Subway simulator
Case study

Marco Scotto
Software Engineering
2004/2005
Outline

➢ **Requirements**
  ➢ Use cases
  ➢ Class Identification
  ➢ Class Diagrams
  ➢ Sequence & Activity Diagrams
Vision of the subway control system

- Develop a system whose purpose is to control a subway line where trains run with no driver.
- Line is composed by two tracks, one in each direction.
- Along the line, there is a sequence of stations, where the trains must stop.
- In the two terminal stations, the trains change direction, passing to the opposite track.
Trains

- The train movement is controlled by signals.
  - Red signal: the train must stop until the signal color changes to green
  - Green signal, the train may proceed
- Protection signals are placed before any station, to control the entrance to the station.
  - becomes green when the tail of the train overcomes the next start signal
- Start signals are placed and at the end of the station platform to allow trains to exit safely
  - becomes green if the tail of the train overcomes the next protection signal
Sensors

- Sensors on the tracks signal when:
  - the head of a train goes over a signal,
  - the tail of the train goes over a signal.

- Sensors on the train signal that all its doors are closed.
  - Trains are prevented from starting when any door is open.

Control System

- The subway control system must manage:
  - the signal state
  - the train speed

- The subway control system is also equipped of a simulator of train movement:
  - to check the effectiveness of the control system in absence of the actual railway
  - simulates the running of the trains, and the activation of track sensors
Scenario

- Trains go along the line in both ways
- A train:
  - stops at every station
  - opens its doors
  - lets them open for a given time period depending on the station
  - it closes the doors
  - if such operation is successful and if no train is present on the track between it and the following station (the start signal is green), it leaves
Outline

- Requirements
- **Use cases**
  - Class Identification
  - Class Diagrams
  - Sequence & Activity Diagrams
Actors

- **User**
  - a person in charge of setting-up and running the system.

- **Sensor**
  - an external device signaling the presence of a train

- **Train**
Use cases (simulator)

- Loading the subway
- Loading the trains
- Starting the control system
- Starting the simulation
- System visualization
- A train arrives at a signal
- The tail of a train overcomes a signal
- Train at a start signal
- Train at a terminal station
- Train start command
Use cases (UML)
Loading the subway

- A parser reads the file and builds the subway description.
  - If Error: signals the error in an error file and stops
- A subway is composed of a line with stations and tracks
- The line:
  - two tracks (left and right direction)
  - along the subway line, there is a sequence of stations, where the trains stop.
- The track:
  - sequence of track sections
    - line sections connecting two stations
    - station sections
Loading the trains

- The **user** defines the trains specifying their data in a **file with a proper format**
- A **parser** reads the file and builds the train **descriptions**.
  - If error: signals the error in an error file and stops
- **Train** data include their **identifier, length, acceleration, deceleration and maximum speed**. A train cannot be longer than a **track section**
- **Trains** must be on **station sections**, stopped at start signals
Starting the control system

- After loading the subway and trains description, the user gives the system the command to start.
- The software connections to actual track and train sensors are made. These sensors are initialized and checked.
- A consistency check of the whole system is made.
  - If OK: the control system is started.
  - Else, the errors are signaled.
Starting the simulation

- **Simulator** allows simulation of:
  - the running of the trains to check the **effectiveness** of the control system in **absence** of the actual **railway**
    - computes the **train movement**
    - generates the proper **sensor output signals**, and the **door closed signals**
  - the **activation** of track sensors

- The user gives the system the command to start the simulation of the system

- When the simulation is started, the user may visually control the system and the **train motion**
System visualization

A user interface

- shows:
  - the line
  - the train positions on the line
  - the color of the signals

- is updated at every event concerning train movement

- the user may stop the simulation with a proper command
A train arrives at a signal

- Signals are always positioned at the end of track sections
- Every signal is associated to a track end sensor
- If the head of the train is at a signal
  - the sensor signals to the system the event
- If the signal is
  - red: the train stops (until the signal changes its color)
  - green: the train proceeds (adjusting its movement to reach the train’s maximum speed)
- The train movement is computed:
  - trying to minimize the time the train takes to arrive to the next signal
  - taking into account train acceleration, deceleration and maximum speed
  - the velocity the train must have when its head arrived to the next signal (always zero in this case study)
- When the train overcomes a signal, the signal’s color immediately changes to red
The tail of a train overcomes a signal

- When the **tail of the train** overcomes a signal, the sensor signals to the system the event
- The overcome signal remains red, but the signal before it becomes green, since the **protected track section** is now free
Train at a start signal

- At a start signal train always stops and opens its doors
  - the train is in a station, and must let passengers exit from and enter into the train
  - accomplished by letting the start signal always being red
- After a proper time delay, depending on the station, the train closes its doors
  - the doors sensors sense that they are closed
  - the start signal may change color
- If there is a train on the line section after the station section ended by the signal, this remains red
  - becomes green when the tail of the train will free the section
- If there is no train on the two subsequent sections, it becomes green
Train at a terminal station

- A train can enter a **terminal station** only if no other train is in on its sections.
- Once arrived at the end of the **platform**, where there is a signal whose color is always red, the train:
  - changes direction
  - is ready to leave the station, when the start signal becomes green.
Train start command

- A train is stopped in front of a signal whose color became green:
  - the control system sends the train a **start command**

- The train computes its **optimal motion**
  - up to the next red signal, and follows this motion
  - trying to minimize the time the train takes to arrive at zero speed to the next red signal, taking into account train:
    - acceleration,
    - deceleration
    - maximum speed
Outline

- Requirements
- Use cases
- Class Identification
- Class Diagrams
- Sequence & Activity Diagrams
Nouns

user
subway
layout
file with a proper format
parser
description
error
error file
line
station
track
left and right direction
subway line
train
sequence
track section
line section
station section
protection signal
access
start signal
interchange section
terminal station
course
way
identifier
name
length
meter
delay
minimum time
train door
sensor
head of a train
tail of the train
succession of stations and line sections
data
length value
second
acceleration
deceleration
maximum speed
speed
train direction
Km/hour
m/sec²
command to start (simulation or system)
software connection
track sensor
train sensor
check of consistency
control system
simulator
train movement
effectiveness
absence
railway
activation
sensor output signal
door closed signal
train motion
user interface
train positions
color of the signal
event
command
end of section
track end sensor
red
green
time
passenger
time delay
doors sensor
platform
start command (from a signal)
Nouns (continued)

- **Entities external to the system:**
  - User, file with a proper format, error file, passenger, train door, platform.

- **Synonyms:**
  - Railway (subway),
  - subway line (line),
  - course (direction),
  - way (direction),
  - succession of stations and line sections (line),
  - length value (length),
  - track end sensor (track sensor),
  - train sensor (doors sensor)

- **Nouns denoting actions:**
  - Access,
  - command to start (simulation or system),
  - check of consistency,
  - train movement,
  - activation,
  - train motion,
  - command
Nouns (continued)

- **Low-level data, attributes, data types:**
  Left and right direction, identifier, name, length, meter, delay, minimum time, second, acceleration, deceleration, maximum speed, speed, train direction, Km/hour, m/sec2, train position, color of the signal, red, green, time, time delay

- **Irrelevant nouns:**
  Layout, description, sequence, data, software connection, effectiveness, absence, user interface

- **Potential objects of the system:**
  Subway, parser, line, error, station, track, train, track section, line section, station section, protection signal, start signal, interchange section, terminal station, sensor, track sensor, control system, simulator, sensor output signal, door closed signal, event, doors sensor, start command (from a signal)

- **To be defined:**
  Head of a train, tail of the train, end of section
Classes

- **Objects describing the layout of the subway**
  Subway, line, station, track, track section, line section, station section, protection signal, start signal, interchange section, terminal station, sensor, track sensor

- **Objects describing the trains and their movement**
  Train, sensor output signal, door closed signal, event, doors sensor, start command (from a signal)

- **System objects and design objects**
  Parser, error, control system, simulator
Class Diagram
Note

- Class **Line** simply holds two tracks, and has no specific data or behavior
  - It can be merged with class **Subway**
- A **Sensor** should have an identifier.
  - **Track Sensor** and **Signal** are a kind of **Device** (a more complex hierarchy, with class “Device” at its root)
- In the model, there are obvious constraints, such as:
  - A **LineSection** is associated to the protection signal at its end
  - A **StationSection** is associated to the start signal at its end
  - A **TerminalStation** has an **InterchangeSection**
Class Diagram (adjusted)
Note

- Relationships among Stations and TrackSections are clumsy. A Station has two StationSections, and this association is inherited by class TerminalStation, clearly referring to InterchangeSection.

- Unclear if (terminal station)
  - a TerminalStation has a single InterchangeSection,
  - or one InterchangeSection and two other track sections

- In reality, a TerminalStation contains three TrackSections, but here:
Class Diagram (adjusted)
First Iteration: Subway Parser

- The subway parser is an object able to read from a file the description of a subway, and to create the corresponding subway objects.

- The key object of the subway parser:
  - the parser itself
  - object holding the station data
    - (StationRecord registers all station data reported in the description file)
Subway Parser (class diagram)
Subway Dynamic Objects

The dynamic objects continuously change states during the simulation

- the **train** has a position on a track section, a speed and a doors’ status (open or closed)
- the **signal** has a color
- the **simulator** manages an event queue. The events that may occur during the simulation are:
  - a train is put on a track section (when the simulation starts)
  - a train head arrives at a signal
  - a train tail overcomes a signal
  - the doors of the train are closed
  - the simulation is ended
Event queue

- Insert into the queue the events of train inputs
- Pop from the queue the event with lower time
- Executes the event possibly generating and inserting other events into the queue
- is the queue empty?
  - NO: Pop from the queue another event with lower time
  - YES: End of simulation

---

Software Engineering 35