

myTaxiService

Software Engineering 2 - Project

Alberto Maria Metelli Riccardo Mologni

Politecnico di Milano
M. Sc. in Computer Science and Engineering

DD Presentation, 9th December 2015

Outline

- 1 Introduction
 - Pattern + Style = Architecture
 - JEE
- 2 Architectural design
 - Component view
 - Deployment view
 - Runtime view
- 3 Algorithm design
 - Taxi queue management
- 4 User interface design

Outline

- 1 Introduction
 - Pattern + Style = Architecture
 - JEE
- 2 Architectural design
 - Component view
 - Deployment view
 - Runtime view
- 3 Algorithm design
 - Taxi queue management
- 4 User interface design

Pattern + Style = Architecture

PROBLEM



CONTEXT

Pattern + Style = Architecture

PROBLEM



PATTERN



STYLE



CONTEXT

Pattern + Style = Architecture

PROBLEM



PATTERN

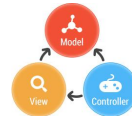


STYLE



CONTEXT

MVC



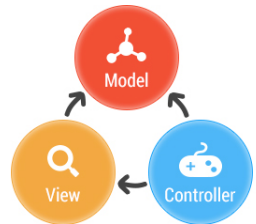
CLIENT/SERVER

MVC

Architectural pattern

Problem: applications with user interface

- Separation of concerns
- Design and conquer development
- Maintainability



Client/Server

Architectural style

Context: distributed application

- Separation of roles
- Lot of source of information
- Just one elaboration point
- Maintainability



Three tier

Architectural style flavour

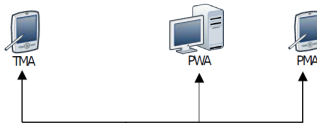
- Different computational and storage resource
- Presentation on clients (Tier 1)
- Separation between
 - data (Tier 3) and
 - business logic (Tier 2)
- Further separation within Tier 2:
 - visualization (web server)
 - processing (application server)



Overall architecture

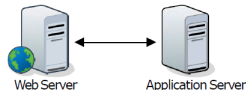
Presentation tier

Passengers and taxi drivers use the client applications to send request through the web. Clients perform input validation, send request to the server and show the response received from the server to the users.



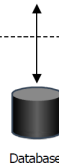
Application tier

A server web receives the clients' requests and an application server performs the requested actions. If needed, the application server composes a query to ask the database and sends the result to complete the action.



Data tier

The database stores and manages all the data required to perform the service. It receives query from the logic layer, computes the answer and sends back the result.



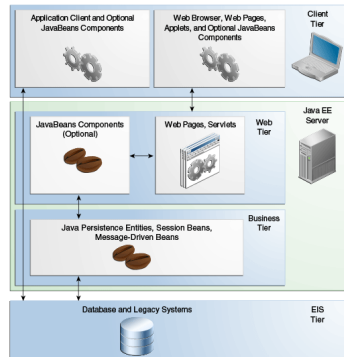
Outline

- 1 Introduction
 - Pattern + Style = Architecture
 - JEE
- 2 Architectural design
 - Component view
 - Deployment view
 - Runtime view
- 3 Algorithm design
 - Taxi queue management
- 4 User interface design

JEE Architecture

Motivations

- Modularity
- Reliability
- Security
- Portability
- Scalability



Outline

- 1 Introduction
 - Pattern + Style = Architecture
 - JEE
- 2 Architectural design
 - Component view
 - Deployment view
 - Runtime view
- 3 Algorithm design
 - Taxi queue management
- 4 User interface design

Subsystems

Definition

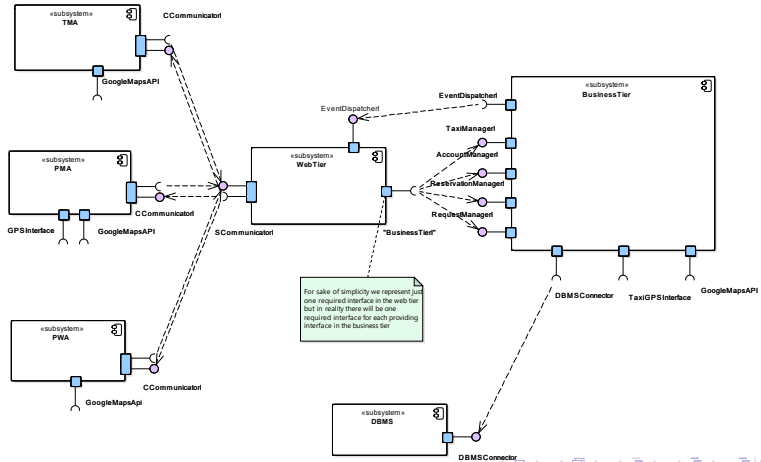
A *subsystem* is a group of components that belong to the same “role”

- TMA (Taxi Mobile Application)
- PMA (Passenger Mobile Application)
- PWA (Passenger Web Application)
- Web tier
- Business tier
- DBMS

Subsystems

High level component diagram

Link



Components

Business Tier

Definition

A *component* is a cohesive and little coupled group of functionalities that can be almost mapped to a programmatic class.

- AccountManager
- RequestManager
- ReservationManager
- GoogleMapsCommunication
- DBManager
- TaxiManager
(*macro-component*)
 - TaxiSelector
 - TaxiPositionFinder
 - TaxiQueueManager
 - TaxiStateChanger

Business Tier - Component diagram

All *Manager should be connected to the required interface. Event2Dashboard, for graphical reasons, those connections are not represented.



Outline

- 1 Introduction
 - Pattern + Style = Architecture
 - JEE
- 2 Architectural design
 - Component view
 - Deployment view
 - Runtime view
- 3 Algorithm design
 - Taxi queue management
- 4 User interface design

Cloud deployment

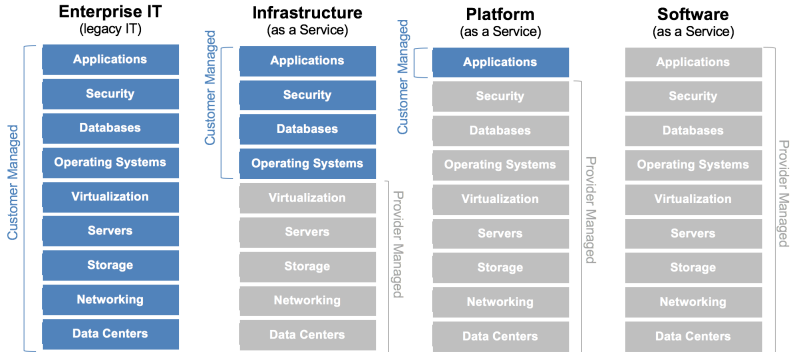
Motivations

IaaS deployment

- Different traffic loads: upward and downward *scalability*
- Costs: no acquisition costs, *pay-as-you-go*
- Security: physical security
- Availability: redundancy in hw and configurations, automatic backup

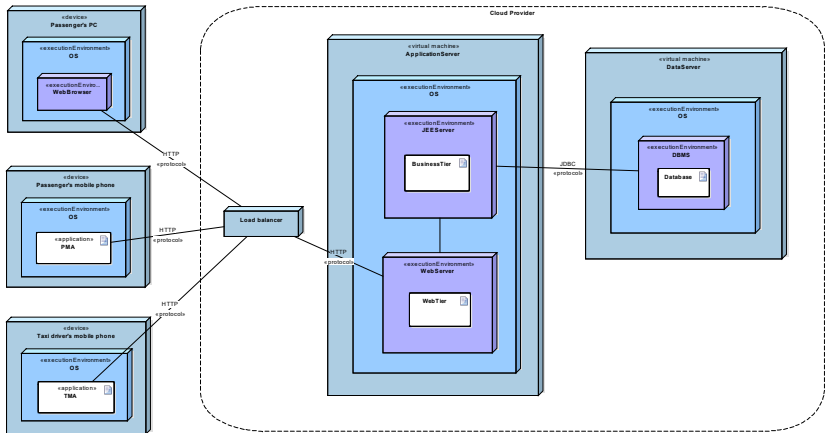


Cloud deployment Comparison



Cloud deployment

Deployment diagram



Outline

- 1 Introduction
 - Pattern + Style = Architecture
 - JEE
- 2 Architectural design
 - Component view
 - Deployment view
 - Runtime view
- 3 Algorithm design
 - Taxi queue management
- 4 User interface design

Request

Sequence diagram

Link

Reservation

Sequence diagram

Link

Outline

- 1 Introduction
 - Pattern + Style = Architecture
 - JEE
- 2 Architectural design
 - Component view
 - Deployment view
 - Runtime view
- 3 **Algorithm design**
 - Taxi queue management
- 4 User interface design

Taxi queue management

The problem - 1

- Z set of zones
- N number of available taxis at the moment
- n_i number of requests per minute in zone $i \in Z$
- q_i actual number of available taxis in zone $i \in Z$
- t_i suitable number of available taxis in zone $i \in Z$

$$t_i = \frac{n_i}{\sum_i n_i} N$$

- $t_{i,min}$ minimum acceptable number of available taxis in zone $i \in Z$

$$t_{i,min} = 0.7 t_i$$

Taxi queue management

The problem - 2

Given the current distribution of taxis among the zones
 $Q = \{q_i | i \in Z\}$ how can we move a set of taxis in order to satisfy
the *demand constraint* $q_i \geq t_{i,min} \forall i \in Z$ **minimizing** the total
number of zones traveled?

Taxi queue management

Analysis - 1

Let's partition Z in $\{Z_+, Z_-\}$

- $Z_+ = \{i \in Z \mid q_i \geq t_{i,min}\}$ zones with more taxis than needed.
- $Z_- = \{i \in Z \mid q_i < t_{i,min}\}$ zones with less taxis than needed.
- d_{ij} distance between (number of zones in between) zone $i \in Z_+$ and zone $j \in Z_-$.

Taxi queue management

Analysis - 2

x_{ij} number of taxis to be moved from zone $i \in Z_+$ and zone $j \in Z_-$ (decision variables).

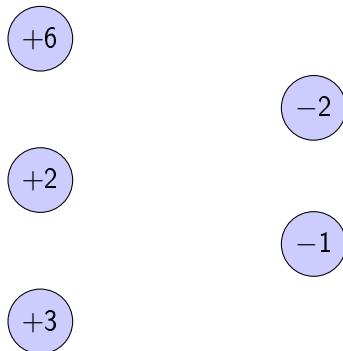
$$\min \sum_{i \in Z_+} \sum_{j \in Z_-} x_{ij} d_{ij}$$

s.t.

- $x_{ij} \geq 0 \forall i \in Z_+, j \in Z_-$ (non negativity constraint);
- $q_i - \sum_{j \in Z_-} x_{ij} \geq t_{i,min} \forall i \in Z_+$ (availability constraint);
- $q_j + \sum_{i \in Z_+} x_{ij} \geq t_{j,min} \forall j \in Z_-$ (demand constraint);
- x_{ij} integer $\forall i \in Z_+, \forall j \in Z_-$.

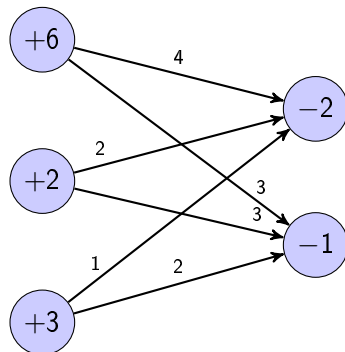
Taxi queue management

Example - 1



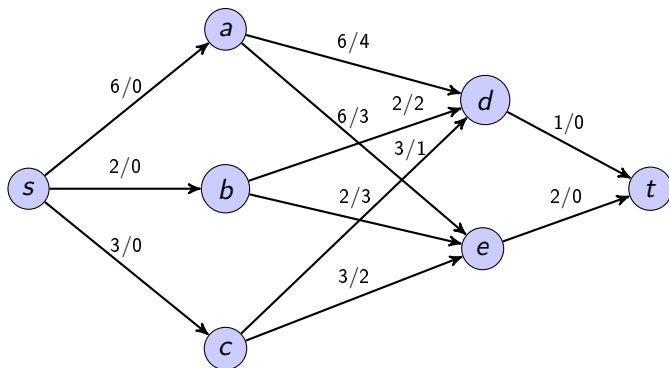
Taxi queue management

Example - 2



Taxi queue management

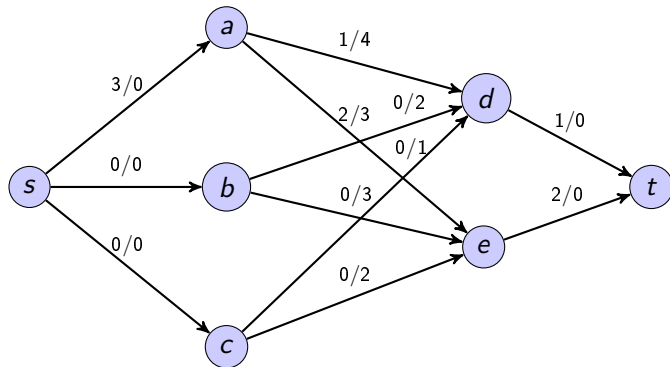
Example - Network



$$k_{ij}/d_{ij}$$

Taxi queue management

Example - a maximum flow found with Edmonds-Karp

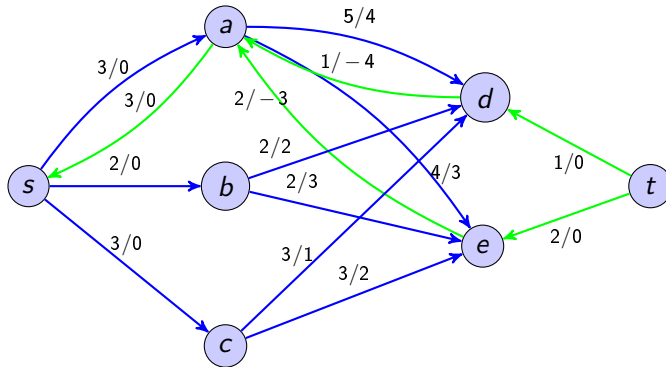


$$x_{ij}/d_{ij} \quad \sum_{i \in Z_+} \sum_{j \in Z_-} x_{ij} d_{ij} = 10$$

Taxi queue management

Example - Minimum cost flow - Incremental network

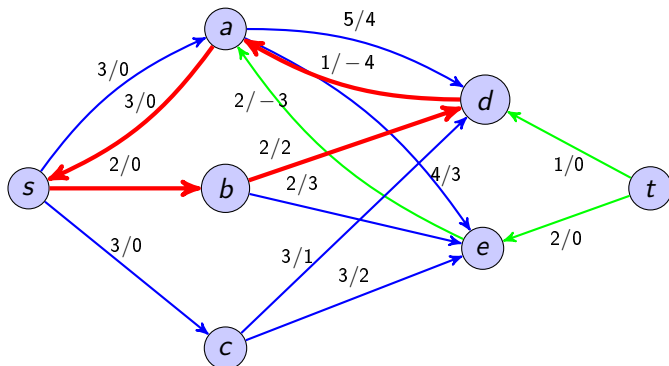
Let's have a look at the residual network. Could we find a negative cost cycle?



Taxi queue management

Example - Minimum cost flow - Negative cost canceling

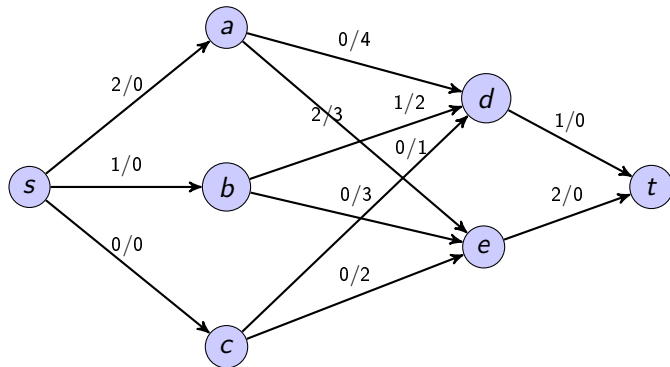
Yes! $s - b - d - a - s$



$$\overline{k_{ij}}/\overline{d_{ij}} \quad \delta = \min\{2, 4, 1, 3\} = 1$$

Taxi queue management

Example - New maximum flow with smaller cost



$$x_{ij}/d_{ij} \quad \sum_{i \in Z_+} \sum_{j \in Z_-} x_{ij} d_{ij} = 8 < 10$$

Taxi queue management

Algorithm

- ① Find a maximum flow
- ② While there exists a negative cost cycle
 - ① Build the incremental network
 - ② Find the negative cost cycle
 - ③ Update the flow

The complexity is $O(|Z||A'|^2 k_{max} d_{max})$

With some refining (minimum mean cycle)
 $O(|Z|^2 |A'|^2 \log(|Z| d_{max}))$

Taxi queue management

Algorithm

- 1 Find a maximum flow
- 2 While there exists a negative cost cycle
 - 1 Build the incremental network
 - 2 Find the negative cost cycle
 - 3 Update the flow

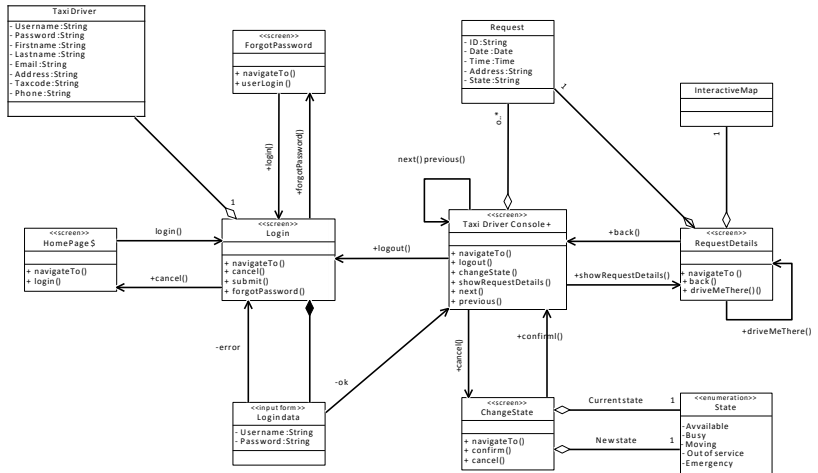
The complexity is $O(|Z||A'|^2 k_{max} d_{max})$

With some refining (minimum mean cycle)

$O(|Z|^2 |A'|^2 \log(|Z| d_{max}))$

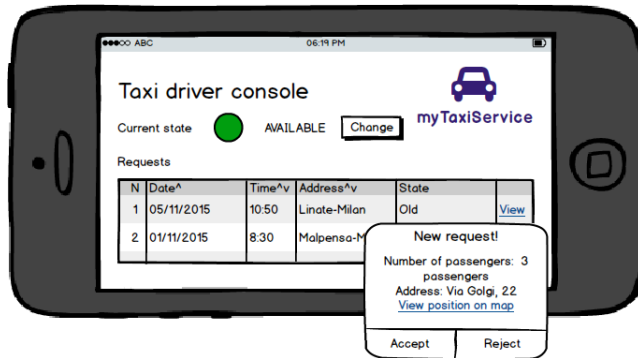
TMA interface design

UX diagram



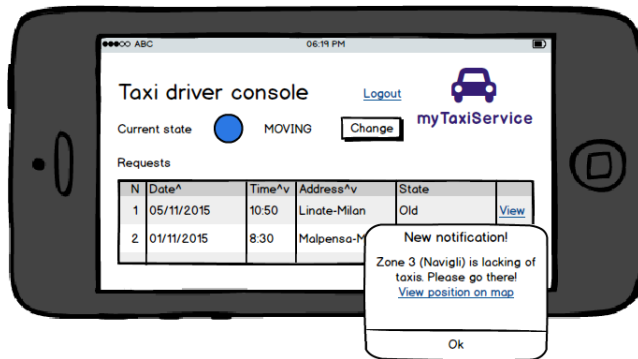
TMA interface design

Mockups - 1



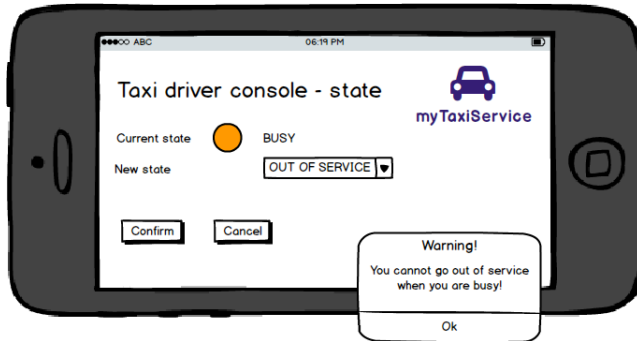
TMA interface design

Mockups - 2



TMA interface design

Mockups - 3



References

- IEEE Software Engineering Standards Committee, “IEEE Standard for Information Technology - Systems Design - Software Design Descriptions”, IEEE Std 1016TM-2009 (Revision of IEEE Std 1016-1998).
- ISO/IEC/ IEEE 42010 “Systems and software engineering - Architecture description”, First edition 2011-12-01.
- Software Architecture: Foundations, Theory, and Practice. Richard N. Taylor, Nenad Medvidovic, Eric Dashofy.
- Software Engineering 2 course slides.
- Federico Malucelli, Lecture notes.
- RASD (Requirements Analysis and Specification Document) of the *myTaxiService*.

Questions

