EvaP

Jennifer Stamm, Stefan Neubert

Winter Term 2015/16

Inhaltsverzeichnis

1 Introduction

1.1 EvaP – An Evaluation System

The online platform EvaP is used for evaluation of courses at the Hasso Plattner Institue (HPI). Its development started in 2011 when the student representatives decided to redevelop the former system EvaJ. Now, it is an Open Source project hosted on GitHub, even though the main developers are still part of the student representative team.

EvaP's time to shine is at the end of each semester. Each university course is evaluated with specific questionaires chosen by the lecturer. Students are allowed to give anonymous feedback to different aspects of the lecture, the lecturer itself and additional tutors through a grading system and comments. EvaP encourages evaluation by rewarding participation with points. These reward points can be redeemed for currency to use at HPI events. EvaP's latest feature is the distribution of course grades through the platform.

1.2 Motivation – Why Should EvaP Be Analyzed, Verified and Tested?

EvaP is an important source for feedback at HPI. The platform offers students a way to express their critique anonymously. It documents the feedback for lecturers. Thus, they do not need to collect and save the feedback themselves. Additionally, they can take their time to evaluate the student's feedback. Furthermore, the evalutation of all courses is saved centrally. This allows to gain an overview over the quality of HPI courses and compare feedback over time as well as with other courses. Therefore, EvaP is an important tool at the HPI.

Since EvaP is developed by students, the responsible persons and main developers change regularly as older students graduate and new ones enroll. The change of responsible persons shifts the view of which features, programming paradigms and quality assurance are most important. Consequently, the requirements change. The change of main developers means a loss of knowledge about the existing code. Besides, the Open Source aspect allows several developers without inside knowledge to add code. Even though all code is reviewed and checked by the main developers, they may not grasp it as well as self-written code. Events like Hackdays or Hacking Hours are used to promote EvaP's development. While these practices ensure the advancement of EvaP, they may endanger its quality.

Because of EvaP's importance at the HPI, its quality should be ensured. Therefore, we will analyze, verify and test the software within the scope of this lecture.

2 Milestone 1

As suggested the duration of milestone 1 is from the beginning of the project until the end of december. Milestone 1 includes the set up of the software project which led to

the discovery of the first bug. It includes the first steps taken to gather information, get comfortable with the project and planning of the project. Lastly, it includes the phase of testing the project regarding graph coverage.

2.1 Set Up

- set up of vagrant on windows does not work for submodules - unnoticed by developers because of different os

2.2 First Steps

Incidentally, this is the first semester for the student representatives to host *EvaP Hacking Hours* biweekly. This is an event to give students a space to develop and work on EvaP. We will use it as a possibility to stay in contact with the main developers. As testers of EvaP it is a valuable resource to be able to talk directly with developers. This way we can gather current information easily. We are able to verify the content of old artifacts with them as well as our future findings.

2.2.1 Application Survey

The current main developer of new features is Johannes Wolf. He is supported by Johannes Linke who is mainly responsible for a good coding style, including code review and refactoring. We interview them about information regarding the first steps of our project.

Development Paradigm and Development Languages There are no development paradigms explicitly determined. But as stated there is a main developer responsible for code review. As it is all newly developed code is reviewed before it is accepted. Everyone is able to review code and discuss it with the coder and other reviewers. This practice shall ensure readable code and distribute knowledge about code changes. Additionally, it is implicitely assumed that paradigms of the used development languages are followed. The development languages are:

- Python 3 through the Django framework
- HTML
- Javascript
- CSS

As an example for paradigms given by the languages we checked the document known as *PEP 0008*. This is the style guide for Python code written by Guido van Rossum, the author of Python, and followed by most open source projects in Python. Link: `https://www.python.org/dev/peps/pep-0008/`

Requirements / Specification / Documentation / Artefacts Requirements were elaborated at the start of EvaP's predecessor EvaJ several years ago. No original artifacts were stored even though most requirements still hold. Examples are:

- Evaluation should be anonymously
- Written feedback is only readable to a small, strongly specified selection of people
- Written feedback is reviewed before it is available to the lecturer

As expected of an open source project managed by students there is no formal specification of EvaP. Though there are different information gathered in a wiki hosted on Github. (Link: `https://github.com/fsr-itse/EvaP/wiki`) The responsibility to specify new features is on the main developer, Johannes Wolf. He collaborates directly with the users of the feature. Another feature of Github is used to track bugs: the label *[T] Bug* for *issues*.

Current testing status / Bug repositories GitHub, TravisCI, gemnasium, landscape

Personal involvement User, tester, developer

2.3 Initial Test Plan

2.3.1 Five V&V Questions

1. When do verification and validation start?
   When are they complete?
2. What particular techniques should be applied during development?
3. How can we assess the readiness of a product?
4. How can we control the quality of successive releases?
5. How can the development process itself be improved?

2.3.2 Test Classifications and Approaches

Validation vs. Defect Testing

Development, Release, User Testing

Unit/Component, Integration, System Testing

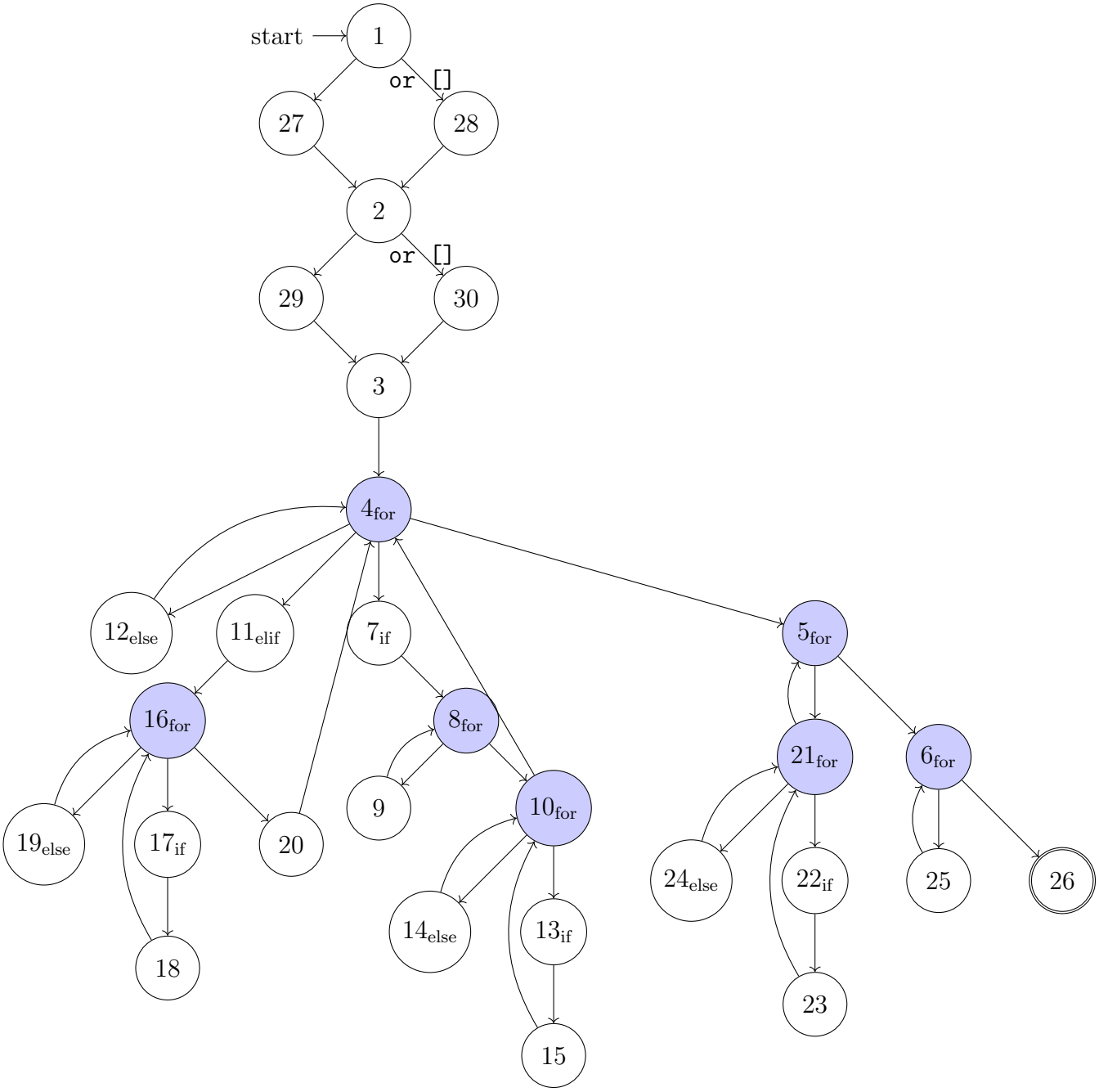

Abbildung 1: Control Flow Graph of the Function `send_publish_notifications` in `evap/evaluation/tools.py`