Advanced Software Engineering (LAB)

Stefano Forti

name.surname@di.unipi.it

Department of Computer Science, University of Pisa

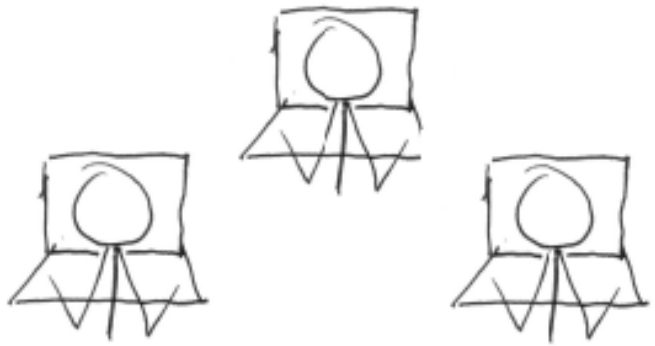

Q
&
A

Previously on ASE…

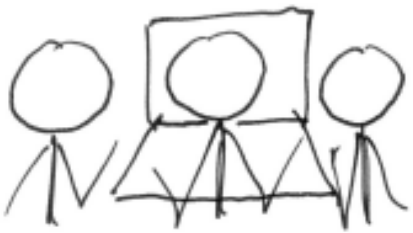
Is Teamwork Important?

- Most real-world software can't be developed by one person in a reasonable amount of time.
- So, teams are needed.
- The problem is... if the team doesn't work well *together* then the project will fail.
- It is not the team leader's responsibility to make the team work well, it is the entire team's responsibility to make the team work well.
- Succeed together or fail together.
- Do matter how good a programmer you are, most companies will not hire you, if you can't work well in a team.

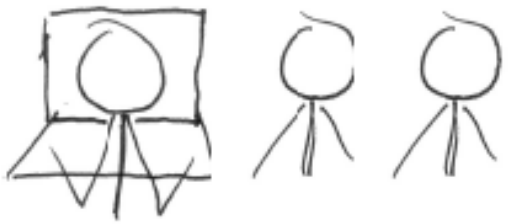
Types of Teams



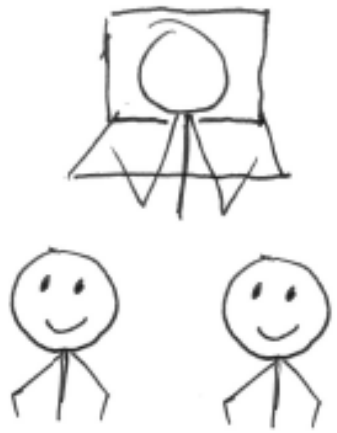
Divide And Conqueror



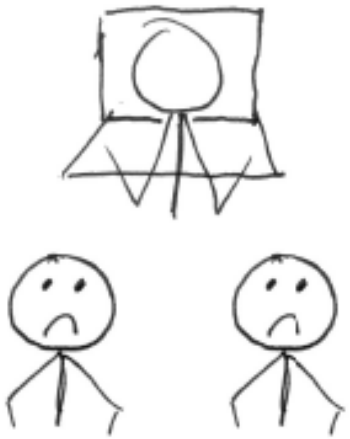
Trio



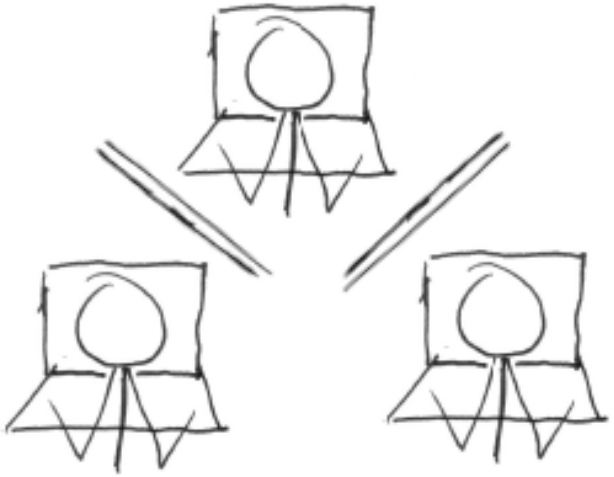
Relay



Drop Outs



One Man Band



Every Man for Himself

Characteristics of Good Teams

- Commitment
- Participation
- Communication
- Trust
- Respect
- Support
- Effective decision making
- Fun

How to?

- Commitment
  - Work hard: Don’t expect others to do what you won’t do yourself
  - Want the project to succeed
- Participation
  - Include everyone in discussions
  - Ask people’s opinion
  - Don’t be shy
  - Understand why it is being done this way
  - Don’t zone out
  - Who is doing what? Clearly assign tasks and/or roles. Rotate them.
- Communication
  - Regular discussions - meeting, email, skype, call, messaging
  - Explain why things were done a certain way
  - Discuss, don’t argue
  - Keep to the point
  - Give you opinion, provided it is constructive
  - Be responsive, e.g. “Can’t do it now, will have it done by 9.”
- Respect
  - Be on time
  - Listen to others and consider their opinion
- Trust
  - Do what you said you will do, when you said you will do it.
- Support
  - Help each other
  - Offer to help
  - Allow space for each other to work
- Effective decision making
  - Be aware of when a decision is needed (and when it isn’t)
  - Sometimes a poor decision is better than no decision
  - Have reasons for your decisions
  - Write down your decisions in an email
- Fun
  - Meetings over coffee
  - Have a laugh
  - Rewards when something if finished or working

Meetings

- What is the objective of the meeting (goals)?
- What topics do you need to talk about (agenda)?
- How long will each item take?
- What preparation is needed?
- When & where

- Hold the meeting
- Write down the findings (during the meeting)
- Write down the action points (during the meeting)
- Review the agenda & objectives, is everything covered?
- Email the minutes

Resolving Arguments

- Kick for touch
- Cool off
- Think through your and the other person's point of view
- Circle back with a facilitator
- Disentangle the argument
  - state the points of agreement
  - discuss the points of disagreement
  - get to the core reasons of why

All I Really Need To Know I Learned In Kindergarten

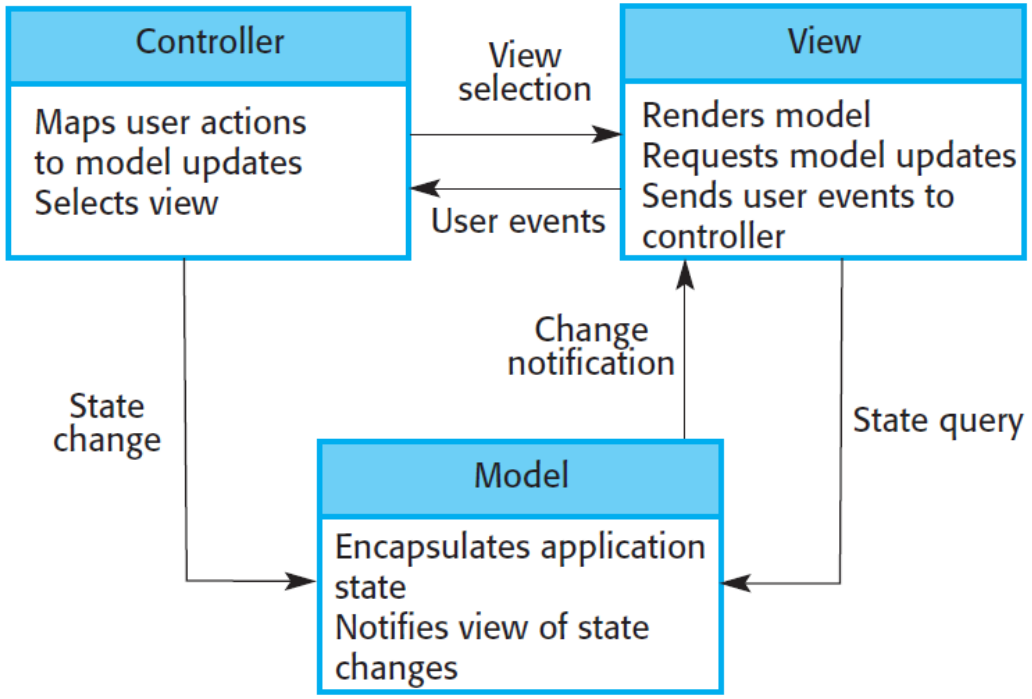
Most of what I really need
To know about how to live
And what to do and how to be
I learned in kindergarten.
Wisdom was not at the top
Of the graduate school mountain,
But there in the sandpile at Sunday school.

These are the things I learned:

Share everything.
Play fair.
Don't hit people.
Put things back where you found them.
Clean up your own mess.
Don't take things that aren't yours.
Say you're sorry when you hurt somebody.
Wash your hands before you eat.
Flush.
Warm cookies and cold milk are good for you.
Live a balanced life -
Learn some and think some
And draw and paint and sing and dance
And play and work everyday some.
Take a nap every afternoon.
When you go out into the world,
Watch out for traffic,
Hold hands and stick together.
Be aware of wonder.

by Robert Fulghum

Model-View-Controller

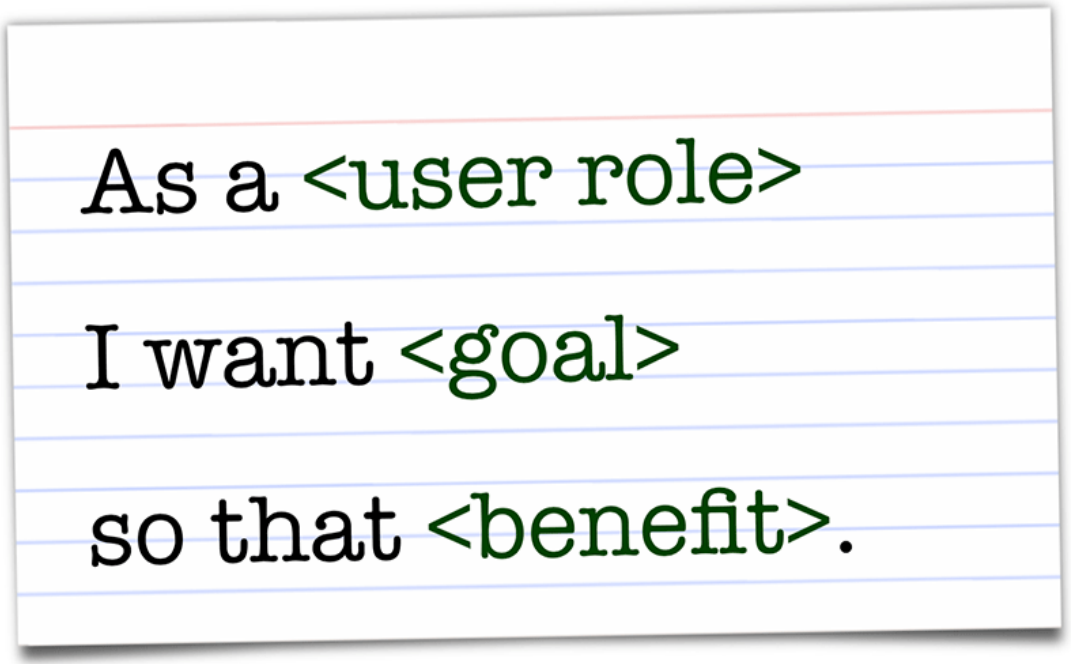


- **Model** – manages the data
- **View** – displays the Model for a particular context (e.g. web view, PDF)
- **Controller** – manipulates the Model to change its state

Dice with Microservices

Dice with Microservices is a social network for playing with storyteller dice sets.

User Stories



- Simple descriptions of interactions users have with an application, usually written when a project starts.

As a	writer or reader
I want to	sign up
So that	I can join the app community
Priority	1.1

As a	writer or writer
I want to	login
So that	I can access the app
Priority	1.2

As a	writer or reader
I want to	log out
So that	I stop using the app
Priority	1.3

As a	writer
I want to	have my wall
So that	everybody can see my profile info
Priority	1.4

As a	writer
I want to	roll the dice
So that	I can write a story
Priority	1.5

As a	writer
I want to	write a story on the rolled dice
So that	I can post it
Priority	1.6

As a	reader
I want to	see list of all stories
So that	I can read them
Priority	1.7

As a	reader
I want to	open a story
So that	I can see all info about it (dice roll, (dis)likes…)
Priority	1.8

As a	reader
I want to	add my like to a story
So that	I can be social
Priority	1.9

As a	reader
I want to	add my dislike to a story
So that	I can be social
Priority	1.9

As a	writer
I want to	the app to check whether my story is valid
So that	I can post only valid stories
Priority	2.1

As a	writer
I want to	choose the number of dice for my story
So that	i can change the difficulty of the game
Priority	2.1

As a	writer
I want to	to access my wall
So that	I can see my stories and my stats (e.g., #/avg/ratio of likes/dislikes, avg dice per story, …)
Priority	2.2

As a	writer
I want to	delete one of my stories
So that	I can delete the ones I want
Priority	2.2

As a	reader
I want to	to see a list of all writers and their last story
So that	I can decide to visit their walls
Priority	2.3

As a	reader
I want to	see the list stories written in a custom time period
So that	I can browse a subset of all stories
Priority	2.4

As a	reader
I want to	read a random recent story
So that	I can see new stuff
Priority	2.4

As a	reader
I want to	to see the wall of a given writer
So that	I can read all her stories
Priority	2.4

As a	reader
I want to	follow/unfollow a writer
So that	I can create my net
Priority	2.5

As a	writer
I want to	see a list my followers
So that	I feel better and I can visit their walls
Priority	2.5

As a	writer
I want to	select a thematic dice set
So that	I can vary more my stories
Priority	2.6

As a	reader
I want to	get a periodic email digest on the writers I follow
So that	I can what happened
Priority	3

As a	reader
I want to	to see a list of recent stories based on my interest (i.e., followed + similar + random writers)
So that	I can explore the social network
Priority	3

As a	reader
I want to	get a telegram message whenever one of the writers I follow posts a story
So that	I can immediately go and read it
Priority	3

As a	reader
I want to	to search for users and stories
So that	I can explore the social network
Priority	3

As a	writer
I want to	to write just a draft of a new story
So that	I can complete it later
Priority	3

App Components

- We give you a skeleton app implementing obfuscated stories

As a	writer or reader
I want to	sign up
So that	I can join the app community
Priority	1.1

As a	writer or writer
I want to	login
So that	I can access the app
Priority	1.2

As a	writer or reader
I want to	log out
So that	I stop using the app
Priority	1.3

As a	writer
I want to	have my wall
So that	everybody can see my profile info
Priority	1.4

As a	writer
I want to	roll the dice
So that	I can write a story
Priority	1.5

As a	writer
I want to	write a story on the rolled dice
So that	I can post it
Priority	1.6

As a	reader
I want to	see list of all stories
So that	I can read them
Priority	1.7

As a	reader
I want to	open a story
So that	I can see all info about it (dice roll, (dis)likes…)
Priority	1.8

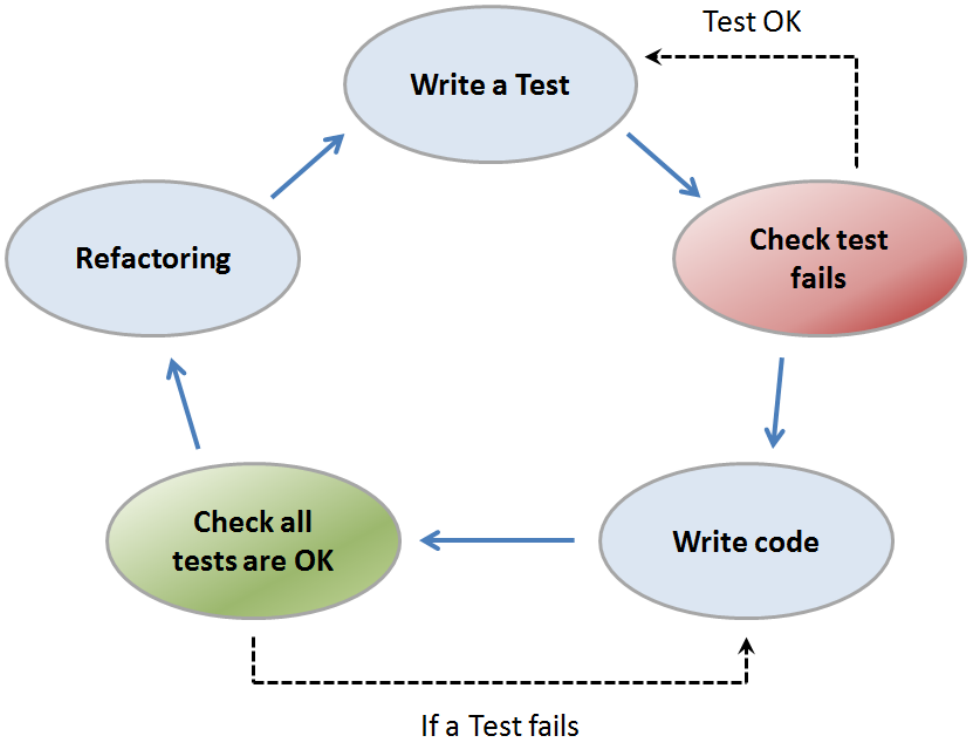
As a	reader
I want to	add my like to a story
So that	I can be social
Priority	1.9

As a	reader
I want to	add my dislike to a story
So that	I can be social
Priority	1.9

To do

- Download the primer code from the Moodle.
- Work in class (today and tomorrow) to bootstrap your group project.

- Homework (Test-Driven Development):
  - implement all High Priority stories
  - implement all Medium Priority stories, and
  - implement at least two Low Priority story



Skeleton Model

- 2 database tables:
  - `User`
  - `Story`
  - `Likes`

implemented using **Flask-SQLAlchemy**.

- If you need more, you can add other and change the existing ones tables.

SQLAlchemy

- SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.
- Used at

Flask-SQLAlchemy

http://flask-sqlalchemy.pocoo.org/2.3/

```python
from werkzeug.security import generate_password_hash, check_password_hash
import enum
from sqlalchemy.orm import relationship
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class User(db.Model):
    __tablename__ = 'user'
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    email = db.Column(db.Unicode(128), nullable=False)
    firstname = db.Column(db.Unicode(128))
    lastname = db.Column(db.Unicode(128))
    password = db.Column(db.Unicode(128))
    strava_token = db.Column(db.String(128))
    age = db.Column(db.Integer)
    weight = db.Column(db.Numeric(4, 1))
    max_hr = db.Column(db.Integer)
    rest_hr = db.Column(db.Integer)
    vo2max = db.Column(db.Numeric(4, 2))
```

- You can specify the tables using `Model` as base class.
- Flask-SQLAlchemy wraps all calls to SQLAlchemy and exposes a session object to your Flask app views to manipulate the model.

Skeleton View

- When a request is received, and a view is invoked, SQLAlchemy sets up a DB session object inside an application context.
- We use **Jinja** functions (embedded in Flask) to “compose” the view.

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/users')
def users():
    users = db.session.query(User)
    return render_template("users.html", users=users)

if __name__ == '__main__':
    db.init_app(app)
    db.create_all(app=app)
    app.run()
```

http://jinja.pocoo.org/docs/2.10/templates/

- Jinja2 is a full featured template engine for Python
- Flask incorporates Jinja and helpers like `render_template`.
- It can format e-mails too.

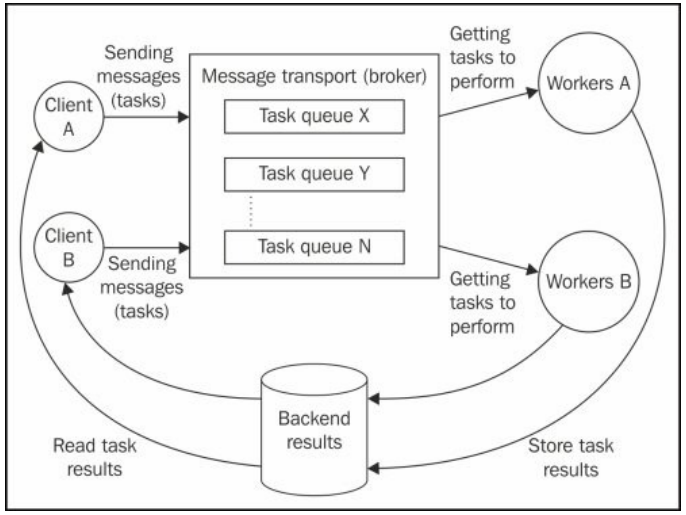
```
<html>
  <body>
    <h1>User List</h1>
    <ul>
      {% for user in users: %}
      <li>
      {{user.firstname}} {{user.lastname}}
      </li>
      {% endfor %}
    </ul>
  </body>
</html>
```

Background Tasks

Who is using celery

Instagram mozilla ADD-ONS AdRoll

- Celery is an **asynchronous task queue** based on distributed message passing.
- It is focused on real-time operations but supports scheduling as well.
- The execution units, called **tasks**, are executed concurrently on a single or more worker servers using multiprocessing.



celery

'en/latest/index.html

Example code fetches runs from Strava.

- Background features run on their own **outside the request/response cycle** and use the SQLAlchemy models to do their job.
- An **intermediary message broker** oversees passing messages back and forth between the application and Celery. E.g.,


```python
from celery import Celery
from stravalib import Client
from monolith.database import db, User, Run

BACKEND = BROKER = 'redis://localhost:6379'
celery = Celery(__name__, backend=BACKEND, broker=BROKER)
_APP = None

@celery.task
def fetch_all_runs():
    global _APP
    # init [...]

    with app.app_context():
        q = db.session.query(User)
        for user in q:
            if user.strava_token is None:
                continue
            runs_fetched[user.id] = fetch_runs(user)

    return runs_fetched
```

Checklist

- A Linux distro properly installed (e.g., Ubuntu, lubuntu)
- Python and Flask.
- Redis and Celery:

```
pip install redis
```

```
pip install celery
```

The Monolith

- You should use Celery to:
  - Asynchronously compute the like, dislike and metrics counts
  - Asynchronously manage email/bot messaging if you decide to implement such stories
- To make Celery work periodically, have a look at Celery **Periodic Tasks** [http://docs.celeryproject.org/en