Team Project
–
Elevator Simulator

Overview

1. Requirement Analysis
2. Use cases
3. Organizational structure and Programming Language
4. Process model

- ➢ Requirements
- Use cases
- Structure
- Process model

Requirements – Gerneral

- Two floor office building
- One elevator
- Platform independent
- Object oriented - Java
- Capacity of floor and elevator: 1 person

Requirements – Floor and elevator

- ➢ Requirements
- Use cases
- Structure
- Process model

- Elevator:
  - Elevator needs 5 seconds to reach the next floor
  - Button with light
  - Bell
  - Door
- Floor
  - Button with light
  - Arrival-light

➢ Requirements

- Use cases

- Structure

- Process model

Requirements – Simulator

- Simulator:
    - Clock starting with zero
    - Increments by 1 every second
    - Simulator sends time to the scheduler and elevator every second

- ➢ Requirements
- Use cases
- Structure
- Process model

Requirements – Scheduler

- Scheduler:
  - Two random start cases
    - Button pressed in floor 1
    - Button pressed in floor 2
  - Time 1 and 2 randomly between 5-20 (time until button pressed)
  - Floor empty (person x enter elevator)

    → creation of a new Person
  - Scheduler creates next time Tn+1[Tn+5...Tn+20]

✓ Requirements

➢ Use cases

● Structure

● Process model

Use Case – General

1. Name
2. Version
3. Goal
4. Summary
5. Actors
6. Preconditions
7. Triggers
8. Basic course of events
9. Alternative paths
10. Postconditions
11. Business rules
12. Notes
13. Author
14. Date

Use Case #1

- ✓ Requirements
- ➢ Use cases
- ● Structure
- ● Process model

• Name:	Elevator summoned case 1
• Version:	1.0
• Goal:	Transport person to another floor (on shortest possible way)
• Summary:	Elevator summoned by one person
• Actors:	Person, Elevator
• Preconditions:	The scheduler summons a Person. Clock-time is delivered without failures
•Triggers:	Button is pressed by the person

Use Case #1 cont.

- ✓ Requirements
- ➢ Use cases
- ● Structure
- ● Process model

- Basic course of events
    1. Button is pressed
    2. Elevator is not on the floor
    3. Person waits for elevator
    4. Elevator reach floor
    5. Person enters the elevator
    6. Elevator moves to the other floor
    7. Person exits the elevator

✓ Requirements

➢ Use cases

- Structure

- Process model

Use Case #1 cont.

- Alternative paths:

  1. Button is pressed
  2. Person enters the elevator (stays on same floor)
  3. Elevator moves to the other floor
  4. Person exits the elevator

- Postconditions: New person is generated
- Business rules: NA
- Notes: NA
- Authors: Group members/development Team
- Date 01.05.2014

- ✓ Requirements
- ➢ Use cases
- ● Structure
- ● Process model

Use Case #2

• Name:	Elevator summoned case 2
• Version:	1.0
• Goal:	Transport person to another floor (on shortest possible way)
• Summary:	Elevator summoned by two person at the same time
• Actors:	Person 1, Person 2, Elevator
• Preconditions:	The scheduler summons the Persons. Clock-time is delivered without failures
•Triggers:	Buttons are pressed by both person

Use Case #2 cont.

✓ Requirements

➢ Use cases

- Structure
- Process model

- Basic course of events

1. Buttons are pressed
2. Person who stays on the same floor like the elevator enters the elevator
3. Elevator moves to the other floor
4. The other Person waits for elevator
5. Elevator reaches floor
6. Person exits elevator other person enters elevator
7. Elevator moves to the other floor
8. Person exits the elevator

✓ Requirements

➢ Use cases

● Structure

● Process model

Use Case #2 cont.

• Alternative paths:	NA
• Postconditions:	New person is generated
• Business rules:	NA
• Notes:	NA
•Authors:	Group members/development Team
•Date	01.05.2014

✓ Requirements

✓ Use cases

➢ Structure

- Process model

Team structure and programming language

- Team structure:

→ **Democratic Decentralized**

  - No permanent leader
  - Decisions on problems and approach are made by group consensus
  - Communication among team members is horizontal

- Programming language:

→ Object oriented → Platform independent → **Java**

- ✓ Requirements
- ✓ Use cases
- ✓ Structure
- ➢ Process model

Process model

- fix and clear requirements
- fix deadline
- no fix budget
- no system engineering -> only software engineering

→ **Waterfall Model**

Any questions?

Thanks for your attention!