# EECS 22L: Software Engineering Project in C Language

## Lecture 7

Rainer Dömer

doemer@uci.edu

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

## Lecture 7: Overview

- Course Administration
  - Midterm course evaluation
  - New project setup
- Project 2
  - Focus points
  - Introduction
  - Application specification

# Course Administration

- Midterm Course Evaluation: Results
  - Participation
    - 23 out of 77 students (29.87%)
    - ➢ Not quite representative
  - Student Feedback
    - Mostly very positive and encouraging
      - Overall letter grade: 18 “A”, 3 “A-”, 1 “B+”
    - Specific comments
      - ➢ **`MidtermEvaluation_Report.pdf`**



# Course Administration

- New Teams
  - New teams have been formed for Project 2
    - Almost all preferences have been met
      - ➢ A number of teams stay together unchanged
    - Very few cyclic conflicts
  - 16 new teams, 4 to 6 members each
    - See individual team emails
- New Team Accounts
  - All team accounts have been cleared (all files deleted)
  - New passwords will be distributed in discussion session
    - Use lab sessions this week to set up fresh team account

## Project 2

- New Focus Points
  - Graphical User Interface (GUI)
    - mandatory
  - Version control with CVS
    - mandatory
  - Testing
    - Unit test
    - System test
  - Client Server Application
    - TCP/IP communication via sockets

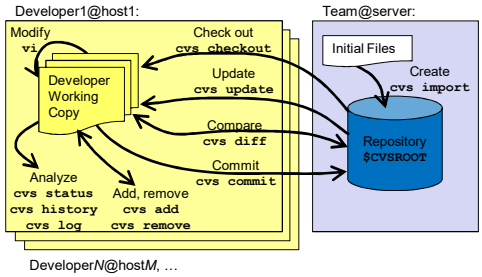


## Project 2 Focus: GUI

- Graphical User Interface (GUI)
  - Differences to command-line interface
    - Input through events
    - Output through 2D pixel interface
    - Event-based main control loop
  - Available GUI libraries
    - GTK: GNOME or GIMP Toolkit
      - To be discussed in an upcoming lecture
      - **`http://www.gtk.org/`**
    - SDL: Simple DirectMedia Layer
      - Refer to Lecture 4
      - **`http://www.libsdl.org/`**

# Project 2 Focus: CVS

- Version Control using CVS
  - Source code management
    - For successful team work, version management is critical
    - Once properly set up, it's all a matter of '**`cvs update`**' and '**`make clean all`**'




➢ Refer to Lecture 3 and online documentation



# Project 2 Focus: Testing

- Perform Unit Tests and System Test
  - Test each module/component individually
    - With specific test data
    - Automatically
  - Run each module/component in debug mode
    - **`make test`**
  - ➢ Top-level **`Makefile`** should provide targets to run tests for each unit and the entire system
  - Unit tests
    - **`% make test_gui`**
    - **`% make test_client`**
    - **`% make test_server`**
  - System test
    - **`% make test`**

# Project 2 Focus: Testing

- Perform Automated Unit and System Test
  - Test each module individually (with specific test data)
  - Use **`make test`** to run each module in debug mode
    - Example: Student records (see EECS 22, Lectures 14 ff.)

```
/* Student.c: maintaining student records */
...
#ifdef MAIN  /* test the student record functions */
int main(void)
{   STUDENT *s1 = NULL;
    s1 = NewStudent(1001, "Jane Doe", 'A');
    PrintStudent(s1);
    [...]
    return 0;
} /* end of main */
#endif /* MAIN */
/* EOF */
```

```
% vi Student.c
% make Student
gcc -Wall -ansi -g -c Student.c -o Student.o
gcc -DMAIN -Wall -ansi -g Student.c Student.o -o Student
```



# Project 2 Focus: Client-Server App

- Client Server Software Architecture
  - Client App: sends requests to the server
  - Server Program: processes clients' requests
- Networked Inter-Process Communication
  - Internet protocol
  - TCP/IP communication via sockets
  - ➢ Details to be discussed in Lecture 8

# 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.*

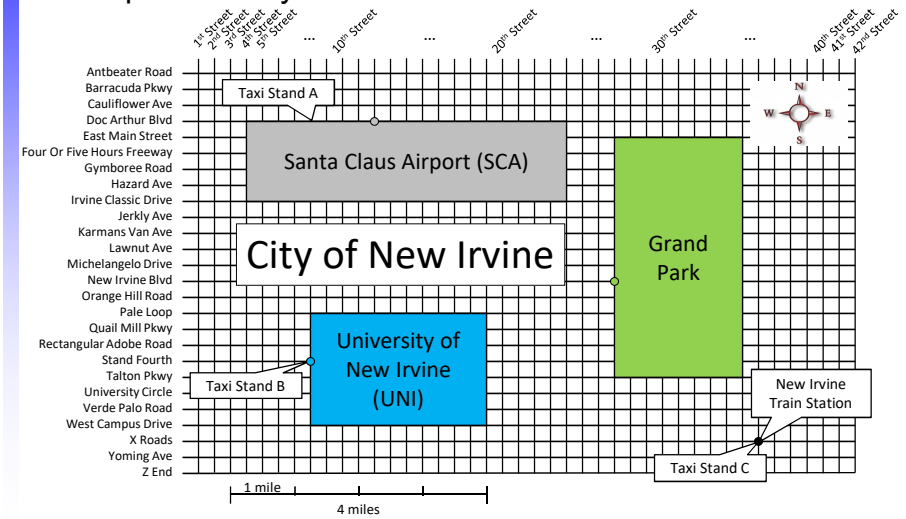
    *The City of New Irvine expects transportation by taxi to be very flexible and efficient for individual people and small groups. Customers may request rides from and to every street corner within the city limits, to be served at the time of request or at a defined later time as specified by the customer.*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

EECS22L: Software Engineering Project in C, Lecture 7 (c) 2017 R. Doemer 11

# Project 2: Cover Story

- Map of the City of New Irvine




EECS22L: Software Engineering Project in C, Lecture 7 (c) 2017 R. Doemer 12

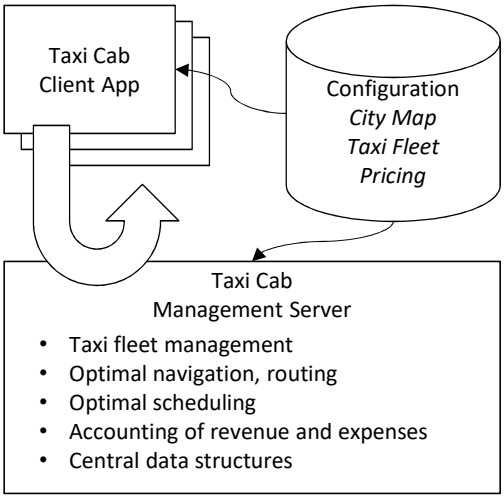
# Project 2: Cover Story

- 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: Software Architecture

- Overall System Specification (designed by consultant)



# 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)