

# THE OPENSUBMIT PROJECT

How to grade 1200 code submissions

Dr. Peter Tröger, Operating Systems Group, TU Chemnitz

(with Frank Feinbube, Bernhard Rabe, Kai Fabian)

<https://github.com/troeger/opensubmit>

PART I: PURPOSE

PART II: TECHNOLOGY

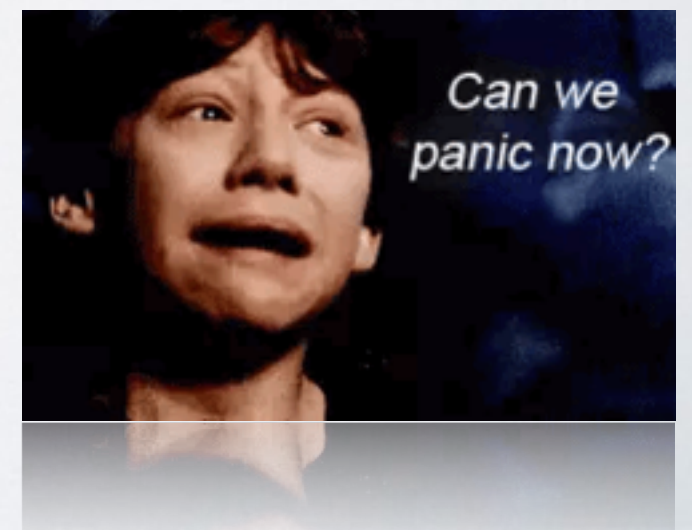
# BACKGROUND

- University researcher and teacher, 13 years of experience
- Courses from 5 to 6000 participants
- Winter 2015/16:  
Embedded programming course with 300 students,  
~1200 code submissions to grade
- OpenSubmit developed + used at  
Hasso Plattner Institute Potsdam and TU Chemnitz
- Replaced Roundup-based solution (issue tracker)



# THE ENVIRONMENT

- Homework assignments („Übungsblätter“)
  - Often more important than the lecturing
  - Students develop text, equations or **code**
  - Written / printed paper, also e-mail and file transfer
- Tutors provide comments (and grade)
- May impact final course grade



## Submission Instructions

To submit your homework, create a folder named **lastname\_firstinitial\_hw#** and place your IPython notebooks, data files, and any other files in this folder. Your IPython Notebooks should be completely executed with the results visible in the notebook. We should not have to run any code. Compress the folder (please use .zip compression) and submit to the CS109 dropbox in the appropriate folder. *If we cannot access your work because these directions are not followed correctly, we will not grade your work.*

## Labs

## Electronic Submission

To submit homeworks in COMP212, you will run a command line script that will send the code (and its subdirectories) to the comp212 account. There are some points on which the program expects instructions **exactly**.

1. Log onto an owl net machine (such as the computers in Ryon).
2. It is easiest for the following steps if you `cd` to the directory right above the directory your files are in `~/comp212/hw1`, then type `"cd ~/comp212"`.
3. Run the **turnin** script with the project name given for the homework. This project name is given on the assignment page. The format of the script is as follows:  
`turnin -c comp212 -p projname dirname`  
 where *projname* is the project name given on the assignment page and *dirname* is the directory name, "hw1" in the above example.

**Important:** Do not put a trailing slash ("/") on the directory name, as this will cause the program to fail to work correctly.

Also note that projects with more than one submission ("milestones") will have unique project names. Do not submit a later part of the project into the earlier project name.

All homework must be submitted on Turing using the `cs70submit` program. You may develop your code on any system you choose, but we will be testing it on Turing using the `g++ -Wall -pedantic`. Each assignment will require you to submit two or more files with specified names. One of these files will always be named `README`, and will contain the English-language documentation for the program. The other file(s) will contain the program itself. The `README` file will usually have a later deadline than the other files.

Unless otherwise specified, you *must* use the filenames specified in the assignment, or points will be deducted from your grade. The only variation allowed is in file extensions: although the assignments will specify the `.cc` extension (e.g., `median.cc`), we will allow any extension accepted by `g++` (i.e., `.cc`, `.C`, `.cxx`, `.cpp`, or `.c++`).

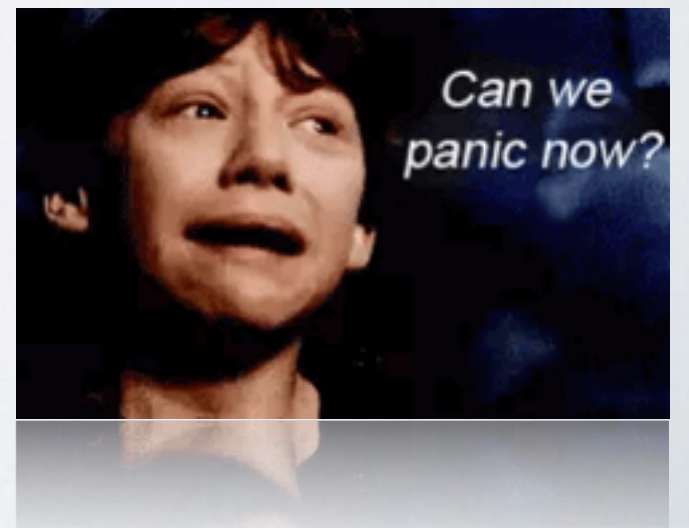


# PROBLEM 1: FORMATS

- Students are way too creative
  - C code as scanned picture or .docx
  - Visual Studio / Eclipse project folder mess
  - Archive formats you never heard about (.s7z) or don't want (.rar)
  - Endless dependencies on third-party header files
  - Complete repos or third-party sources included

# PROBLEM 2: REPRODUCIBILITY

- „Must be your fault, it definitely worked on my laptop.“
- „I have this weird compiler message, can you fix it?“
- „I submitted the wrong file, can you replace `foo.c` with `bar.cpp`?“
- „We rely on `strangelib.so`. It's written in `NoTeS/readme.docx.zip`“
- Tasks with special hardware (e.g. OpenCL)
- Test machine on the evening before deadline ...



# PROBLEM 3: GRADING


- Coordinate work of the teaching team
- Keep students informed about grades and comments
- Handle re-submissions and submission corrections
- Manage database of final scores
- Deal with cheaters
  - Copying of code on test machines
  - Stack-Overflow'ing







# MOODLE?

- Moodle (and most other LMSs) are all-inclusive
  - Online teaching, participant management, grading books, collaboration, file management, calendar, ....
  - Course owners need training. Seriously?
  - Assignments are just a small part.
  -  Plugins for large PHP projects. Not fun.
- OpenSubmit is focussing on assignments **only**.

# OPENSUBMIT PRINCIPLES

- **I) Minimalism**

- Don't let teaching policies live in code
  - Assignment rules vary widely in different institutions and groups
  - Example: When and how form student groups.
  - Create a tool for humans to implement these policies
- Reduced student interface.
- Clear workflows for teachers.

# STUDENT UI

OpenSubmit [Dashboard](#) [Courses](#) [Settings](#) [Logout](#)

## New Submission

CLT 2016 Presentation (Verteilte Betriebssysteme WS15/16)

**Please note:**  
You can find further information in the [assignment description](#).

**Notes**

**Attachment**  
[Datei auswählen](#) Keine Datei ausgewählt

[Save](#)

OpenSubmit [Courses](#) [Settings](#) [Logout](#)

## Dashboard

Peter Tröger <peter.troeger@informatik.tu-chemnitz.de>

### Open Assignments

Course	Assignment	Start	Deadline	Time Left	Action
<a href="#">Verteilte Betriebssysteme WS15/16</a>	<a href="#">CLT 2016 Presentation</a>	March 18, 2016, 8:55 a.m.	March 19, 2016, 11 a.m.	1 day, 2 hours	<a href="#">+ New Submission</a>

### Active Submissions

ID	Course	Assignment	Submitted by	Authors	Status	Action
950	<a href="#">SWES WS15/16</a>	<a href="#">Assignment 1, Task 4</a>	Peter Tröger	Peter Tröger (peter.troeger@...) OSG Submit Admin (peter.troeger@...)	Graded	<a href="#">Details</a>
385	<a href="#">SWES WS15/16</a>	<a href="#">Assignment 1, Task 3</a>	Peter Tröger	Peter Tröger (peter.troeger@...) OSG Submit Admin (peter.troeger@...)	Graded	<a href="#">Details</a>



# OPENSUBMIT PRINCIPLES

- **II) Teacher-driven development (TDD)**
  - Develop capabilities ,on-the-fly', focus on needs of correctors
  - People sometimes call it user-centric, agile or design thinking
  - Your development process must fit to that



Teacher Backend

Teacher Backend

WELCOME, PETER. [VIEW SITE](#) / [LOG OUT](#)

Home • Teacher Backend • Submissions

### TEACHER BACKEND

- Assignments [+ Add](#) [Change](#)
- Courses [+ Add](#) [Change](#)
- Grading schemes [+ Add](#) [Change](#)
- Gradings [+ Add](#) [Change](#)
- Submission files [+ Add](#) [Change](#)
- Submissions [+ Add](#) [Change](#)
- Test machines [+ Add](#) [Change](#)

SWES WS15/16

Open assignments: 0

Submissions to be graded: 0

Grading finished, not notified: 0

Registered students: 301

Authoring students: 285

[Show grading table](#)

[eMail to students](#)

[Download course archive](#)

### Grading Table - SWES WS15/16

Select submission to change

Q  Search

Action:  Go 0 of 100 selected

	SUBMISSION	CREATED	SUBMITTER	AUTHORS	COURSE	ASSIGNMENT	STATE	GRADING	GRADING NOTES	GF
<input type="checkbox"/>	3044	Jan. 29, 2016, 2 a.m.			SWES WS15/16	Assignment 5, Task 4	Closed, student notified	3		
<input type="checkbox"/>	3043	Jan. 29, 2016, 1:25 a.m.			SWES WS15/16	Assignment 5, Task 3	Closed, student notified	3		
<input type="checkbox"/>	3042	Jan. 29, 2016, 1:18 a.m.			SWES WS15/16	Assignment 5, Task 3	Withdrawn	-		
<input type="checkbox"/>	3041	Jan. 29, 2016, 1:04 a.m.			SWES WS15/16	Assignment 5, Task 3	Withdrawn	-		
<input type="checkbox"/>	3040	Jan. 29, 2016, 12:52 a.m.			SWES WS15/16	Assignment 5, Task 3	Withdrawn	-		
<input type="checkbox"/>	3039	Jan. 29, 2016, 12:26 a.m.			SWES WS15/16	Assignment 5, Task 1	Closed, student notified	3		
<input type="checkbox"/>	3038	Jan. 29, 2016, 12:08 a.m.			SWES WS15/16	Assignment 5, Task 4	Closed, student notified	0		
<input type="checkbox"/>	3037	Jan. 29, 2016, 12:06 a.m.			SWES WS15/16	Assignment 5, Task 4	Withdrawn	-		

#### FILTER

By Submission Status

[All](#)

[To be graded](#)

[Grading not finished](#)

[Grading finished](#)

[Closed](#)

By Course

[All](#)

[SWES WS15/16](#)

[Verteilte Betriebssysteme WS15/16](#)

By Assignment

[All](#)

[Assignment 1, Task 1](#)

[Assignment 1, Task 2](#)

[Assignment 1, Task 3](#)

[Assignment 1, Task 4](#)

[Assignment 2](#)

[Assignment 3, Task 1](#)

[Assignment 3, Task 2](#)

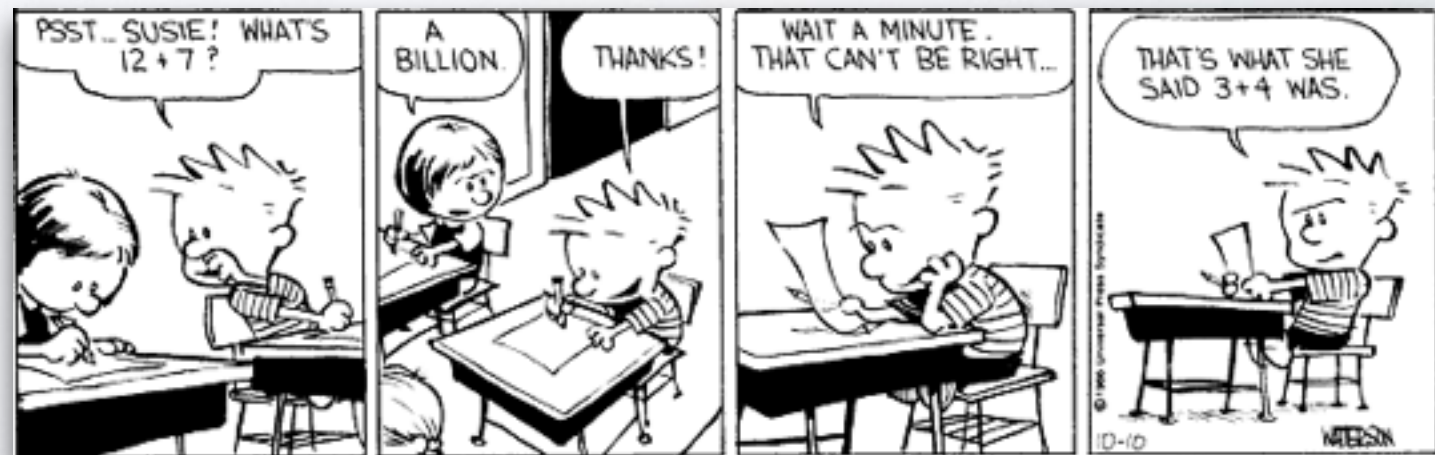
[Assignment 3, Task 3](#)

[Assignment 4, Task 1](#)

Select course to change						Select course to change									
Action: <input type="text"/> 0 of 23 selected						Search: <input type="text"/>									
Course	Active	Owner	Assignments	Max authors	Actions	First Name	Student ID	Assignment 1, Task 1	Assignment 1, Task 2	Assignment 1, Task 3	Assignment 1, Task 4	Assignment 2	Assignment 3, Task		
Datenbanksysteme I SS16		Thorsten Papenbrock (thorsten.papenbrock@...)		3	Download course archive Show grading table email to students			1	3	3	3	4	5		
Datenbanksysteme I SS15/16		Maximilian Jenders (maximilian.jenders@...)	DBS II WS2015 Aufgabe 1.1, DBS II WS2015 Aufgabe 1.2, DBS II WS2015 Aufgabe 1.3, DBS II WS2015 Aufgabe 1.4, DBS II WS2015 Aufgabe 1.5, DBS II WS2015 Aufgabe 1.6, DBS II WS2015 Aufgabe 1.7, DBS II WS2015 Aufgabe 1.8, DBS II WS2015 Aufgabe 1.9, DBS II WS2015 Aufgabe 1.10, DBS II WS2015 Aufgabe 1.11, DBS II WS2015 Aufgabe 1.12, DBS II WS2015 Aufgabe 1.13, DBS II WS2015 Aufgabe 1.14, DBS II WS2015 Aufgabe 1.15, DBS II WS2015 Aufgabe 1.16, DBS II WS2015 Aufgabe 1.17, DBS II WS2015 Aufgabe 1.18, DBS II WS2015 Aufgabe 1.19, DBS II WS2015 Aufgabe 1.20, DBS II WS2015 Aufgabe 1.21, DBS II WS2015 Aufgabe 1.22, DBS II WS2015 Aufgabe 1.23, DBS II WS2015 Aufgabe 1.24, DBS II WS2015 Aufgabe 1.25, DBS II WS2015 Aufgabe 1.26, DBS II WS2015 Aufgabe 1.27, DBS II WS2015 Aufgabe 1.28, DBS II WS2015 Aufgabe 1.29, DBS II WS2015 Aufgabe 1.30, DBS II WS2015 Aufgabe 1.31, DBS II WS2015 Aufgabe 1.32, DBS II WS2015 Aufgabe 1.33, DBS II WS2015 Aufgabe 1.34, DBS II WS2015 Aufgabe 1.35, DBS II WS2015 Aufgabe 1.36, DBS II WS2015 Aufgabe 1.37, DBS II WS2015 Aufgabe 1.38, DBS II WS2015 Aufgabe 1.39, DBS II WS2015 Aufgabe 1.40, DBS II WS2015 Aufgabe 1.41, DBS II WS2015 Aufgabe 1.42, DBS II WS2015 Aufgabe 1.43, DBS II WS2015 Aufgabe 1.44, DBS II WS2015 Aufgabe 1.45, DBS II WS2015 Aufgabe 1.46, DBS II WS2015 Aufgabe 1.47, DBS II WS2015 Aufgabe 1.48, DBS II WS2015 Aufgabe 1.49, DBS II WS2015 Aufgabe 1.50, DBS II WS2015 Aufgabe 1.51, DBS II WS2015 Aufgabe 1.52, DBS II WS2015 Aufgabe 1.53, DBS II WS2015 Aufgabe 1.54, DBS II WS2015 Aufgabe 1.55, DBS II WS2015 Aufgabe 1.56, DBS II WS2015 Aufgabe 1.57, DBS II WS2015 Aufgabe 1.58, DBS II WS2015 Aufgabe 1.59, DBS II WS2015 Aufgabe 1.60, DBS II WS2015 Aufgabe 1.61, DBS II WS2015 Aufgabe 1.62, DBS II WS2015 Aufgabe 1.63, DBS II WS2015 Aufgabe 1.64, DBS II WS2015 Aufgabe 1.65, DBS II WS2015 Aufgabe 1.66, DBS II WS2015 Aufgabe 1.67, DBS II WS2015 Aufgabe 1.68, DBS II WS2015 Aufgabe 1.69, DBS II WS2015 Aufgabe 1.70, DBS II WS2015 Aufgabe 1.71, DBS II WS2015 Aufgabe 1.72, DBS II WS2015 Aufgabe 1.73, DBS II WS2015 Aufgabe 1.74, DBS II WS2015 Aufgabe 1.75, DBS II WS2015 Aufgabe 1.76, DBS II WS2015 Aufgabe 1.77, DBS II WS2015 Aufgabe 1.78, DBS II WS2015 Aufgabe 1.79, DBS II WS2015 Aufgabe 1.80, DBS II WS2015 Aufgabe 1.81, DBS II WS2015 Aufgabe 1.82, DBS II WS2015 Aufgabe 1.83, DBS II WS2015 Aufgabe 1.84, DBS II WS2015 Aufgabe 1.85, DBS II WS2015 Aufgabe 1.86, DBS II WS2015 Aufgabe 1.87, DBS II WS2015 Aufgabe 1.88, DBS II WS2015 Aufgabe 1.89, DBS II WS2015 Aufgabe 1.90, DBS II WS2015 Aufgabe 1.91, DBS II WS2015 Aufgabe 1.92, DBS II WS2015 Aufgabe 1.93, DBS II WS2015 Aufgabe 1.94, DBS II WS2015 Aufgabe 1.95, DBS II WS2015 Aufgabe 1.96, DBS II WS2015 Aufgabe 1.97, DBS II WS2015 Aufgabe 1.98, DBS II WS2015 Aufgabe 1.99, DBS II WS2015 Aufgabe 1.100	2	Download course archive Show grading table email to students										
Project Seminar: Parallel and Distributed Systems		Frank Feinbue (frank.feinbue@...)	Task 1.2, Task 2.1, Task 1.3, Task 1.1, Task 2.3, Task 4.1, Task 2.2, Task 4.2, Task 4.3, Task 3.2, Task 3.3, Task 3.1, Task 5.1, Task 6.1, Task 6.3, Task 6.2	1	Download course archive Show grading table email to students			0	3	3	2	4	0		
Betriebssysteme I WS15/16		Bernhard Rabe (bernhard.rabe@...)	Aufgabe 5.2, Aufgabe 4.4 Windows, Aufgabe 3.4 Linux, Aufgabe 3.4 Windows, Aufgabe 3.2, Aufgabe 2.5b, Aufgabe 2.5a, Aufgabe 1.4b, Aufgabe 1.4a, Aufgabe 1.3, Test, No official assignment!	4	Download course archive Show grading table email to students			1	3	3	3	5	4		
Information Retrieval and Web Search WS 15/16		Ralf Krestel (ralf.krestel@...)	Introduction, Text Transformation, Retrieval Models I, Index Creation, Retrieval Models, Vector Space, Probabilistic and Language Models, Evaluation, Querying, Crawling, Link Analysis, Visualizing the Search Results, Combining Evidence, Web Search, Performance, Implementation Statistics	2	Download course archive Show grading table email to students			1	3	3	3	6	5		
Mathematik 2 SS15		Sören Tietbohl (soeren.tietbohl@...)	Mat SS15 Blatt 2	1	Download course archive Show grading table email to students			1	3	2	3	6	5		
Datenbanksysteme I SS15		Thorsten Papenbrock (thorsten.papenbrock@...)	DBS SS15 Task 1.1, DBS SS15 Task 1.2, DBS SS15 Task 1.3, DBS SS15 Task 1.4, DBS SS15 Task 1.5, DBS SS15 Task 1.6, DBS SS15 Task 1.7, DBS SS15 Task 1.8, DBS SS15 Task 1.9, DBS SS15 Task 1.10, DBS SS15 Task 1.11, DBS SS15 Task 1.12, DBS SS15 Task 1.13, DBS SS15 Task 1.14, DBS SS15 Task 1.15, DBS SS15 Task 1.16, DBS SS15 Task 1.17, DBS SS15 Task 1.18, DBS SS15 Task 1.19, DBS SS15 Task 1.20, DBS SS15 Task 1.21, DBS SS15 Task 1.22, DBS SS15 Task 1.23, DBS SS15 Task 1.24, DBS SS15 Task 1.25, DBS SS15 Task 1.26, DBS SS15 Task 1.27, DBS SS15 Task 1.28, DBS SS15 Task 1.29, DBS SS15 Task 1.30, DBS SS15 Task 1.31, DBS SS15 Task 1.32, DBS SS15 Task 1.33, DBS SS15 Task 1.34, DBS SS15 Task 1.35, DBS SS15 Task 1.36, DBS SS15 Task 1.37, DBS SS15 Task 1.38, DBS SS15 Task 1.39, DBS SS15 Task 1.40, DBS SS15 Task 1.41, DBS SS15 Task 1.42, DBS SS15 Task 1.43, DBS SS15 Task 1.44, DBS SS15 Task 1.45, DBS SS15 Task 1.46, DBS SS15 Task 1.47, DBS SS15 Task 1.48, DBS SS15 Task 1.49, DBS SS15 Task 1.50, DBS SS15 Task 1.51, DBS SS15 Task 1.52, DBS SS15 Task 1.53, DBS SS15 Task 1.54, DBS SS15 Task 1.55	120	Download course archive Show grading table email to students				1	3	3		6	5	
Data Mining and Probabilistic Reasoning SS15		Ralf Krestel (ralf.krestel@...)	Data Mining Assignment II - Basics, Data Mining Assignment I - Introduction, Data Mining Assignment III - Logical Models, Data Mining Assignment IV - Association Rules Task, Data Mining Assignment IV - Geometric Models, Data Mining Assignment V - Probabilistic Models, Data Mining Assignment VI - Graphical Models, Data Mining Assignment VII - Ensemble Methods	2	Download course archive Show grading table email to students			1	3	3	3	6	5		
Data Profiling and Data Cleaning WS14/15		Thorsten Papenbrock (thorsten.papenbrock@...)	DPDC WS14 Exercise 1, DPDC WS14 Exercise 2, DPDC WS14 Exercise 3 - NEU, DPDC WS14 Exercise 4, DPDC WS14 Exercise 5	2	Download course archive Show grading table email to students			0	3	3	3				
Programmiertechnik I WS14/15		Sven Köhler (sven.koehler@...)	What is digital?, Christmas Melody, Information representation, Tiny Photoshop, Transposed Generation	2	Download course archive Show grading table 										

# EXAMPLE: DUPLICATE DETECTION

- Good students don't cheat, bad student cheat badly
- Solvable with half-smart MD5 calculation on student file uploads
  - Results in **duplicate report** for correctors (minimalism policy)
  - Added in the last semester in 2 weeks
- Future versions will rely on fuzzy content comparison

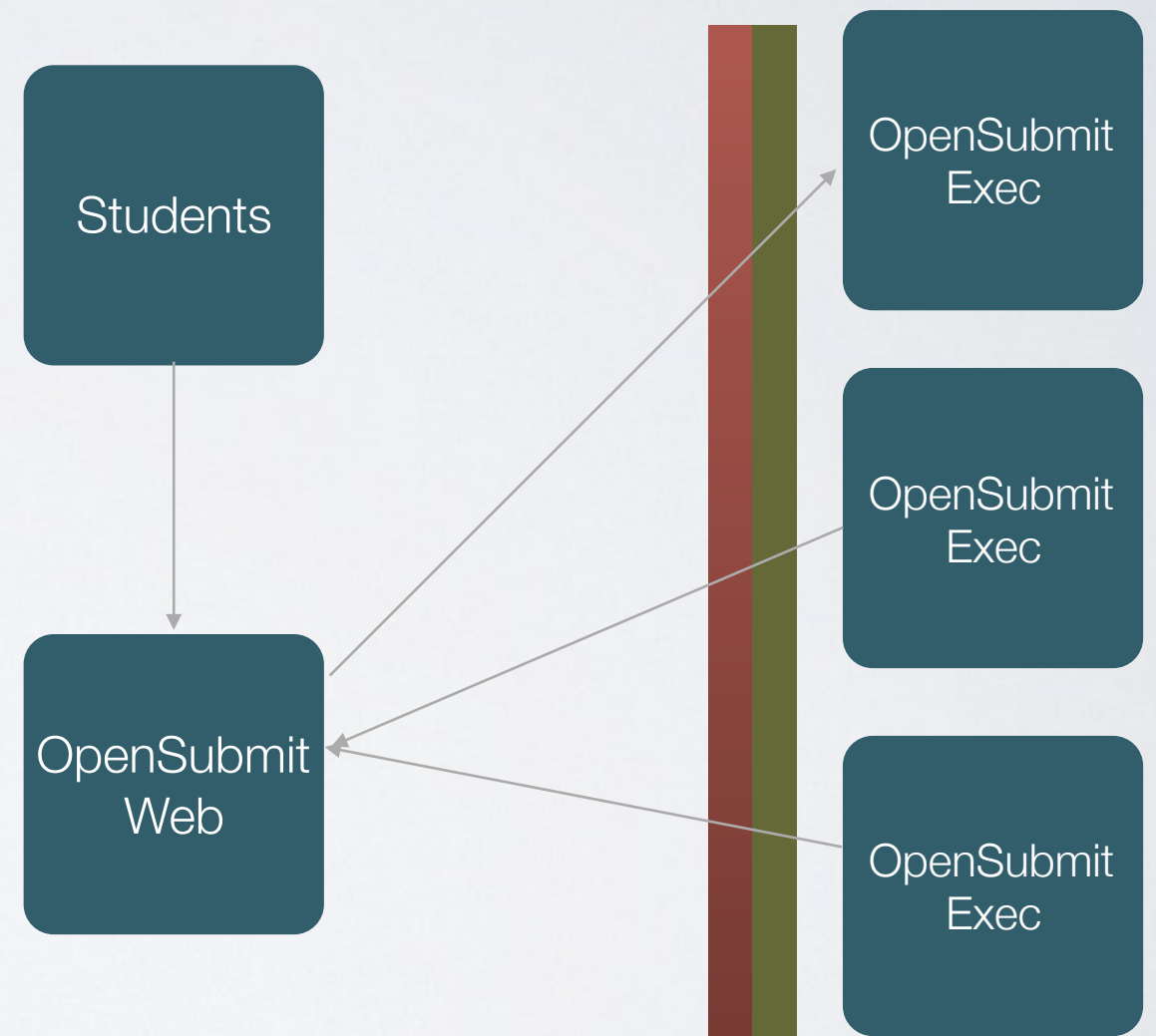




# OPENSUBMIT PRINCIPLES

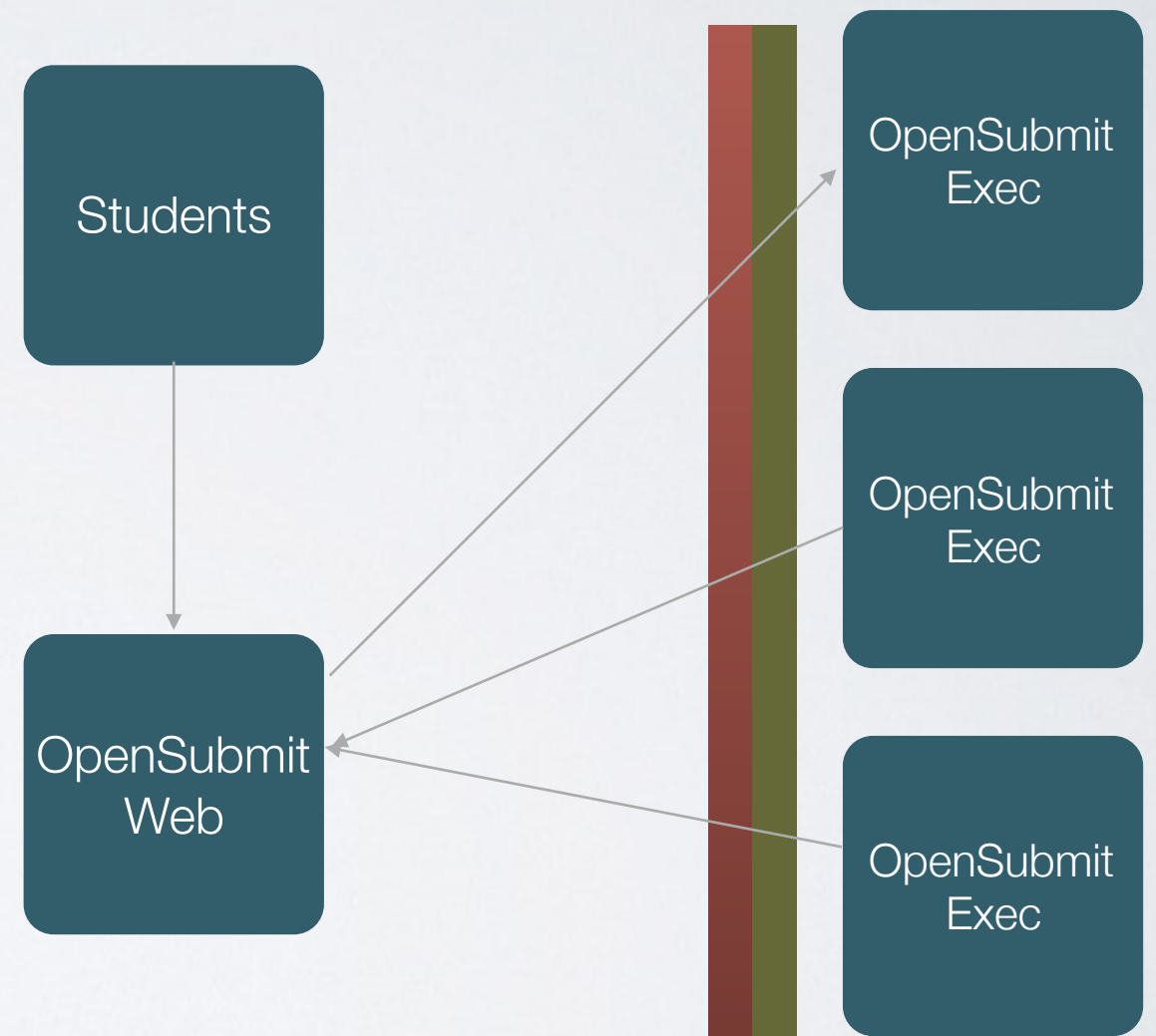
- **III) Assignment validation**

- **Compile** and **validate** student file uploads
- Different tests and tests machines per assignment
- Direct feedback, chance for withdrawal before deadline
- Dedicated **full test** for grading after deadline



# VALIDATION SCRIPT

- Something written by the assignment creator
- Executor daemon downloads the student submission and this script
- Must run the (compiled) student code and print debug output
- STDOUT shown to student, error code as success indicator
- Full test works the same way



# STUDENT UI

## Submission Details

Submission ID	950
Course	<a href="#">SWES WS15/16</a>
Assignment	<a href="#">Assignment 1, Task 4</a>
Submitter	Peter Tröger (peter.troeger@...)
Authors	Peter Tröger (peter.troeger@...), OSG Submit Admin (peter.troeger@...)
Notes by submitter	
File	<a href="#">submission_RRZ5x3w.zip</a> Compilation test result: <pre>gcc -o bitcount bitcount.c</pre> Validity test result: <pre>Congratulations! Your code seems to work.</pre>
State	Graded
Grading	3
Grading Notes	



# TEACHER UI

## Submission and test results

### Notes:

Stored upload: [bitreverse\\_CCpjYQH.zip](#) (Preview) New upload:   

Compilation test: Test output from 10.10.10.10:

```
gcc -o bitreverse bitreverse.c
```


Validation test: Test output from 10.10.10.12:

Congratulations! Your code seems to work.

Full test: Not enabled.

## Grading

Status: ☐ Grading not finished ☐ Grading finished

Grading:   

## Archive Preview

### bitreverse/bitreverse.c

```
#include<stdio.h>
int main (int argc, char *argv[])
{
    unsigned char y;
5.    int input_argument = atoi (argv[1]);

    y = 0xFF ^ input_argument;
    printf("%d",y);
    return 0;
10. }
```

### bitreverse/makefile

# ASSIGNMENT

- Relates to a course
- Has a grading scheme, start time, hard deadline and description link
- May have a **soft deadline**
- May include **compilation** of file upload
- May have a **validation script** executing the student code
- May have a **full test script** executing the student code
- Has **grading scheme** as collection of arbitrary **grading** strings

# SUBMISSION

- A **submission** is handed in by a single **student**
  - Can declare group members, no group management (minimalism)
  - Submission and results show up for all of them
- Whole submission can be explicitly **withdrawn**, not deleted
- **Grading** and **grading notes** visible when tutors triggered notification
- **Status** from student and tutor perspective looks different



# STATES

```
STATES = (                                     # States from the backend point of vi
    (RECEIVED, 'Received'),
    (WITHDRAWN, 'Withdrawn'),
    (SUBMITTED, 'Submitted'),
    (TEST_COMPILE_PENDING, 'Compilation test pending'),
    (TEST_COMPILE_FAILED, 'Compilation test failed'),
    (TEST_VALIDITY_PENDING, 'Validity test pending'),
    (TEST_VALIDITY_FAILED, 'Validity test failed'),
    (TEST_FULL_PENDING, 'Full test pending'),
    (TEST_FULL_FAILED, 'All but full test passed, grading pending'),
    (SUBMITTED_TESTED, 'All tests passed, grading pending'),
    (GRADING_IN_PROGRESS, 'Grading not finished'),
    (GRADED, 'Grading finished'),
    (CLOSED, 'Closed, student notified'),
    (CLOSED_TEST_FULL_PENDING, 'Closed, full test pending')
)

STUDENT_STATES = (                             # States from the student point of vi
    (RECEIVED, 'Received'),
    (WITHDRAWN, 'Withdrawn'),
    (SUBMITTED, 'Waiting for grading'),
    (TEST_COMPILE_PENDING, 'Waiting for compilation test'),
    (TEST_COMPILE_FAILED, 'Compilation failed'),
    (TEST_VALIDITY_PENDING, 'Waiting for validation test'),
    (TEST_VALIDITY_FAILED, 'Validation failed'),
    (TEST_FULL_PENDING, 'Waiting for grading'),
    (TEST_FULL_FAILED, 'Waiting for grading'),
    (SUBMITTED_TESTED, 'Waiting for grading'),
    (GRADING_IN_PROGRESS, 'Waiting for grading'),
    (GRADED, 'Waiting for grading'),
    (CLOSED, 'Graded'),
    (CLOSED_TEST_FULL_PENDING, 'Graded')
)
```

Status	Action
Compilation failed	Details
Graded	Details
Graded	Details
Compilation failed	Details
Graded	Details
Validation failed	Details

Assignment	State	Grading	Grading
Task 6.2	Validity test failed	(None)	⊘
Aufgabenblatt 5	Closed, student notified	Passed	✓
Aufgabenblatt 5	Closed, student notified	Passed	✓
Aufgabenblatt 5	Withdrawn	(None)	⊘
Aufgabenblatt 5	Closed, student notified	Passed	✓
Aufgabenblatt 5	Closed, student notified	Passed	✓
Task 6.3	All tests passed, grading pending	(None)	⊘

# CURRENT FEATURES

- Social login (SAML, OAuth, OpenID)
- Simple student front-end
- Compilation, validation and full test on assignment-specific test machines
- Output of compilation and validation scripts shown to students
- E-mail notification on state changes
- History of assignment file uploads
- Central overview of grading progress
- Rich tutor support:  
Archive preview, duplicate detection, full tests, filtering
- All students notified at-once after grading is completely finished
- Excel-style grading table
- (Usable) course archive download

# 1200 SUBMISSIONS?

- OpenSubmit supports teachers grading workflow
  - Trivial work sharing in the teaching team
  - Bad cheaters are already identified
  - All student code is proven to work (validation script)
  - Triggering an extra test is a mouse click away
  - Progress always visible



# AUTO GRADING?

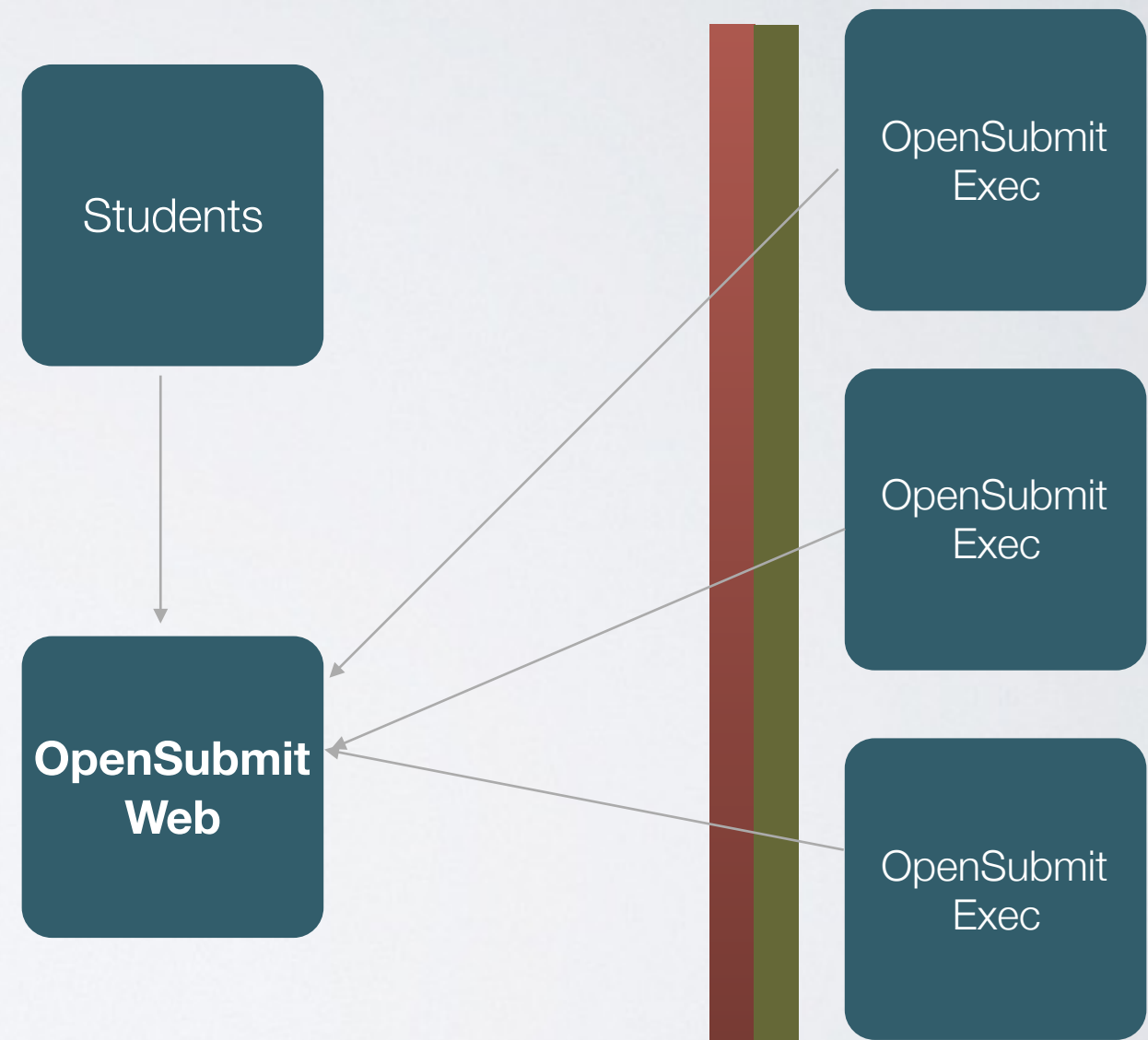
- University teachers would love that, especially with MOOCs
  - Option 1: Students develop (blindly) against a test suite
  - Option 2: Analyze code AST to derive some score
  - Option 3: Check for OS-visible behaviors on execution
  - Option 4: ???
- Very hard to generalize. Also a legal problem.
- Validation script concept allows all of them.

PART I: PURPOSE

PART II: TECHNOLOGY

# OPENSUBMIT-WEB

- Django web application
  - Started with Django 1.3, followed all migrations up to recent 1.9
  - Python 2.7 (sorry Martin), mod\_wsgi, Apache 2.4, PostgreSQL
  - Varying set of third-party code, including JavaScript libs
- Some experiences over the years





# EXPERIENCES: THIRD-PARTY CODE

- **Pro:** Other people **solve** your **existing problem**
  - Authentication, templating, API management, fancy UI, crypto, testing, ...
  - They fix the really bad bugs for you
- **Con:** Other people **create** your **new problem**
  - Your environment is not their environment
  - Even when you fix it, the pull request may take some time
  - You must be willing to understand their code, too

# EXPERIENCES: THIRD-PARTY CODE

- Development vs. integration trade-off
- The maintenance problem is always there, but in different flavors
- Example: django-reversion vs. own model implementation
- Example: Django REST framework vs. custom HTTP API
- Lessons learned
  - Popularity and age are good indicators ([djangopackages.com](http://djangopackages.com))
  - Avoid solutions with „advanced magic“

# EXPERIENCES: DATA MIGRATION

- Database migration needed on model changes
  - South was ok for that, Django  $\geq 1.7$  support is even better
  - They handle model migration, but **data migration is your problem**
- Lessons learned
  - Create migration-friendly data models. You have only one try.
  - Data attributes may later become 1:N relationships.
  - Put more in queries and less in foreign keys (Django reverse lookup)

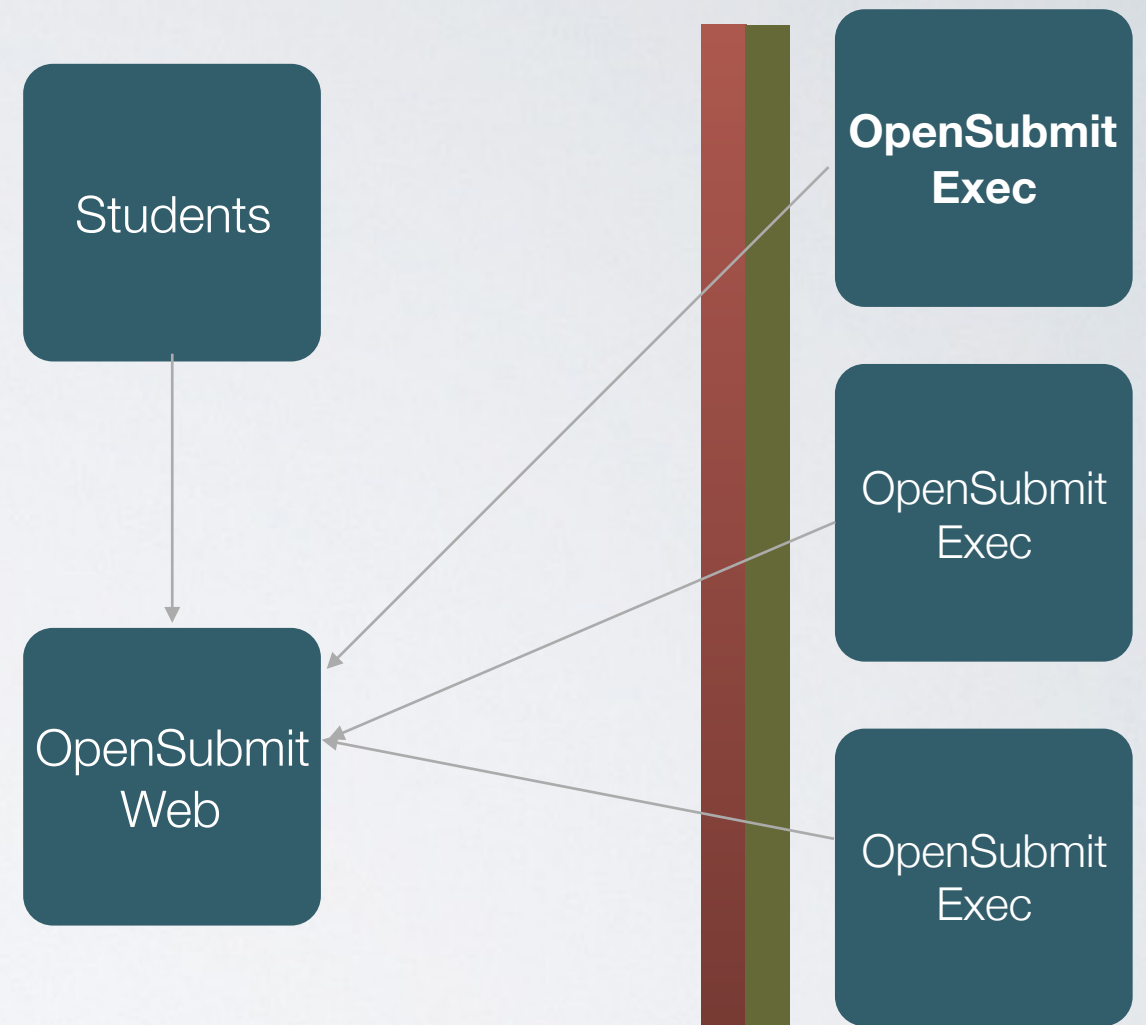


# EXPERIENCES: WEB AUTHENTICATION

- I hate separate accounts.
  - There is only social login in this project.
  - All universities anyway have their own single sign-on.
- **python-social** has pluggable backends and seamless Django integration
- Lessons learned
  - If you change your authentication code base, hell breaks lose.
  - Identity mapping based on e-mail no longer helps.

# OPENSUBMIT-EXEC

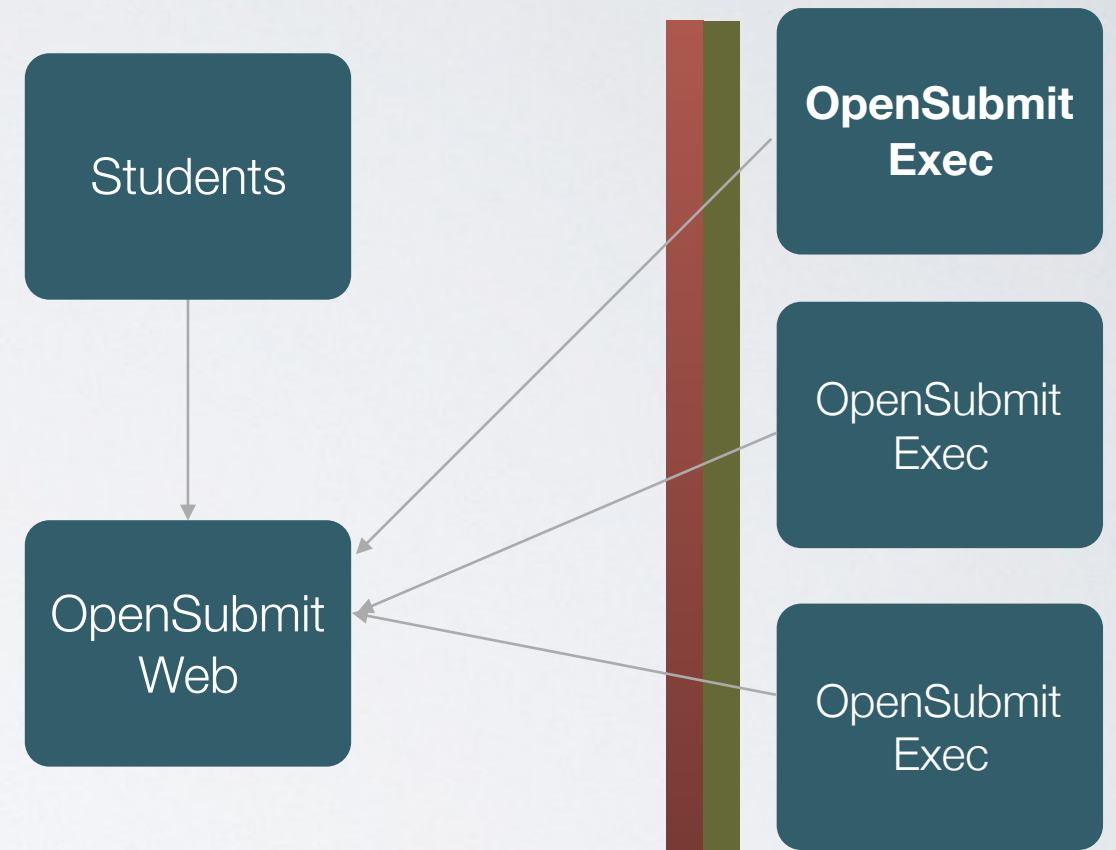
- Python 3 script on test machines
  - Download of validator and student code from predefined web host
  - Called by cron, or manually for testing
  - Isolation of student code through dedicated (virtual) machines



- Timeout for deadlocking student code, always report a result
- Handling of archive obscurities on the test machine, not on web server

# EXPERIENCES: PUSH VS. PULL

- Executors ask for jobs via HTTP
  - Pulling high frequency cron job
  - Push-receiving daemon would reduce latency, but increases complexity
  - Trivial to operate, good enough for the scale of this application
- No inbound connectivity needed, outbound connectivity restricted
- Load peaks shortly before assignment deadline, handled by dynamic VM creation





# REGRETTABLE THINGS

- Some early design mistakes hunt you forever
  - No central state transition (!) logic
  - View-driven development, instead of API focus
  - Half-baked configuration management
  - Missing consideration of installation maintenance
- Missed chance for on-the-fly manual writing
- Ignorance of the async job queue problem.



# THINGS DONE RIGHT

- Ignorance of performance issues (until they show up)
- Ignorance of PEP-8 (with 1,5 developers)
- Ignorance of test coverage for non-security stuff
- Minimalism policy
- Trivial UI for most users, complex UI for power users
- Making terrible code public.
- Python, Django, PostgreSQL. They just do the job.



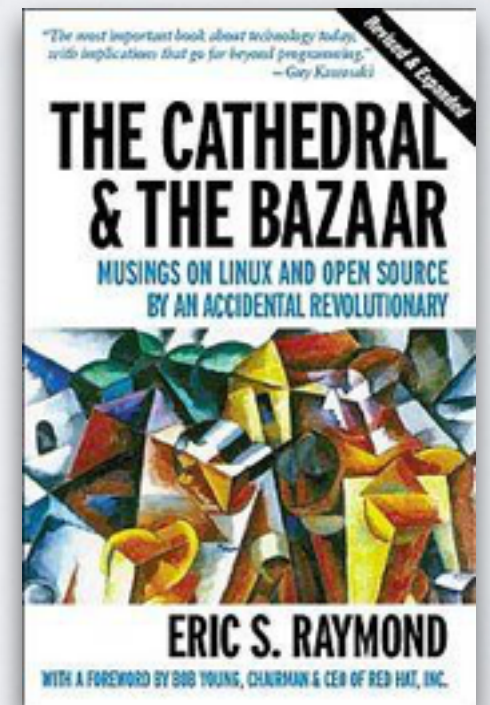
# DJANGO IS STILL AWESOME

- Fulfilled its promises from day one
  - Fantastic documentation
  - Powerful ORM query features
  - Django Admin is the core of the teacher backend
  - Stupidity protection layer (e.g. XSS, input sanitization)
  - Healthy feature addition / deprecation ratio
  - Trustworthy deprecation policy, no surprises on updates



# BIKE-SHEDDING

- „... disproportionate weight to trivial issues“ [Wikipedia]
- In combination with users that see you as vendor, this is annoying
  - Explain your prioritization policy
  - Be the representative for all silent (student) users
  - Remain grateful that they contribute ,something‘
  - There are books about that.
- It's open source. Let them fork and see what happens.



# FUTURE STEPS

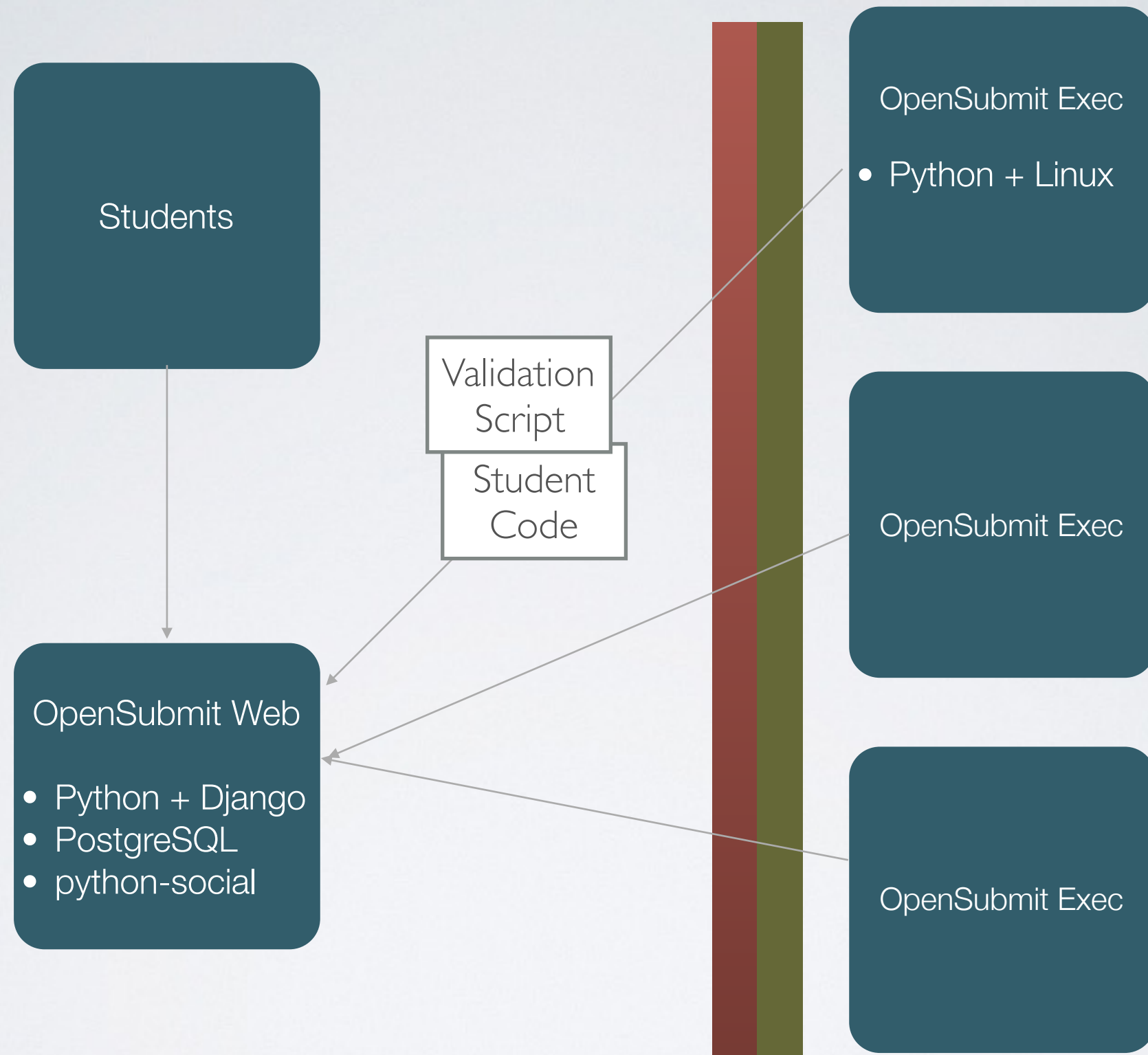
- More test coverage (for the GitHub badge)
- More documentation (for the users)
- Test machines with Vagrant / Docker / libvirt
  - „Hyper-agile cloud-scale load management“
- UI homogenization with (Django) Grappelli
- LTI provider
- Improved grading with sophisticated in-browser preview

# CONCLUSION

- OpenSubmit brings the KISS principle to learning management
  - Focus on one problem, and do this right
- Developing your own tools for daily work is fun, but ...
  - Consider that you will create technical debt.
  - If you want people to use it, you need advertising.







<https://github.com/troeger/opensubmit>