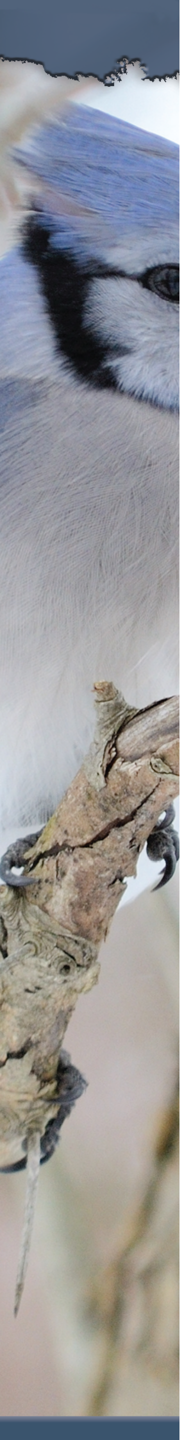
A case study

Whole-application development

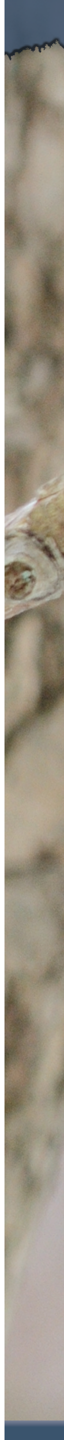
The case study

- A taxi company is considering expansion.
  - It operates taxis and shuttles.
- Will expansion be profitable?
- How many vehicles will they need?



The problem description (1)

- The company operates both individual taxis and shuttles.
- The taxis are used to transport an individual (or small group) from one location to another.
- The shuttles are used to pick up individuals from different locations and transport them to their several destinations.
- When the company receives a call from an individual, hotel, entertainment venue, or tourist organization, it tries to schedule a vehicle to pick up the fare.

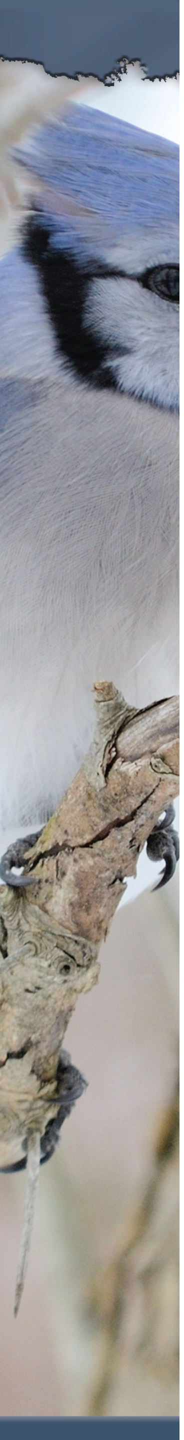


The problem description (2)

- If it has no free vehicles, it does not operate any form of queuing system.
- When a vehicle arrives at a pick-up location, the driver notifies the company.
- Similarly, when a passenger is dropped off at their destination, the driver notifies the company.

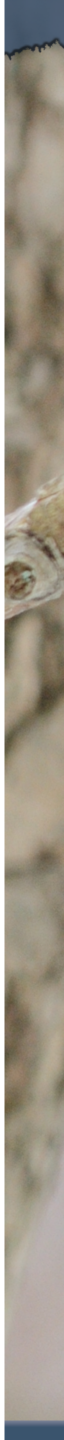
Amendments

- The underlying purpose of the modeling suggests additions:
  - Record details of lost fares.
  - Record details of how each vehicle passes its time.
- These issues will help assess potential profitability.



Discovering classes

- Singular nouns: company, taxi, shuttle, individual, location, destination, hotel, entertainment venue, tourist organization, vehicle, fare, pickup location, driver, passenger.
- Identify *synonyms*.
- Eliminate superfluous detail.



Simplified nouns and verbs

Company
Operates taxis.
Receives calls.
Schedules a vehicle.

Taxi
Transports a passenger.

Location

Passenger

Vehicle
Pick up individual.
Arrives at pickup location.
Notifies company of arrival.
Notifies company of drop-off.

Passenger source
Calls the company.

Shuttle
Transports one or more passengers.

Scenarios

- Follow a pickup through from request to drop off with CRC cards.

PassengerSource	*Collaborators*
Create a passenger.	**Passenger**
Request a taxi.	**TaxiCompany**
Generate pickup and destination.	**Location**

Designing class interfaces

```
public class PassengerSource {
    /**
     * Have the source generate a new passenger and
     * request a pickup from the company.
     * @return true If the request succeeds,
     *         false otherwise.
     */
    public boolean requestPickup()

    /**
     * Create a new passenger.
     * @return The created passenger.
     */
    private Passenger createPassenger()
}
```

Collaborators

- Received through a constructor:

```
new PassengerSource(taxiCompany)
```

- Received through a method:

```
taxiCompany.requestPickup(passenger)
```

- Constructed within the object.
  - Taxi company's vehicle collection.
  - Some such objects may be passed as collaborators to other objects, as above.

Outline implementation

- Once the interfaces are complete, the outline implementation can start.
- The outline implementation tests the adequacy of the interfaces.
  - Expect to have to correct the design.
- Lay down some basic tests that will be repeated as development continues.

Iterative development

- Take relatively small steps towards the completion of the overall application.
- Mark the end of each step with a period of testing.
  - Regression test.
  - Fix errors early.
  - Revisit earlier design decisions, if necessary.
  - Treat errors-found as successes.

Review

- Robust software requires thoughtful processes to be followed with integrity.
  - Analyze carefully.
  - Specify clearly.
  - Design thoroughly.
  - Implement and test incrementally.
  - Review, revise and learn. Nobody's perfect!

Extensions

- If we have time, we'll be adding extensions to the project...