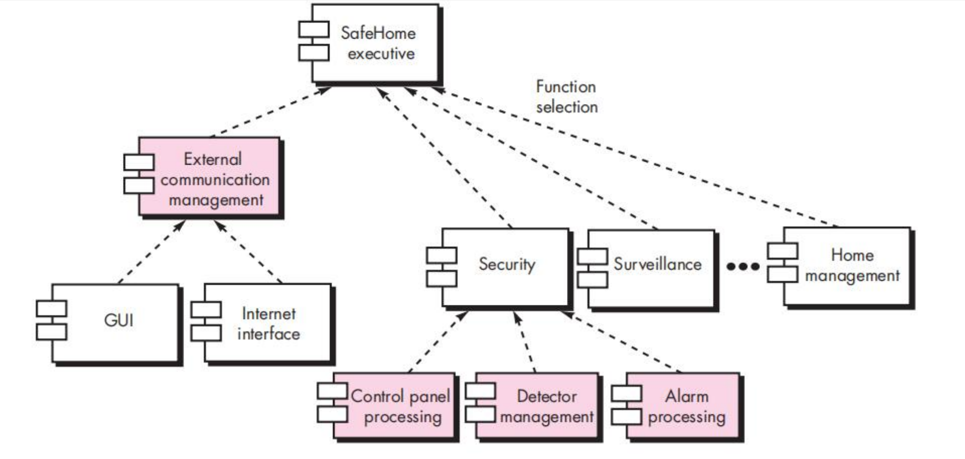


the behavior of the software as a function of time

2. Content - part2 Modeling - Design (1)

- Ch12: Design Concepts (**)
  1. principles, concepts, and practices
  2. design process: data/class design, architectural design, component-level design, interface design
  3. concepts (abstraction, architecture, modularity, information hide, refinement, refactoring, etc)

- Ch13: Architectural Design (***)
  1. Architectural Styles: Data-centered architectures, Data flow architectures, call and return architectures, Object-oriented architectures, Layered architectures
  2. Architectural considerations: Economy , Visibility , Spacing , Symmetry , Emergence
  3. Component diagram



2. Content - part2 Modeling - Design (2)

- Ch14: Component-Level Design (****)
  1. Component : a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. (OO and Conventional view)
  2. OO component design: analysis class (business) + infrastructure class (support class)
     - each class should include all attributes and operations that are relevant to its implementation
     - Basic Design Principles: LSP, DIP, ISP, REP, CCP, CRP
  3. Traditional Component:
     - consider: data structures, data or control object , interface , algorithm (logic)
  4. high Cohesion and low Coupling
  5. Component-level design process for OO software

     ① Identify all design classes (problem domain)

     ② Identify all design classes (infrastructure domain: GUI, OS, object and data management)

     ③ consider component cohesion and coupling
     - message details (collaborate component), interfaces
     - attributes (data types and data structures), processing flow (flow chart)
     - persistent data sources and process class, behavioral representation class
     - deployment diagrams, always consider alternatives

2. Content - part2 Modeling - Design (3)

- Ch15： User Interface Design (****)
  1. Golden Rules.
     - Place the user in control
     - Reduce the user’s memory load
     - Make the interface consistent
  2. process: analysis modeling, design, construction, validation
  3. Design evaluation: KLM, Fitts

- Ch15' Writing code (supplement) (***)
  1. guideline and coding style/rule: readable
  2. documentation:
     - Internal (comment, formatting, etc.)
     - External (documents for desin, test, etc)

2. Content - part3 Quality Management (1)

- Ch19： Quality Concepts (**)
  1. Definition: effective software process, useful product, measurable value
  2. Quality Model: McCall’s Quality Factors
  3. ISO 9126 Quality model:Functionality, Reliability, Usability, Efficiency ,Maintainability , Portability
  4. Cost of Quality: Prevention costs , Internal failure costs, External failure costs
  5. Quality Control: focused on fulfilling quality requirements. (activity to control)
     - method: checklist, Pareto principle, histogram, fishbone graph, etc.
- Ch21： Software Quality Assurance (***)
  1. Methods of SQA:

     Defect prevention: Error blocking, Error source removal, process improvement

     Defect removal: inspection and review, testing

     Defect tolerance: NVP, out-voted,

     recovery: recovery from failure
  2. SQA Group:
     - roles: process review, work products review, audit, record and report.
     - goals: Quality of requirements, design, code; Quality control effectiveness
     - methods: review, statistical analysis
  3. reliability: MTTF, R = MTTF/(1+MTTF)

2. Content - part3 Quality Management (2)

- Ch22：Software Testing Strategies (***)
  1. Testing: objective, Verification and Validation, testing and debug
  2. Process:
     - Unit testing: stub, driver
     - Integration Testing: top down, bottom-up, Bing-Bang, Sandwich
     - Regression testing, smoke testing, validation testing, alpha/beta testing
     - System testing: recovery , security , stress , performance , deployment
- Ch23：Testing Conventional Applications (****)
  1. Concept: “Good” testing, test oracle(expected result)
  2. White-box testing
     - statement coverage, branch coverage, decision coverage
     - branch/decision coverage, combination coverage, basic path coverage
     - Data flow testing, loop testing
  3. Black-box testing
     - Equivalence Partitioning, boundary value analysis
     - Cause-effect graph & Decision table
  4. Combination testing
  5. Usage-based testing (Operational Profile)

2. Content - part3 Quality Management (3)

- Ch24：Testing Object-Oriented Applications (**)
  1. Unit testing: Intra-class testing
  2. Integration testing[inter-class testing]: thread-based testing, use-based testing,cluster testing
  3. Validation testing: use-case in requirement, black-box testing
  4. Methods:
     - Partition: State-based partitioning, Attribute-base, Category-based
     - Inheritance: superclass and subclass
       - ✓ if change some method m() in a superclass, we need to retest m() inside all subclass inherit it
       - ✓ if we change a subclass, we need to retest all related methods inherited from its superclass
       - ✓ Test cases for a superclass method are not necessarily good for retesting its subclass
     - Sequence (Random testing)
     - Behavior (state change)

- Ch29：Software Configuration Management (**)
  1. Configuration management objects: programs, code, documents; tools (svn, github, etc.)
  2. Repository features:
     - versioning, dependency tracking and change management, requirements tracing
     - configuration management, audit trails

2. Content - part4 Managing Software Projects(1)

- Ch31: Project Management Concepts (**)
  1. Management 4P: people, product, process, project
  2. people: Stakeholders, leader, teams paradigms
  3. product
     - scope (Context, Information objectives, Function and performance)
     - problem decomposition
  4. process: activities and actions definition
  5. project: start, maintain momentum, progress, make smart decisions, postmortem analysis
- Ch32: Process and Project Metrics (***)
  1. Why to measure: assess status, adjust , evaluate team's ability
  2. Process metrics
     - quality related, productivity-related (effort expended),
     - statistical SQA data (error categorization & analysis, DRE, etc.),
     - defect removal efficiency, reuse data (reusability)
  3. Project metrics: effort/ effort distribution, errors, schedule, changes
  4. Product metrics: Size(LOC, FP), complexity, Coupling, Cohesion, OO(CK)
  5. Typical size-oriented metrics: errors per person-month, defects per KLOC/FP, etc.

2. Content - part4 Managing Software Projects(2)

- Ch33: Estimation for Software Projects (***)
  1. Plan:
     - software scope: functions/features, input/output; interfaces, performance, constraints, reliability.
     - resources (people: number, skill, location; environment: hardware/software/network; reusable)
     - schedule (ch34)
  2. Estimation
     - size-based: LOC/FP[function decomposition];
     - effort: basic formular; COCOMO II

- Ch34: Project Scheduling (***)
  1. Effort allocation: analysis/design: 40%-50%; construction: 15-20%; testing/installation: 30%-40%
  2. process

     ① effort estimation and allocation

     ② define task network

     ③ timeline chart (gantt chart)
  3. Earned Value Computing

2. Content - part4 Managing Software Projects(3)

- Ch36: Maintenance and Reengineering (*)
  1. Delivering : training and documentation
  2. Software Evolution Policy: maintenance , reengineering
  3. Maintenance type: corrective, adaptive, perfective, preventive
  4. Maintenance effort estimation: Belady and Lehman, COCOMO II
  5. Maintenance metrics: complexity, mean time to repair, number/ratio of changes
  6. Maintenance techniques: change impact analysis
  7. Reengineering: Redocumentation, Reverse engineering, Restructuring, Reengineering

2. Review reference

- text book
- slides / video

3. Grade

- Final exam(70%):
  - test paper (1.5 hours) - 50%
  - small project(2 week) - 50%
- Assignment(20%)： 6 assignments
- Attendance, discussion and quiz (10%)

Final exam： Tencent Meeting. Please install the software in advance, thanks.

4. Deadline

Content	Submission Deadline
Assignment1	17 Jan, 2022
Assignment2(requirement-uml)	6 March, 2022
Assignment3(cohesion&coupling)	20 March, 2022
Assignment4(software testing)	17 April, 2022
Assignment5(software metrics)	20 April, 2022
Assignment6	30 April, 2022
Final exam	?, 2022: 10:30-12:00

5. Project

Finish a software project with the methods, techniques and tools in SE.

- Project topic: no limit
- Team: 3-5 members
- SCM tools: github / gitlab / gitee
- Requirement:
  - Select a team leader.
  - Finish a small but complete project, or some parts of one large project for one team.
  - Everyone of one team should finish at least one function of your project including requirement analysis, design, coding, testing.
  - Everyone’s task should be listed in your project plan.

5. Project (with github/gitlab/gitee，3-5 members per team)

Content	percent	requirement
Management	10%	Make a plan for your objective project. Task list, task assignment, schedule? How to estimate total effort? How to make the quality control? Defect related analysis / evaluation
Requirement analysis	15%	noun and verb analysis: use case diagram, sequence diagram, activity diagram, class diagram, function and non-function
Design	20%	Architectural design, class design, component design, user interface design (KLM or Fits’ law)
Implementation	15%	Programming guideline, part of implementation
Testing	20%	Test strategy, test cases (at least 10 black-box test cases and 10 white-box test cases for each person)
Deployment	10%	Deployment diagram
Maintenance	10%	How to fix bugs or add new features? (Process)

THANKS