EECS 22L: Software Engineering Project in C Language

Lecture 8

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 8: Overview

- Project 2 Technical Discussion and Advise
  - Application specification
  - Software architecture and components
  - Software configuration
- Introduction to Socket Communication
  - Basic terms and concepts
  - Client-server example

Project 2

- Introduction
  - Taxi Cab Management
  - Cover Story:

    *The **City of New Irvine** is soliciting proposals for the development of a new fully-automated taxi cab management service that can optimally coordinate a number of taxi cabs serving the individual public transportation needs of the population and visitors of the city.*

    The City of New Irvine expects transportation by taxi to be very flexible and efficient for individual people and small groups. Customers may request rides from and to every street corner within the city limits, to be served at the time of request or at a defined later time as specified by the customer.
  - Task:

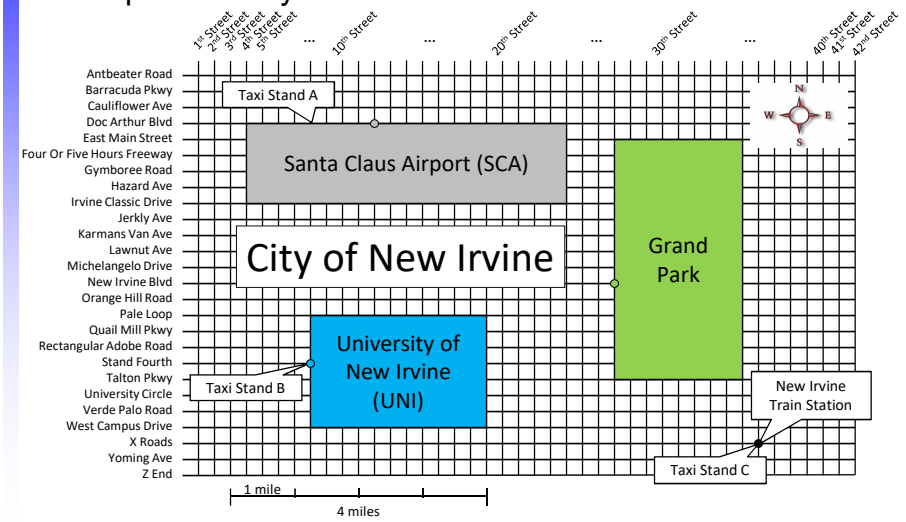
    Design a networked computer server and corresponding client apps where customers' requests are handled and automatically processed. While customers' expectations of punctuality and quick trips are to be met, revenue may be earned and maximized by the taxi company.

Project 2: Cover Story

- Goals and Requirements
  - Customer rides from/to every corner or landmark within the city limits
  - Customer rides as soon as possible (or at specified later time)
  - Up to about 20 minutes ride, obeying 45 MPH speed limit
  - Taxi stands
    - Up to 12 cab cars (with varying passenger capacity)
  - Fair and easy transportation of individuals and small groups
  - Efficiently routed trips
  - Revenue regulated by configuration, to be maximized
  - Central management with dedicated customer apps

Project 2: Cover Story

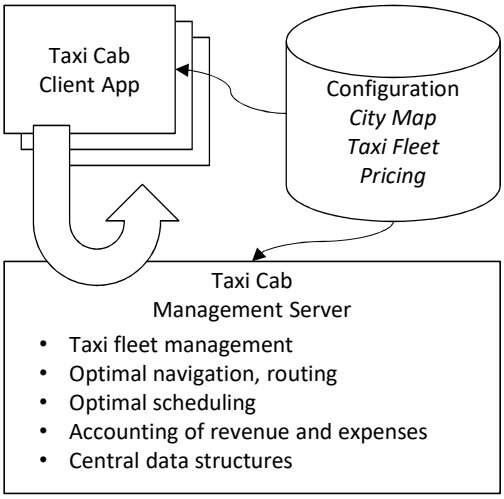
- Map of the City of New Irvine



EECS22L: Software Engineering Project in C, Lecture 8 (c) 2017 R. Doemer 5

Project 2: Software Architecture

- Overall System Specification (designed by consultant)



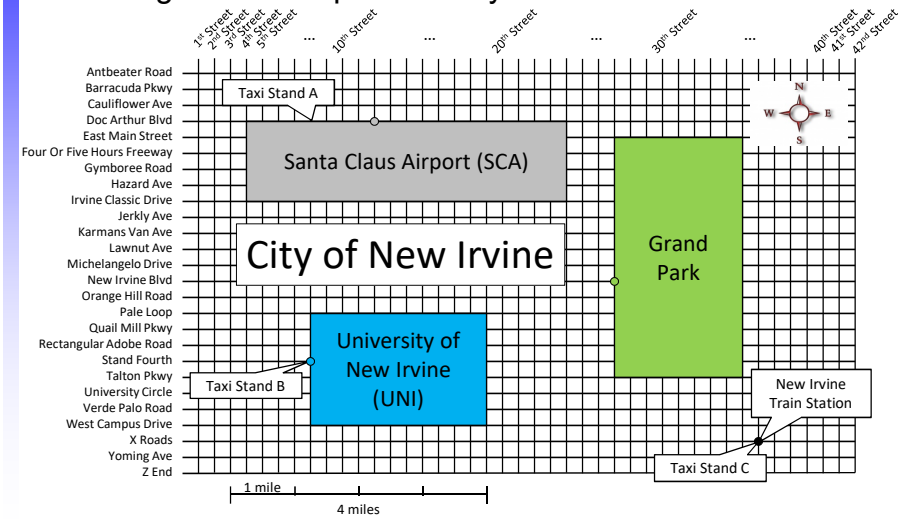
EECS22L: Software Engineering Project in C, Lecture 8 (c) 2017 R. Doemer 6

Project 2: Software Architecture

- Overall System Specification (designed by consultant)
  - Taxi Cab Management Server
    - Central server connected to the internet (with GUI dashboard)
    - Taxi fleet management
    - Optimal navigation, routing (and scheduling)
    - Accounting of revenue and expenses
    - Central data structures
  - Taxi Cab Client App
    - Customer interface to request taxi rides (with GUI interface)
    - Networked client app that communicates with the central server
  - Configuration Files
    - City Map (Grid, street names, landmark locations, etc.)
    - Taxi Fleet (Number of taxi cabs, capacity and parking locations, etc.)
    - Pricing Structure (Base price per hire, distance dependent rate, etc.)
    - ➢ All described in human and machine readable text file

Project 2: Software Configuration

- Configuration: Map of the City of New Irvine



Project 2: Software Configuration

- Configuration: Map of the City of New Irvine
  - Configuration file **"NewIrvine.map"**

```
MAP City of New Irvine

GRID 26 42

STREET_NAMES_EAST_WEST Antbeater Road, Barracuda Pkwy, Cauliflower Ave, Doc Arthur
Blvd, East Main Street, Four Or Five Hours Freeway, Gymboree Road, Hazard Ave, Irvine
Classic Drive, Jerkly Ave, Karmans Van Ave, Lawnut Ave, Michelangelo Drive, New Irvine
Blvd, Orange Hill Road, Pale Loop, Quail Mill Pkwy, Rectangular Adobe Road, Stand
Fourth, Talton Pkwy, University Circle, Verde Palo Road, West Campus Drive, X Roads,
Yoming Ave, Z End

STREET_NAMES_NORTH_SOUTH 1st Street, 2nd Street, 3rd Street, 4th Street, 5th Street,
[…street names omitted for brevity…] 42nd Street

LANDMARK S8(P8,W19) University of New Irvine (UNI)
LANDMARK D4(D12,I24) Santa Claus Airport (SCA)
LANDMARK N27(E27,T35) Grand Park
LANDMARK X36 New Irvine Train Station

TAXI_STAND D8(12) Taxi Stand A
TAXI_STAND S8(12) Taxi Stand B
TAXI_STAND X36(12) Taxi Stand C
```

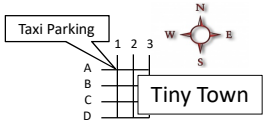

Project 2: Software Configuration

- Configuration: Map of Tiny Town (minimal example)
  - Configuration file **"TinyTown.map"**

```
MAP Tiny Town

GRID 4 3

TAXI_STAND A1(3) Taxi Parking
```



- No landmarks given, so assume default: no landmarks
  - Streets are reaching over the entire grid
- No street names given, so use default names instead:
  - **STREET_NAMES_EAST_WEST A, B, C, D**
  - **STREET_NAMES_NORTH_SOUTH 1, 2, 3**

Project 2: Client-Server Architecture

- Introduction to Socket Communication
  - Basic terms and concepts
    - Point-to-point communication, often designed as client-server model
    - *Client* initiates communication, sends a *request*
    - *Server* waits, services client request, sends a *response*
    - Client and server are software *processes* executing on *hosts*
    - Hosts are typically *distributed* (networked), but can also be identical
    - *Sockets* represent a *network connection* between two processes
    - Sockets operate *bidirectional* (both sides can *send* and *receive*)
    - *Stream sockets* implement *connection-oriented* semantics
      - Data is transported reliably in-order, without loss or duplication
      - *Transmission Control Protocol* over *Internet Protocol*, *TCP/IP*
    - Hosts have Internet Protocol (IP) *addresses* and *ports*
      - Host **`crystalcove.eecs.uci.edu`** has IP address **`128.200.85.14`**
      - Ports below 1024 are reserved (e.g. port 80 for web browsing)
      - Ports above 2000 are typically "free" for application-specific use

Project 2: Client-Server Architecture

- Introduction to Socket Communication
  - Simple client-server example
    - **`http://www.linuxhowtos.org/C_C++/socket.htm`**
    - **`http://www.linuxhowtos.org/data/6/client.c`**
    - **`http://www.linuxhowtos.org/data/6/server.c`**
  - Extended client-server example:
    `~eecs22/SocketTutorial.tar.gz`
    - **`client2.c`**
    - **`server2.c`**
    - **`Makefile`**
    - **`README`**
  - ➢ Online demonstration!

Project 2: Client-Server Communication

- Communication Example: *"Call a Taxi from UNI to SCA"*
  - Client: `Hello Server!`
  - Server: `ERROR invalid message "Hello Server!"`
  - Client: `REQUEST_TAXI S8 TO D4 ASAP`
  - Server: `OK Taxi7 PICKUP S8 10:15 DROPOFF D4 11:42 $8.75 CONFIRM`
  - Client: `OK CONFIRMED`
  - Server: `OK Taxi7 POSITION D8 ETA S8 10:15`
  - Client: `REQUEST_POSITION Taxi7`
  - Server: `OK Taxi7 POSITION J2 ETA D4 11:45`
  - Client: `REQUEST_POSITION Taxi7`
  - Server: `OK Taxi7 POSITION D8`

Project 2: Client-Server Communication

- Protocol Specification Example (Backus-Naur Form, BNF)
  - Client Request

```
<request> ::= REQUEST_TAXI <location> TO <location> <special>*
            |   REQUEST_POSITION <taxi>
<location>::= <position>
            |   CORNER <street_name> AND <street_name>
            |   <landmark_name>
<special> ::= [ASAP]
            |   [AT <time>]
            |   [FOR <number_persons>]
            |   [NON_STOP]
<time>    ::= <hours>:<minutes>
```

  - Server Response

```
<response>::= OK <taxi> PICKUP <position> [<time>]
              DROPOFF <position> [<time>] $<amount> CONFIRM
            |   DECLINED <reason>
            |   OK <taxi> POSITION <position> [ETA <position> <time>]
            |   ERROR <message>
<amount>  ::= <dollars>.<cents>
```