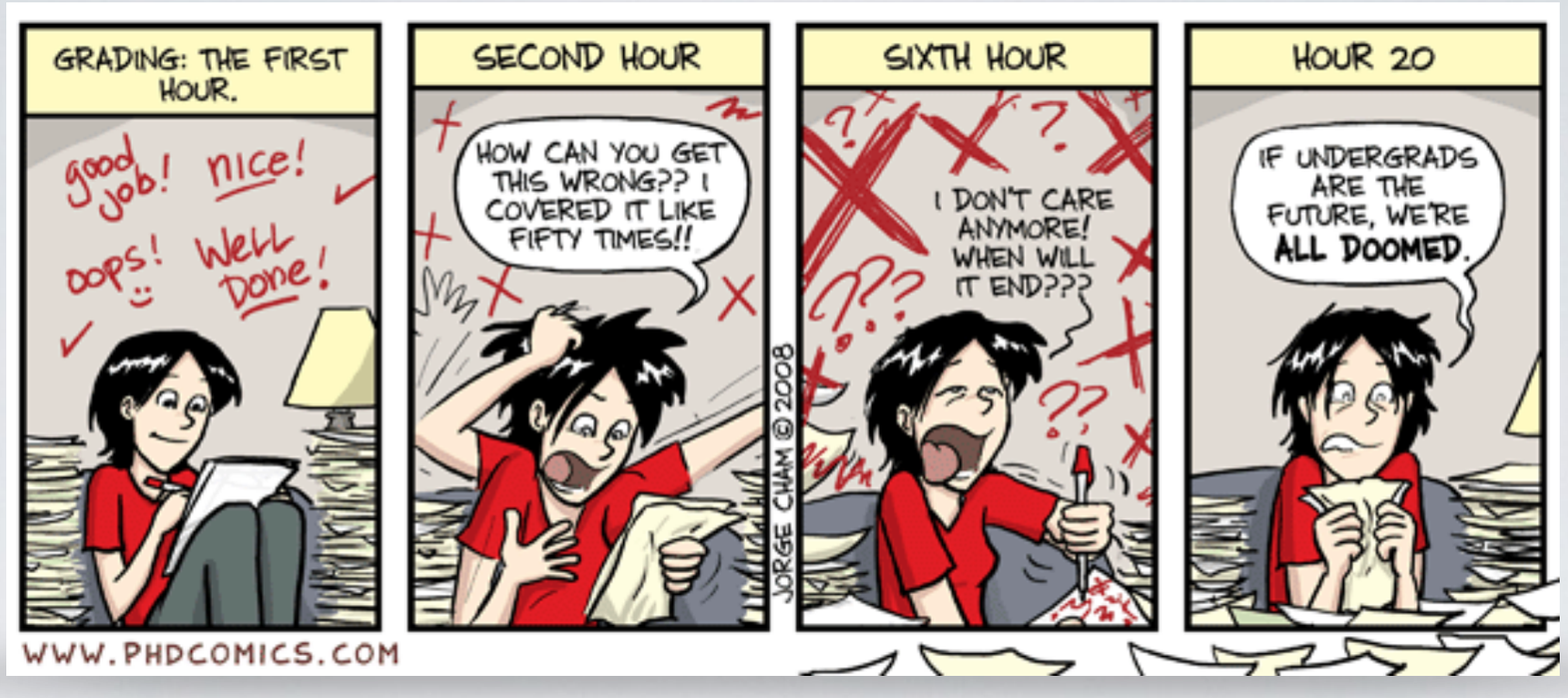


THE OPENSUBMIT PROJECT

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

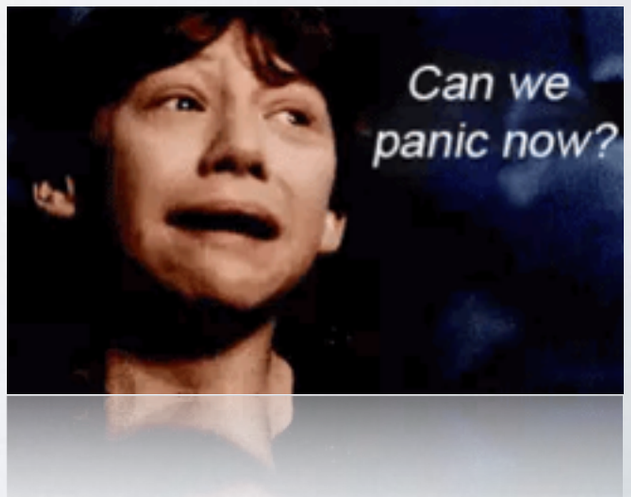
PART II:TECHNOLOGY

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



Electronic Submission

- HW5 Solutions

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

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

1. Log onto an owlnet 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 na
   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
in, "hw1" in the above example.

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

Also note that projects with more than one submission ("milestones") will have uniqu
submission. 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++` compiler, and it is your responsibility to ensure that it will run on that computer using `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++`).

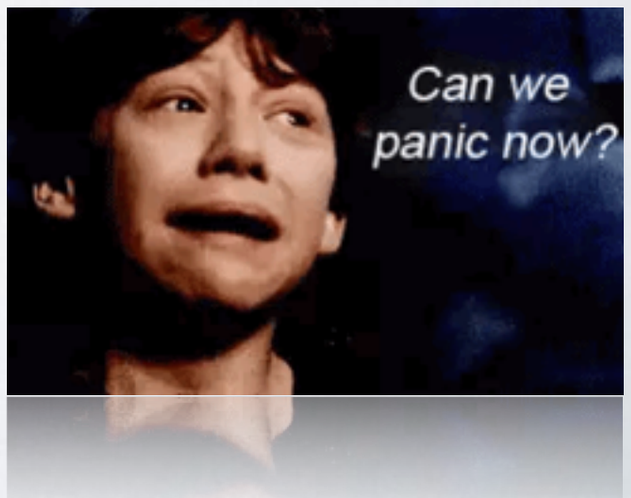
PROBLEM 1: FORMATS

- 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?

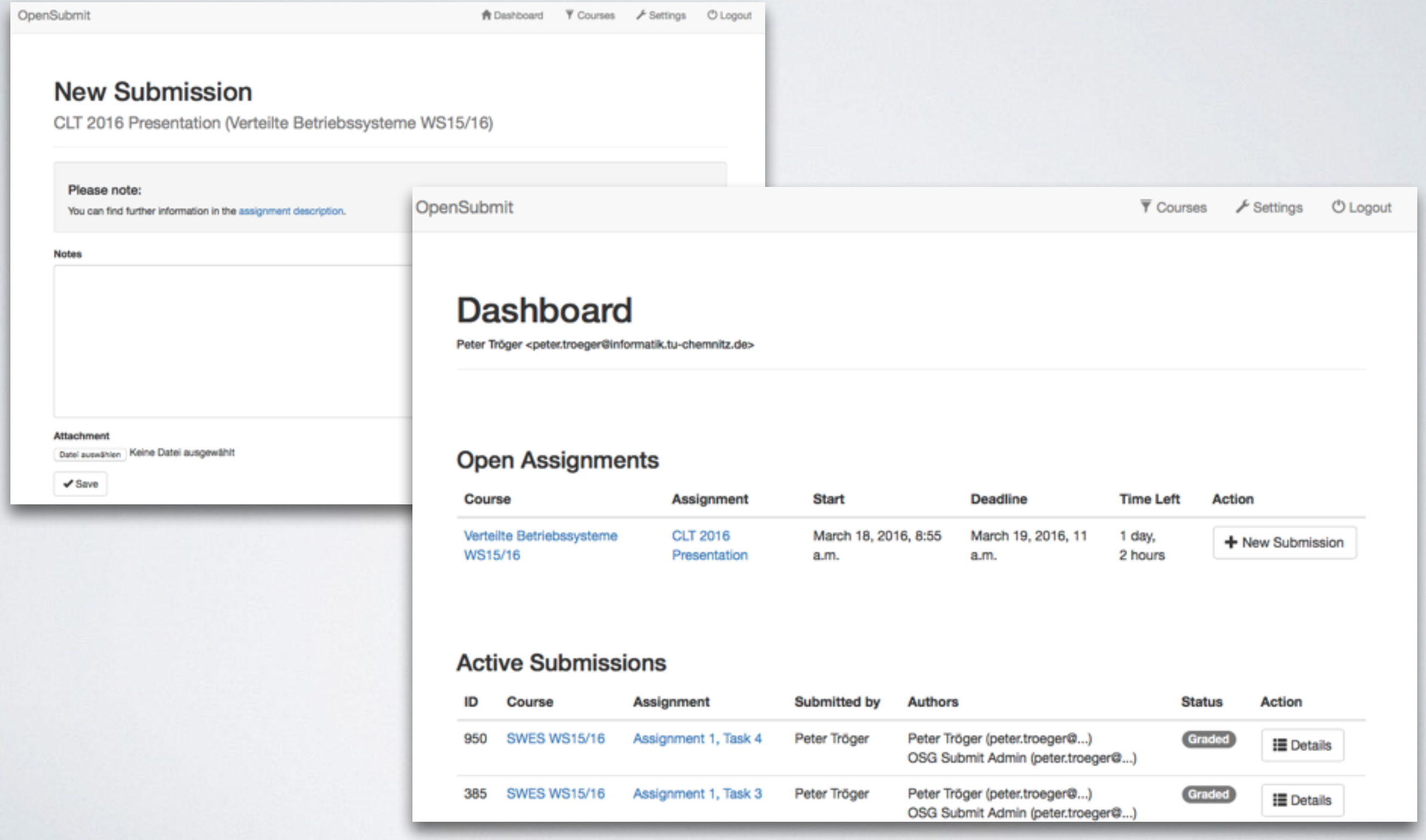
- Moodle (and most other LMSs) are all-inclusive
  - Online teaching, participant management, grading books, collaboration, file management, calendar, ….
  - Course owners need training. Seriously?
  - Assignments are just a small part.
  - 

    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

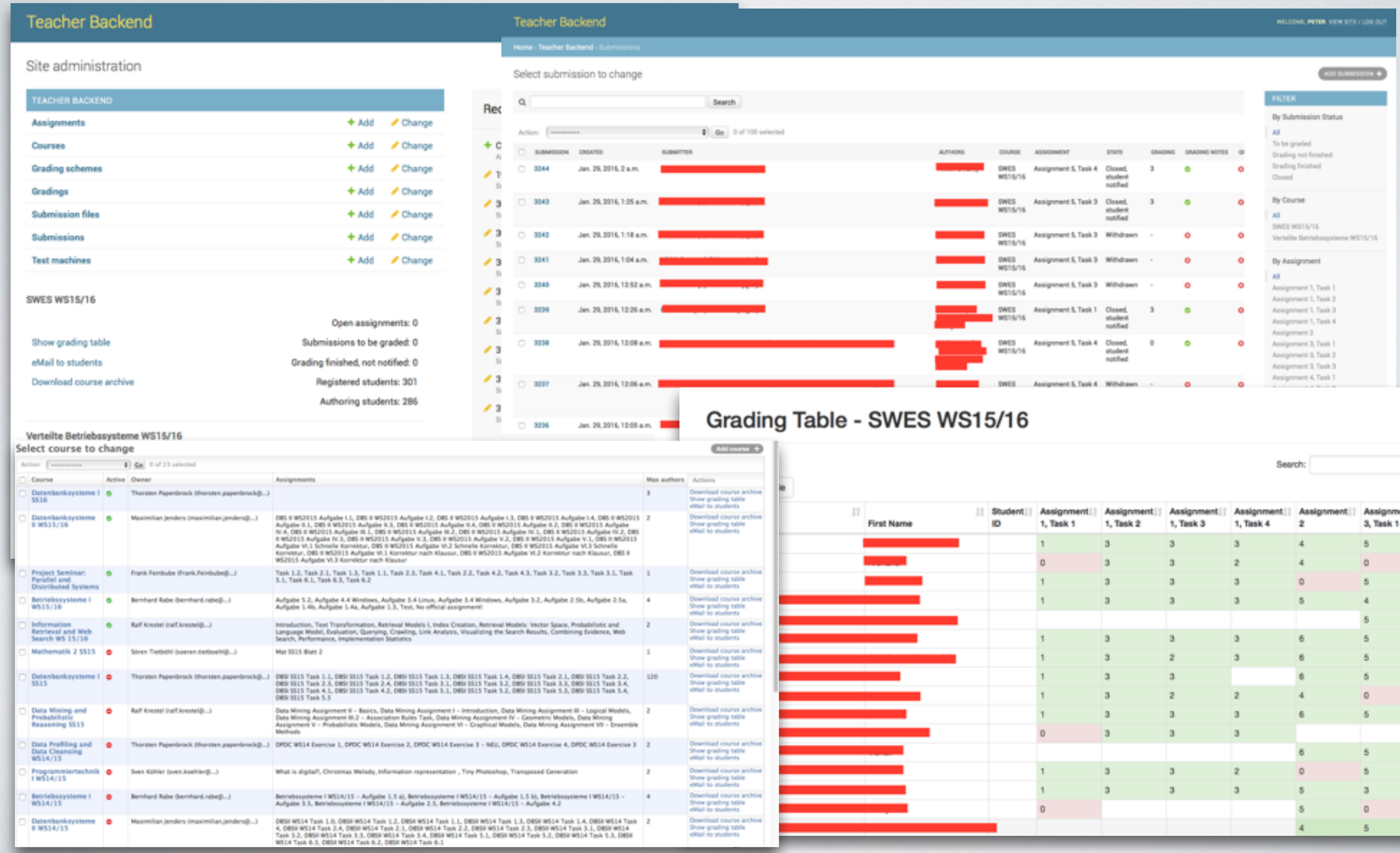

OpenSubmit
Dashboard
Courses
Settings
Logout
New Submission
CLT 2016 Presentation (Verteilte Betriebssysteme WS15/16)
Please note:
You can find further information in the assignment description.
Notes
Attachment
Datei auswählen
Keine Datei ausgewählt
Save
OpenSubmit
Courses
Settings
Logout
Dashboard
Peter Tröger <peter.troeger@informatik.tu-chemnitz.de>
Open Assignments
Course
Assignment
Start
Deadline
Time Left
Action
Verteilte Betriebssysteme WS15/16
CLT 2016 Presentation
March 18, 2016, 8:55 a.m.
March 19, 2016, 11 a.m.
1 day, 2 hours
New Submission
Active Submissions
ID
Course
Assignment
Submitted by
Authors
Status
Action
950
SWES WS15/16
Assignment 1, Task 4
Peter Tröger
Peter Tröger (peter.troeger@...)
OSG Submit Admin (peter.troeger@...)
Graded
Details
385
SWES WS15/16
Assignment 1, Task 3
Peter Tröger
Peter Tröger (peter.troeger@...)
OSG Submit Admin (peter.troeger@...)
Graded
Details

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

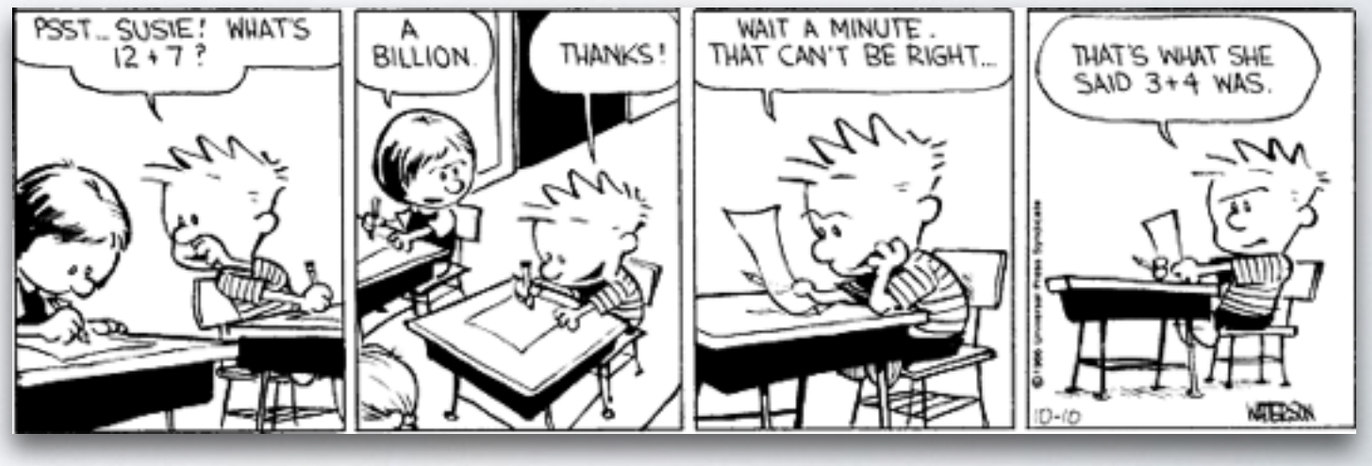
winter term | summer term | winter term | summer term | winter term | winter term

TEACHER UI


Teacher Backend
Site administration
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
Show grading table
Submissions to be graded: 0
eMail to students
Grading finished, not notified: 0
Download course archive
Registered students: 301
Authoring students: 286
Verteilte Betriebssysteme WS15/16
Teacher Backend
Home › Teacher Backend › Submissions
Select submission to change
Search
FILTER
By Submission Status
By Course
By Assignment
Select course to change
Grading Table - SWES WS15/16
Search:
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 1

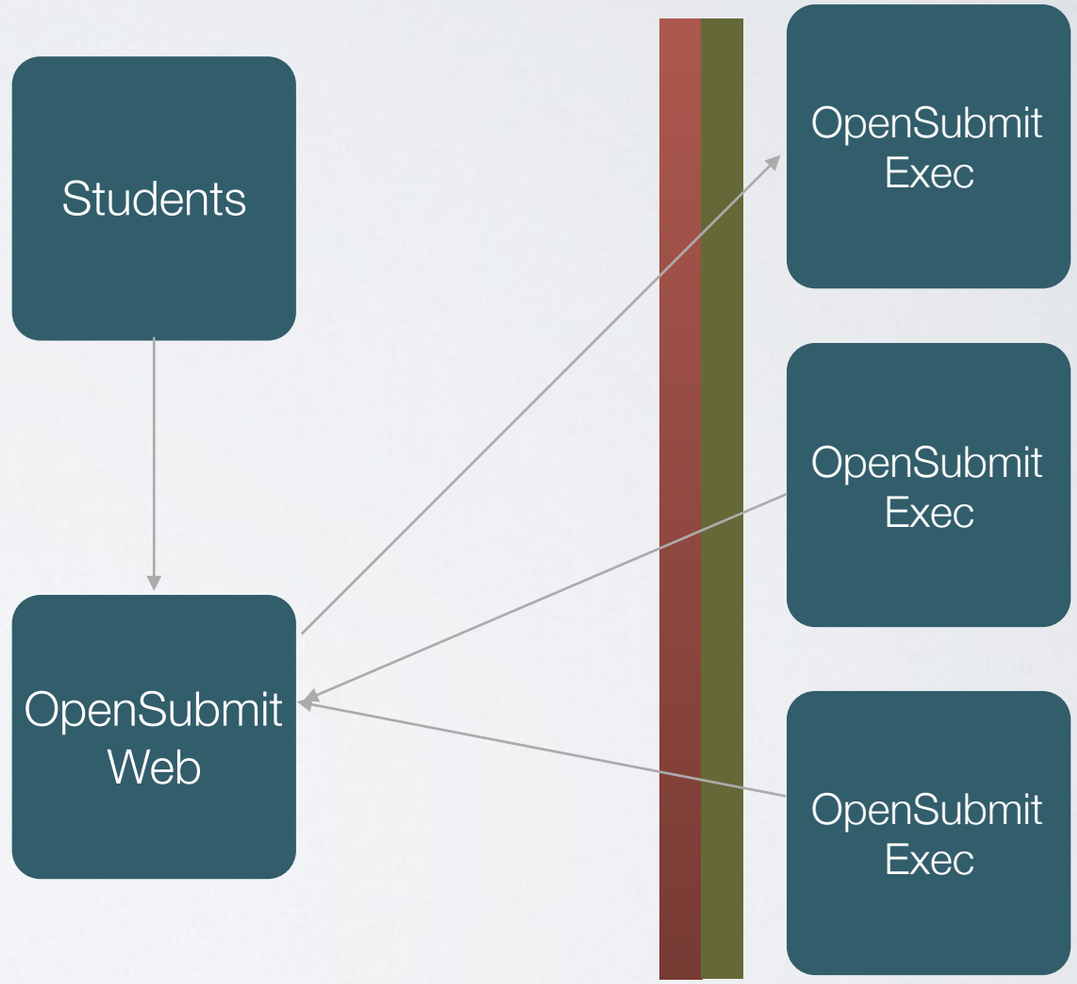
EXAMPLE: DUPLICATE DETECTION

- 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



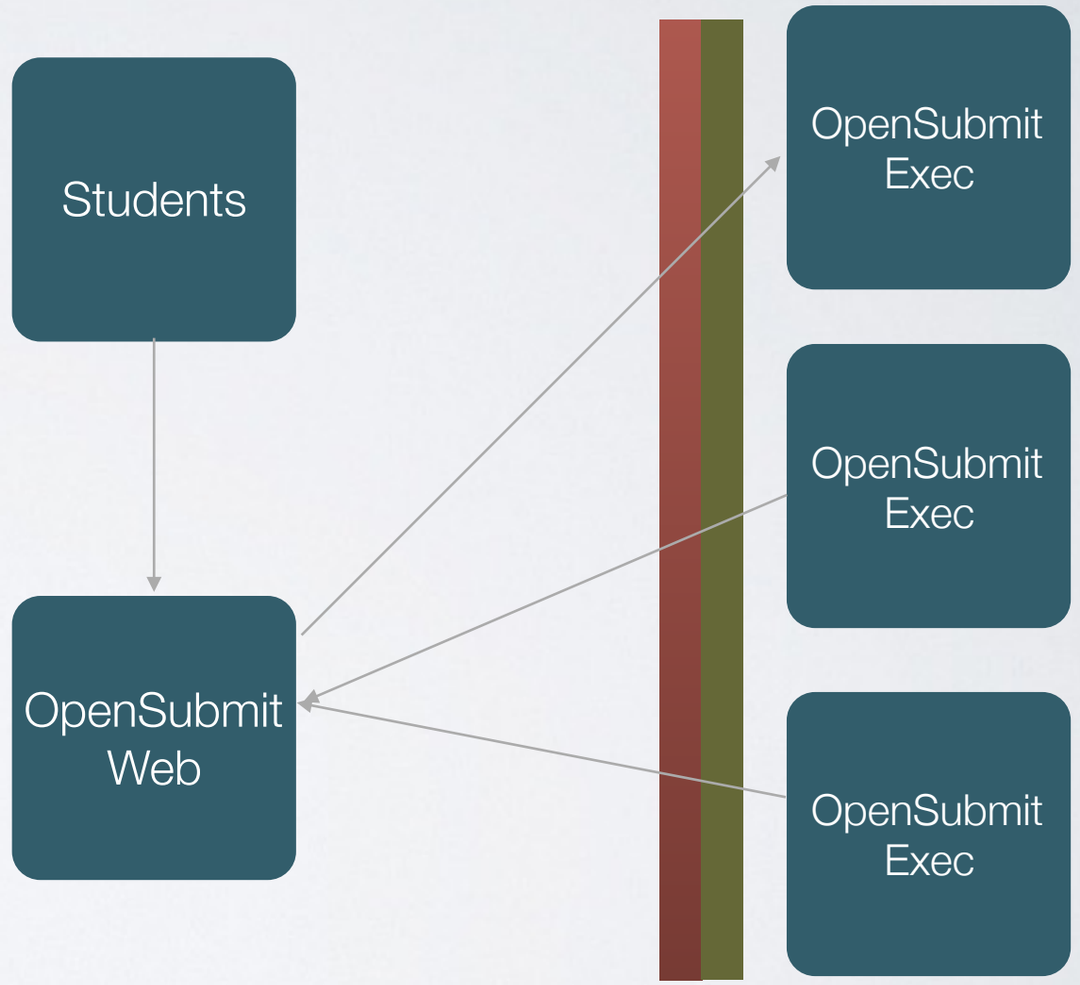
OPENSUBMIT PRINCIPLES

- **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	SWES WS15/16
Assignment	Assignment 1, Task 4
Submitter	Peter Tröger (peter.troeger@...)
Authors	Peter Tröger (peter.troeger@...), OSG Submit Admin (peter.troeger@...)
Notes by submitter	

File	submission_RRZ5x3w.zip
	Compilation test result:
	`gcc -o bitcount bitcount.c`
	Validity test result:
	`Congratulations! Your code seems to work.`

State	Graded
Grading	3
Grading Notes	

TEACHER UI

Submission and test results

Notes:

Stored upload: bitreverse_CCpjYQH.zip (Preview) New upload: 2015-10-30/bitreverse_CCpjYQH.zip

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: 3

Archive Preview

bitreverse/bitreverse.c

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

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

bitreverse/makefile

ASSIGNMENT

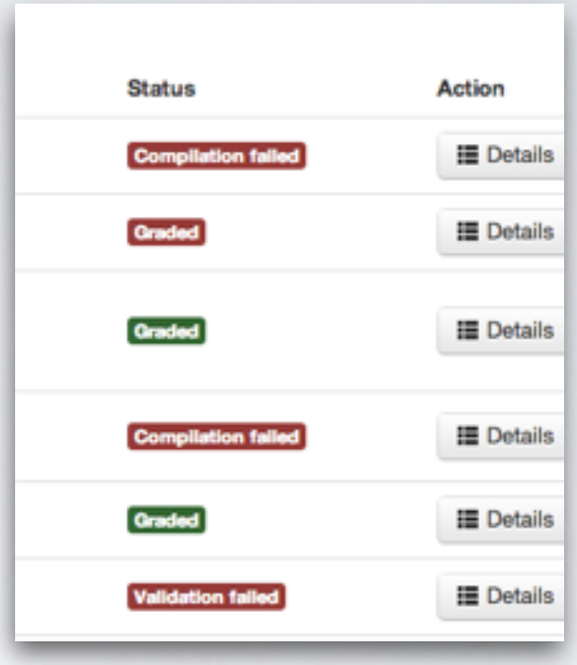
- Relates to a course
- Has a grading scheme, start time, hard deadline and description link
- May have a **soft deadline**
- May include **compilation** of file upload
- May have a **validation script** executing the student code
- May have a **full test script** executing the student code
- Has **grading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



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 alway visible

AUTO GRADING?

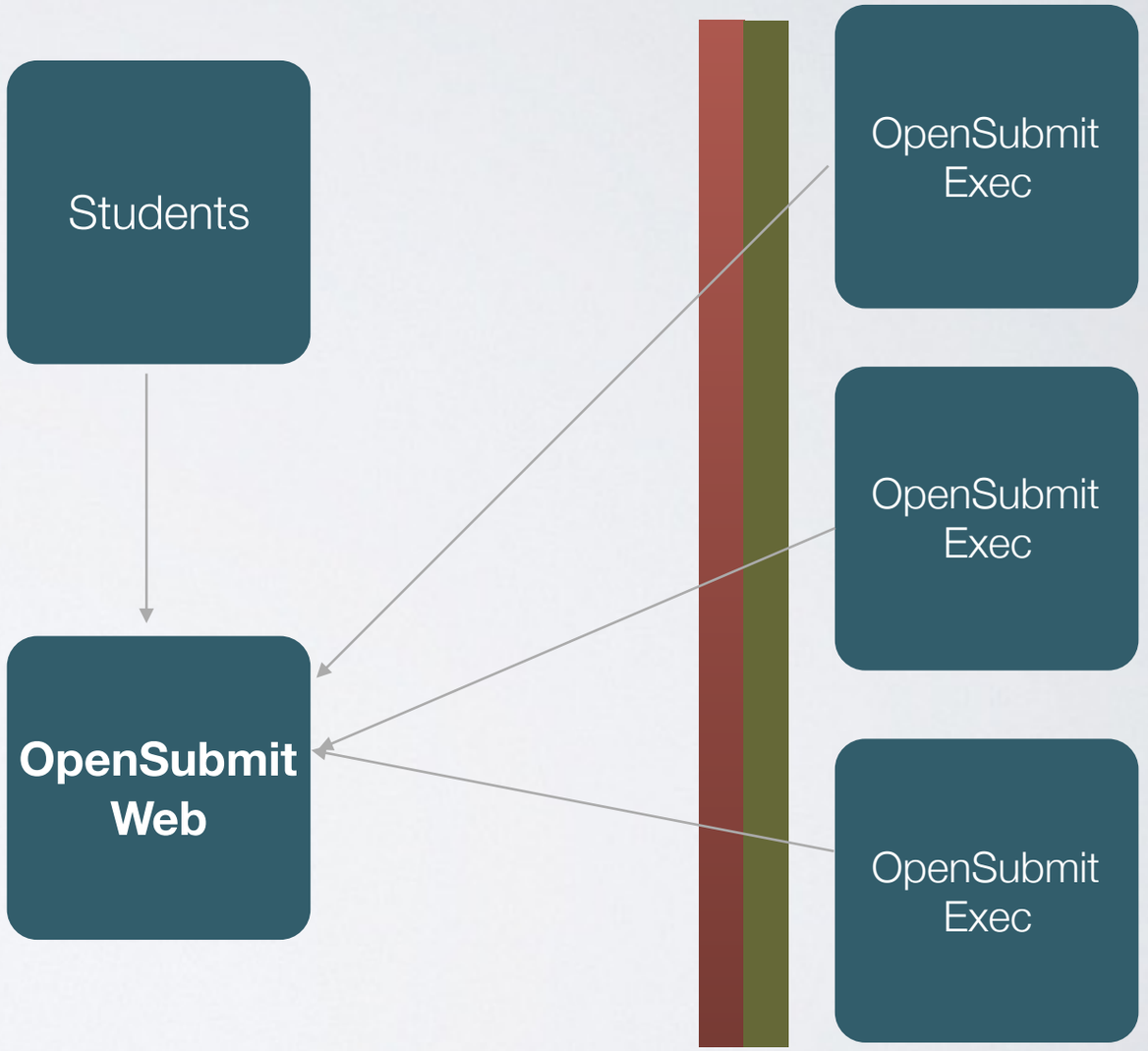
- University teachers would love that, especially with MOOCs
  - Option 1: Students develop (blindly) against a test suite
  - Option 2: Analyze code AST to derive some score
  - Option 3: Check for OS-visible behaviors on execution
  - Option 4: ???
- Very hard to generalize. Also a legal problem.
- 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)
  - Avoid solutions with „advanced magic“

EXPERIENCES: DATA MIGRATION

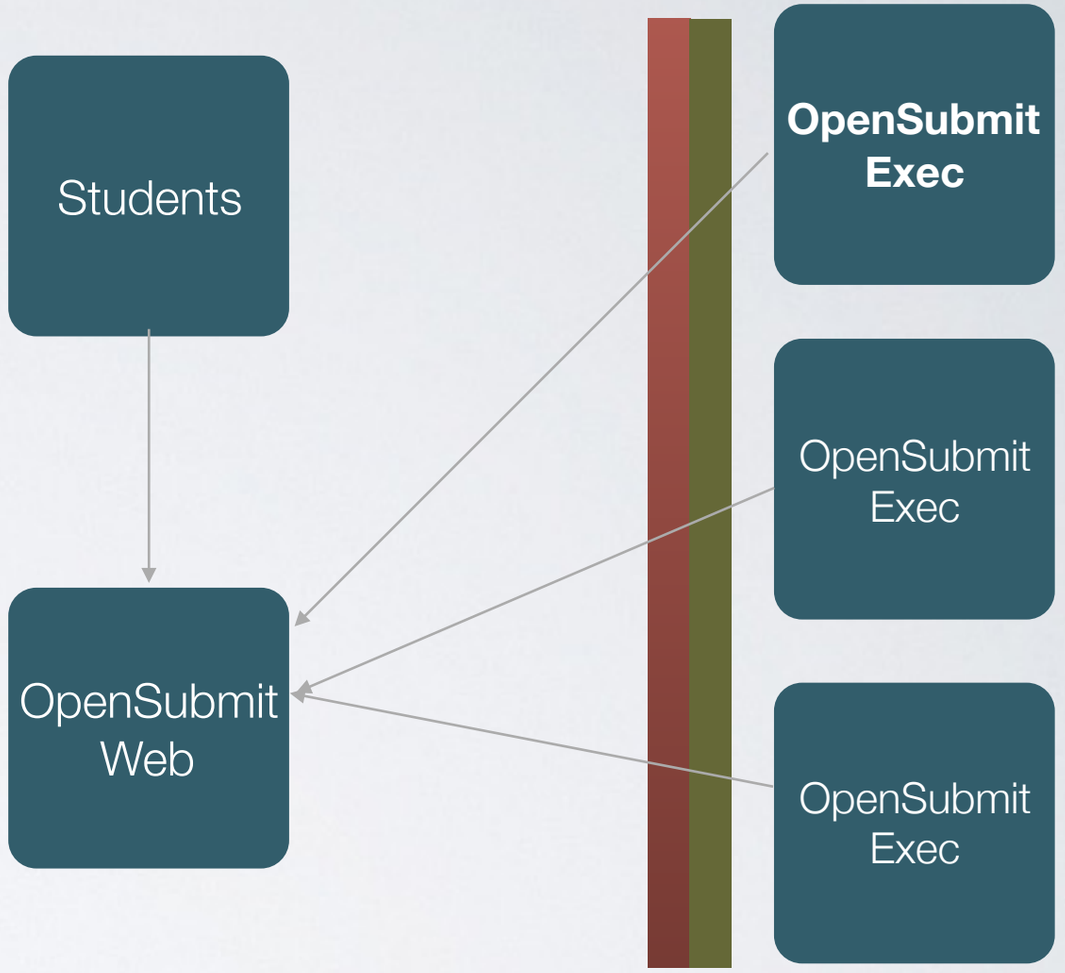
- Database migration needed on model changes
  - South was ok for that, Django>=1.7 support is even better
  - They handle model migration, but **data migration is your problem**
- Lessons learned
  - Create migration-friendly data models. You have only one try.
  - Data attributes may later become 1:N 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

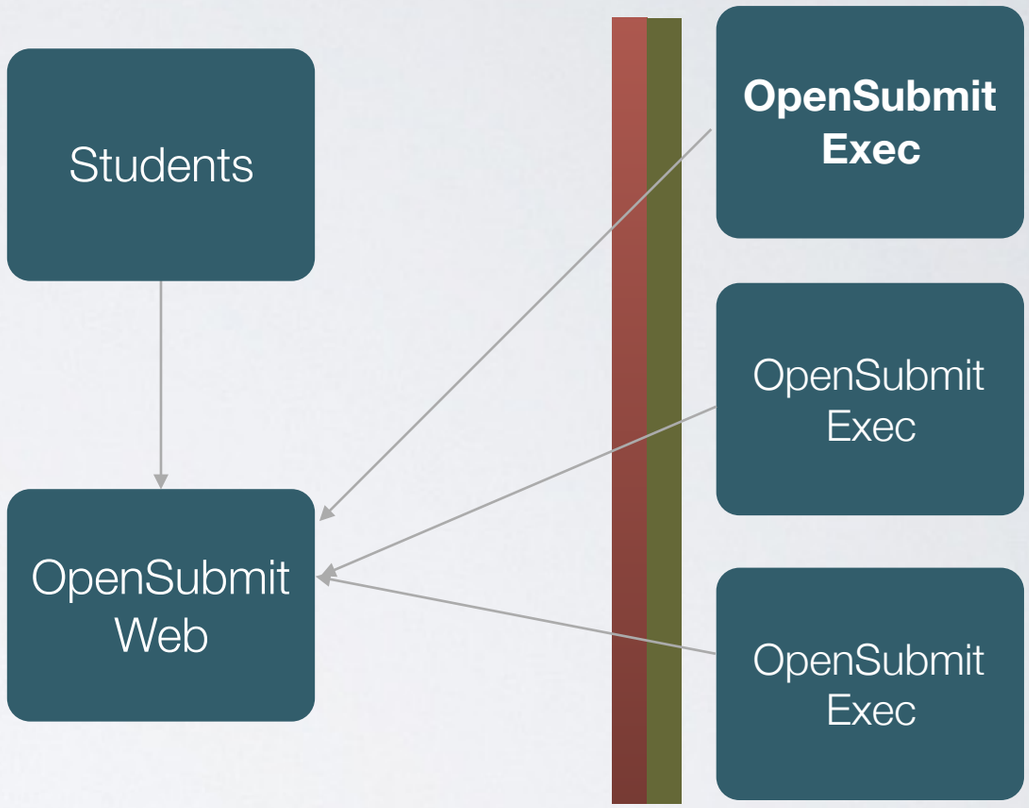
- Python 3 script on test machines
  - Download of validator and student code from predefined web host
  - Called by cron, or manually for testing
  - Isolation of student code through dedicated (virtual) machines



- 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

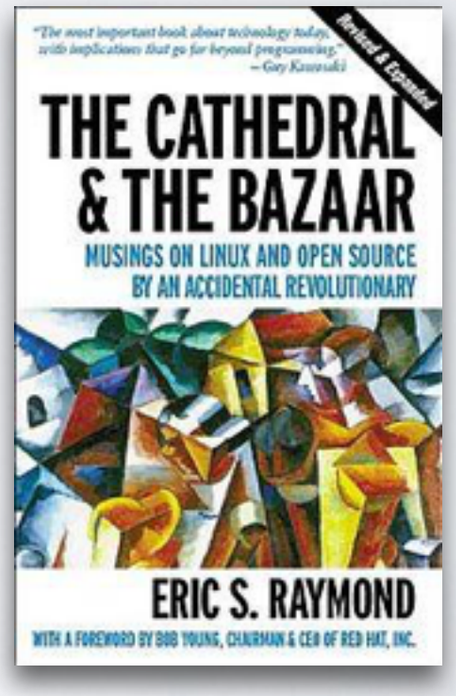
- Minimalism policy
- Trivial UI for most users, complex UI for power users
- Making terrible code public.
- Python, Django, PostgreSQL. They just do the job.

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.



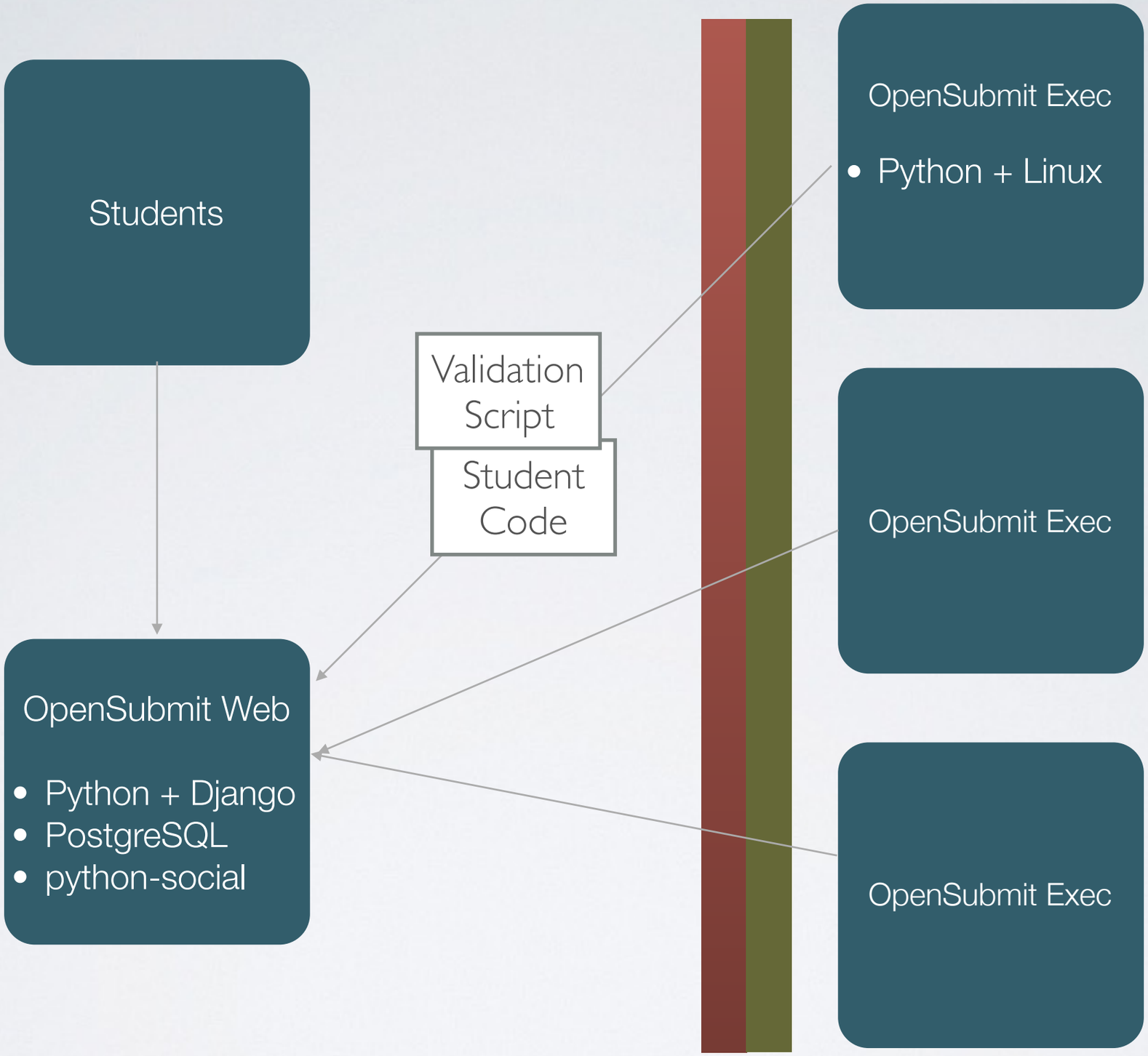
FUTURE STEPS

- 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