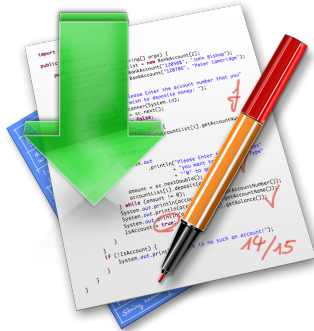


Using the Grit Submission System

Official User Manual for Version 1.2



November 21, 2015

Contents

1	What is Grit?	3
1.1	What does Grit do?	3
1.2	When do you need Grit?	4
1.3	What are the alternatives?	4
2	Getting Help	6
2.1	Online Documentation	6
2.2	Contact us	6
3	Getting Grit	7
3.1	Required Software	7
3.2	Using installation script	8
3.3	Direct installation (by hand)	8
3.4	Installation via Vagrant	9
3.5	Prerequisites	10
4	Running Grit	11
4.1	How to run Grit	11
4.2	How to configure Grit	12
4.3	First Startup	14
4.4	Setting up a submission structure	19
4.5	Further Information about Email and SVN	20
5	Frequently asked Questions	21
6	Credits	22
6.1	Developers Team GRIT	22
6.2	Developers Team VARCID	22
6.3	Special Thanks	23

1 | What is Grit?

Grit is a system, developed by TEAM GRIT and improved by TEAM VARCID, that makes the life of lecturers and tutors of mainly programming courses easier and more comfortable. With Grit, we implemented a server-side one-button solution for fetching and processing student submissions. This is done in such a way that the tutors will not have to worry about not-compiling programs and can therefore concentrate on the actually written code and its quality. Grit helps to maximize the teaching quality by ruling out unnecessary work and lets the tutors focus on the really important matter: Teaching.

1.1 What does Grit do?

With Grit, the students do not directly submit their solutions to our application. To be precise, they will not even notice that you are using Grit when submitting. Rather, Grit fetches the student submissions from whatever system provided for that purpose. As for now, Grit supports Subversion, and Email submissions, however, because of its high modularity, you can easily add your own *fetcher*. At the end of a previously set deadline, Grit fetches the submissions and processes them. As of now, we support Java, C and Haskell submissions, but like *fetchers*, additional modules can be added. After the processing, which includes compiling and testing the submission, using previously set unit tests, we will put that acquired information about the submission in a report, ready for you to download. For now that report will be either a PDF or a plain-text file, but, as you will have guessed, other formats can be implemented easily.

What Grit does not.

Grit does not provide submission features, as it can only import submissions. Also, it is not possible to manipulate code in Grit, because this is not what it was intended for: offline correction on paper. Grit does not replace a lecturer or tutor by adopting all of their correction procedure, however Grit simplifies their work in supporting them with automatable checking routines. So, Grit does not offer any methods to control semantical errors in the program code like checking if the summation really does add the two summands. Furthermore Grit is not suitable to test if the submitted solutions of the students actually solves the given problem or a part of it. Grit cannot check any additional exercise restrictions specified in the assignment paper (e.g. using inherited methods for solving the exercise).

Grit will also not correct any programming faults automatically, but will present you the incorrect passage, the error, the passed tests as well as compiler error message.

1.2 When do you need Grit?

You are a lecturer/tutor of a general programming course, a java workshop, a C++ seminar, ..., in which the participants can or have to submit solutions to a given problem. If you are tired of searching for syntax errors, compiling errors, typos, etc. for hours at a time, then you will definitely appreciate our product to increase the efficiency of your correction by prechecking every submitted solution. All in all, if you are a university or college professor or lecturer, teacher, tutor, or corrector, Grit can help you automate collecting electronically submitted assignments concerning programming code from various resources. Compiling tests and predefined tests can be run so you can easily focus on the important things.

When you do not need Grit.

You will not need Grit if you do not provide electronic submission of assignments for your course. Also, Grit is only suitable for fetching programming code assignments, however, other types of submissions can be added by oneself. Grit does not offer any features for students, as it only provides functionalities used by teachers and correctors.

1.3 What are the alternatives?

TEAM VARCID has compared Grit with the following alternatives. You can find it in another document.

- The Marmoset Project
(marmoset.cs.umd.edu)
Marmoset is a system for handling student programming project submission, testing and code review. It has been developed at the University of Maryland for over 5 years. It works in all different programming languages, and is designed to work well with both very small and very large projects, such as our OS course in which a submission consists of tens of thousands of lines of code.
- BOSS Online Submission System (Beta phase)
(sourceforge.net/projects/cobalt)
BOSS is a course management tool that allows students to submit assignments online in a secure manner. Staff can mark work and run automatic tests on submissions.
- Cafe grader
(gitorious.org/cafe-grader)
Cafe grader is a submission and grading system for programming contest and training. This software was used in APIO'08. It currently has mainly two components: the web submission system and the grader. The web app uses Ruby on Rails; while

the grader was written in plain Ruby. Current activity: process of migrating to git, and developing installation scripts.

- Praktomat

(<https://github.com/KITPraktomatTeam/Praktomat/>)

Praktomat is a quality control system for programming exercises. Students hand in their submissions directly into Praktomat which then compiles and tests them. It was first developed at the University of Passau in 1998 and was later migrated to the Karlsruhe Institute of Technology in 2008. The software is written in Python and uses the Django web framework. It is able to manage Java, other programming languages can be added.

2 | Getting Help

There are various resources available to get the most out of Grit.

2.1 Online Documentation

The online version of the documentation is essentially the same as the one you are reading now, however, you will have access to additional resources concerning troubleshooting. But online you will find the newest version.

2.2 Contact us

Do you have suggestions? Do you miss a feature? Contact TEAM GRIT or TEAM VARCID by mail. You will find our mail addresses at the end of the document.

3 | Getting Grit

If you want to acquire Grit, you have to clone the repository from Github.com by using the following command:

```
git clone git@github.com:team-grit/grit
```

or using the script "install-GRIT.sh". The script can be found at

```
http://github.com/team-grit/grit
```

Note: Both options will clone the complete repository on your computer.

Now you have two options for installation: Either a direct installation on your system or using the virtual machine vagrant. Grit delivers an initial setup for vagrant. Currently direct installation is only supported on a Linux system. Though it is recommended to use vagrant if you want to install Grit on Mac or Windows PC. Both ways of installation are explained in section 3.2 and 3.4, if you want more control during installation you can install Grit by hand, see section 3.3 for more details. Before you start you should check the following section whether all required software can be installed on your system.

Note: These packages will be automatically installed if you use the install-GRIT.sh. described in section 3.2

3.1 Required Software

For installing Grit you need the current version of the following software.

- Java Development Kit (jdk7 or higher)
 - The package contains the javac compiler, which you need to compile the students' Java submissions
- TeXlive (texlive-full)
 - you need Texlive to create your reports
- GNU-C-Compiler (gcc)
 - to compile the students' C submissions

- Glasgow Haskell Compiler (ghc)
 - to compile the students' Haskell submissions
- Subversion (svn)
 - Grit can fetch submissions from subversion repos,
- Secure Copy (openssh server and client)
- G++ Compiler (g++)
- Git
 - you need Git to get Grit.
- Gradle
 - you need Gradle to compile the source code of Grit

3.2 Using installation script

Warning: You need a Linux system with the apt-get package to use the script for a direct installation!

Copy the script install-GRIT.sh into any folder you wish to install Grit. The required software will be installed by the script! Now you have to run the script.

First you have to make the script executable by the following command:

```
chmod u+x install-GRIT.sh
```

Now you can run the installation script:

```
./install-GRIT.sh
```

Grit should now be installed in a folder ./grit/build/install/GRIT relative to the directory where you run the install script.

3.3 Direct installation (by hand)

Warning: You need a Linux system to do a direct installation because Grit uses Unix shell commands.

In this section it is explained how Grit can be installed step by step using unix commands. However all this commands are used by the install-Grit.sh, though it is recommended to use this script explained in section 3.2.

The first step is to install all required files which are listed in 3.1. Now go into the directory where Grit should be installed. There you have to generate a git clone in to this location:

```
git clone git@github.com:team-grit/grit
```


Now there should be a folder called grit. You have to go inside the folder:

```
cd grit
```

In the next step you have to run gradlew:

```
./gradlew install
```

Grit should now be installed in a folder `./grit/build/install/GRIT` relative to the directory where you run gradlew install.

Note: `./gradlew install` compile the source code of Grit and install it by default into the directory `./build/install/GRIT` (compare application directory in 3.2 and 3.3)

3.4 Installation via Vagrant

We recommend the installation via vagrant only for developers, because it set up a well defined space for Grit. However there may be connection issues using a virtual machine, which should be considered using Grit via a virtual machine.

The following instructions and commands should work with any shell like prompt (e.g. `cmd.exe` for windows systems).

To install Grit via vagrant you have to install vagrant first. You can download vagrant here:

```
http://www.vagrantup.com
```

Follow the instruction for installation of vagrant. Now you have to go into the directory where Grit should be installed. Then download Grit via the following command:

```
git clone git@github.com:team-grit/grit
```

After that run gradlew to compile the source code

```
gradlew install
```

And start the virtual machine with this command

```
vagrant up
```

An ubuntu system will be installed on the virtual machine. Connect to the virtual machine via a ssh client you prefer or use the following command

```
vagrant ssh
```

On the virtual machine you go into the application directory with

```
cd GRIT
```

3.5 Prerequisites

3.5.1 SVN repository setup

If you plan to use an SVN repository a "mapping file" must be provided. This file maps the name of the student folders to the student email address. Create a file `students.txt` in the top level of the repository. This file should contain only blank lines or lines formatted like `StudentFolder = StudentMail`. Lines beginning with the `#`-sign will be ignored.

The file might look like this:

```
# this is a comment
stud01 = max.mustermann@test.de
stud02 = elsbeth.musterfrau@test.de
```

4 | Running Grit

Now after successfully setting up your system you can finally begin using Grit. Grit runs as a Java application that sets up a web page. All administration and most of the configuration will run over that website. We tried to make your experience as intuitive as possible, but still there might be some ambiguities. For that we put an overview of the whole user interface at the end of this chapter. But first we will guide you through the typical workflow we designed and explain the elements step by step in the order you will most likely encounter them.

4.1 How to run Grit

You probably have installed Grit in a folder structure similar to `./grit/build/install/GRIT`. There are several scripts which can be used with Grit. Those will be briefly explained.

Note: All of them are found in the root directory `./grit`. However during compilation `runscript.sh` and `shutdownGrit.sh` are copied into `./grit/build/install/GRIT/`.

- `runscript.sh`
 - Starts Grit and creates a file in the same directory which contains information about the process id of GRIT.
- `shutdownGrit.sh`
 - Stops Grit. It looks for a file containing the process id to gently shutdown the process running Grit.

Warning: If you forgot to stop Grit via this script and your PC has to restart, then the process id is very likely different as in the file created by `runscrip.sh`. By the following commands you can try to find the process id and terminate it by hand.

```
netstat -tulpen | grep 8080
kill -15 "process id"
```

- `update-Grit.sh`
 - Update the current Grit source code by fetching and comparing the code from

the master tree. Only if the master tree contains different files, Grit will be updated. The update makes a backup of the current working directory, config files and reports. The backup will always be done despite Grit being up to date. Though it can be used for backup purposes as well. If a path is given like:

```
./update-Grit.sh example\path\where\backup\is\stored
```

the backup folder with the current timestamp will be created at this location, otherwise it will be created in the same folder. The backup folder will not be deleted!

Warning: If Grit is updated the source code has to recompile. In the process all temp directories such as wdir/, res/web/pdf or config/ will be deleted. The update-Grit tries to ensure via the backup that Grit goes back into the state before the update by copying the data from the backup folder into the respective folders. However, further testing has to be done to ensure that preupdate and postupdate states are equal.

- install-GRIT.sh
- install Grit see section 3.2 for more information.
- gritallinone.sh
- this script execute all beformentioned script via the following commands. It can be executed in ./grit.

```
./gritallinone start
```

executes runscript.sh

```
./gritallinone stop
```

executes shutdownGrit.sh

```
./gritallinone update [Path to backup folder]
```

executes update-Grit.sh Path is optional

```
./gritallinone restart
```

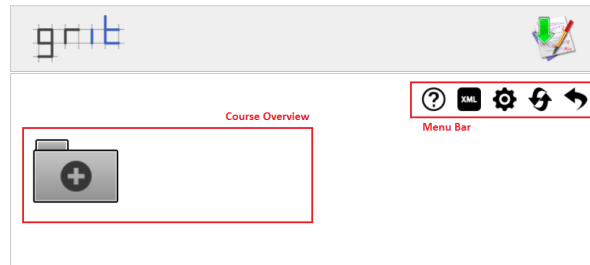
executes shutdownGrit.sh, then runscript.sh

4.2 How to configure Grit

Before you start Grit it is important that you provide the system with all necessary configuration information. Essentially there are two ways you can access the configuration file. The first way is to create a valid fully configured config.xml before booting the system so it can load it while booting. The default config, being created on first startup, can be found

further below.

The second way is to configure the system with the edit xml function on the webinterface. You find the Edit XML button in the menu bar.



The config file contains important data for the server, the mail notification and the admin account. If there is no config present at boot the standard failsafe config will be loaded. This config looks like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config>
  <server>
    <port value="8080"/>
    <loglevel value="INFO"/>
  </server>
  <email>
    <auth host="" port="" senderAddress="" password="" />
  </email>
  <admin>
    <user name="username" email="" password="password"/>
  </admin>
</config>
```

The structure consists of the following items:

<config>: Wrapper tag which specifies when the config starts and when it ends.

<server>: Holds information about the server.

<port>: The port the server is running on.

<loglevel>: The level of information which is saved in the log file. The following levels exist Severe, Warning, Info, Finer, Finest, thereby severe provide the least information while Finest provide the most.

<email>: Holds info about the email account where the notification mails are send from.

<auth>: All relevant authentication info for the mail account. The password, the mail address and the SMTP host server(e.g. smtp.gmail.com).

<admin>: Holds info about the admin account.

<user>: The data with which you want to log into the website and the email the admin want to get notifications on.

Information: Grit tracks all kind of information during running in the background in a logger file, depending of the log level. The file can be found in the following directory `./grit/build/install/log/`.

After the configuration was changed the system needs to be rebooted. This will be done automatically in the edit xml function. You should especially consider this if you changed the config file without the edit xml function on the webinterface.

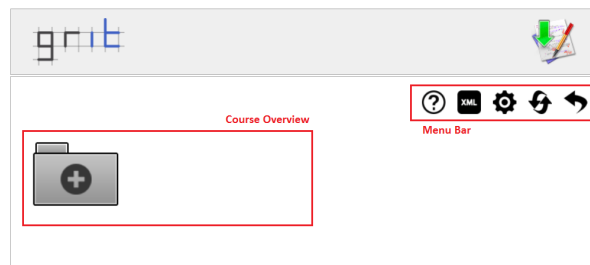
4.3 First Startup

If you are in the `./grit` folder you can start Grit by the following commands

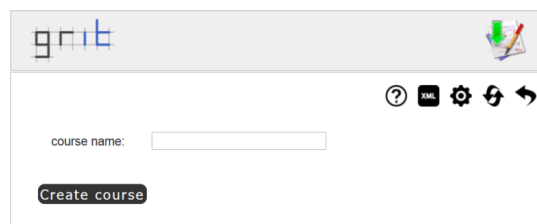
```
./build/install/GRIT/runscript.sh
./gritallinone start
```

After booting up, the web page will be available via the 8080 port of the (virtual) Machine Grit is running on. You will need to activate Javascript in your browser to properly use the website. You will see a username and password prompt. The default username is `username` and the default password is `password`.

On your first visit of the site you will be presented an empty course overview and the menu bar.



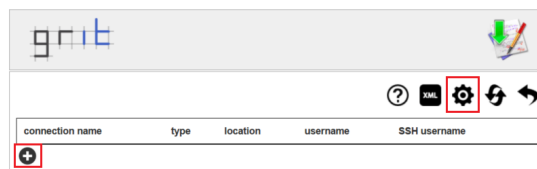
Now you can create a course. For that click on the folder with the plus sign.



You will be prompted to enter the course name, after that click on **Create course**. The created course will now be visible in the overview.



For creating an exercise we will first need to set up a connection so Grit knows where to get the students' submissions from. For that click on the "gear wheel" symbol in the menu bar. You get now presented the connection overview.



To add a new connection click on the plus sign. You have to add the following information:

 A screenshot of the Grit connection creation form. The menu bar is the same. The form contains the following fields:

- connection name:
- connection type:
- location:
- username:
- password:
- structure:

 At the bottom is a 'Create connection' button.

Connection Name: Name of the connection.

Connection Type: Either SVN or Mail are supported by Grit 1.2.

Location: The url of the root folder where the submissions will be. The root folder has to contain the `students.txt`

Username: The name of the SVN account.

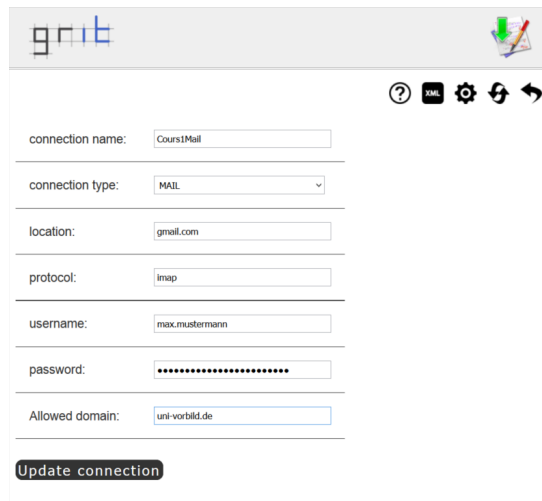
Password: The password for the SVN or Mail account.

Structure: The structure that Grit will encounter so it knows where to get the submission.
For more see Section 4.4.

If you want to fetch submissions from an email server you can switch Connection Type to Email. The window will change slightly. Now there are two new fields.

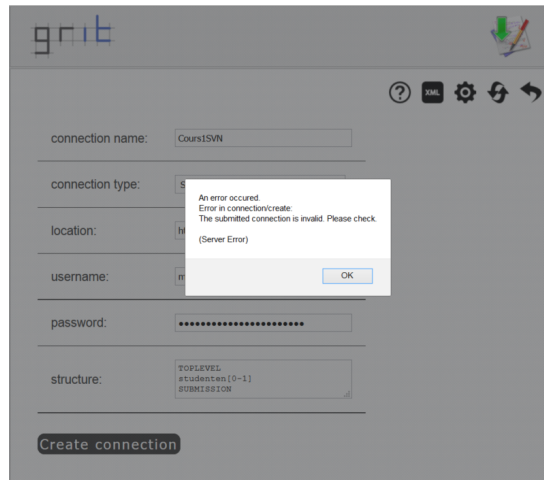
Protocol: Name of the email protocol used for message retrieval, like pop or imap.

Allowed domain: Email domain whose email should be seen as submissions.



The screenshot shows the 'Create Connection' dialog in the Grit application. The dialog has a title bar with the 'grit' logo and a close button. Below the title bar is a toolbar with icons for help, XML, settings, refresh, and back. The main area contains several input fields: 'connection name' (Cours1Mail), 'connection type' (MAIL), 'location' (gmail.com), 'protocol' (imap), 'username' (max.mustermann), 'password' (masked with dots), and 'Allowed domain' (uni-vorbild.de). At the bottom is an 'Update connection' button.

After pressing the **Create Connection** button Grit automatically checks the correctness of the provided connection. You will see the following window if something was wrong with the input (i.e. connection could not be established).



Check the log file for more information, if you encounter this error. If everything works you see the created connection.

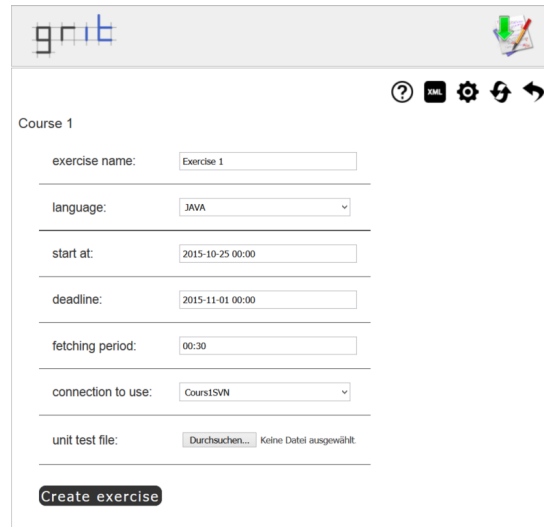
Go back to the course overview. Either by clicking the "back arrow" or by clicking the Grit logo.



Click on the course folder to set up the exercise. In the course folder you create a new exercise by pressing the plus sign.



The following form will appear.



The screenshot shows the Grit web interface for creating a new exercise. At the top, there is a header with the 'grit' logo and a small icon of a green arrow pointing up. Below the header, there is a section titled 'Course 1'. The form contains several input fields: 'exercise name' with the value 'Exercise 1', 'language' with a dropdown menu showing 'JAVA', 'start at' with the value '2015-10-25 00:00', 'deadline' with the value '2015-11-01 00:00', 'fetching period' with the value '00:30', 'connection to use' with a dropdown menu showing 'CourseSVN', and 'unit test file' with a button 'Durchsuchen...' and the text 'Keine Datei ausgewählt.'. At the bottom of the form, there is a button labeled 'Create exercise'.

Exercise Name: The name of the Exercise that will also be printed on the output document (i.e. Exercise 1/Übung 1).

Language: The programming language for the Exercise. Java, C/C++ and Haskell are supported as of Grit 1.2.

Start at: The date the exercise gets handed out to the students.

Deadline: The date when the students must have submitted their submissions.

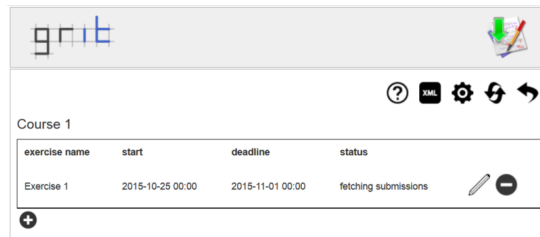
Fetching period: The time intervall when grit checks for new submissions.

Connection to use: Here you will select the connection, previously created, to be used with the exercise.

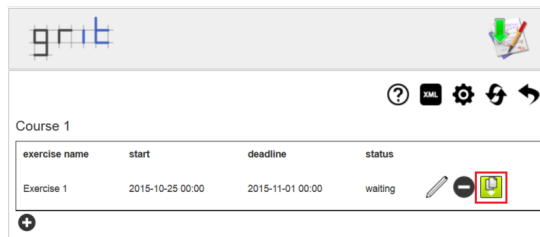
Unit Test file: The (optional) file that contains Unit Tests to test for. Only JUnit is supported by Grit 1.2.

Please consider that the fetching periods have to fit into the interval between start time and deadline. So if you create an exercise e.g. between 10:00 and 18:00, the fetching period must not be 45 minutes because then the last fetch will be 15 minutes past the deadline. Also you should create the exercise *before* the start time, since otherwise the fetching period will be counted from the creation time.

Now you can click `create exercise` and you will see the created exercise in the overview.



As long as the exercise runs, grit automatically checks for new submissions. This is shown by the status field which changes overtime from waiting, fetching, processing to waiting until the deadline is met. If grit waits until it fetches again, you can download a temporary report file by pressing the yellow button. After the deadline is passed you can download the final report at the same location.



4.4 Setting up a submission structure

In order to be able to correctly match student submission files to students in SVN repositories, Grit needs to know the structure of the directory tree it is supposed to look at. For this purpose you need to enter a list of regular expressions which match the path Grit needs to traverse to find submissions. The special tags `TOPLEVEL` and `SUBMISSION` indicate the repository root and the submission level respectively.

For example, your repository might have two folders at its root: `exercise/` and `lecture/`. While the latter contains no submissions, `exercise/` contains a folder for each student: `stud01/`, `stud02/`, `stud03/` and so on. Each of these students has a folder for each exercise: `ex01/`, `ex02/`, `ex03/` and so on.

In order to tell Grit to look in a specific folder (e.g. `ex01`) you would specify the following list:

```

TOPLEVEL
exercise
stud[0-9][0-9]
ex01
SUBMISSION
  
```

`TOPLEVEL` stands for the root of the repository. Then Grit will only look at the `exercise` folder. In there, all folders from `stud01` upto `stud99` will be inspected. In each student

folder Grit looks at the folder ex01. From here the system will retrieve the current submission.

4.5 Further Information about Email and SVN

4.5.1 Email

The e-mail sending function for GRIT allows to set up the system so that e-mails will be sent automatically if any number of the following conditions are complied:

- The file which is delivered by the student is not plausible, i.e the file extension doesn't match with the required one.
- The file which is delivered by the student doesn't compile.
- The subject is invalid

If the SVN-Submission system is used as Connection-Type e-mails will also be sent automatically if any number of the following conditions are complied:

- The student hasn't committed anything 12 hours before the deadline is expired
- The student hasn't committed anything 6 hours before the deadline is expired

If you decided to use email submissions, you already have specified the email address from which Grit should fetch email submissions. In order for this to work as intended, students have to name the email subject according to the following formula:

[course name-exercise name]

E.g. when the course name is "Course" and the exercise name is "1" (for the first exercise), then the subject should be [Course-1] so Grit can find the submission(s). Also consider that submissions handed in after the deadline are not imported.

4.5.2 SVN

For SVN you have to create a different connection for every exercise because the submission structure in the connection specifies which exercise you want to fetch from the repository.

5 | Frequently asked Questions

1. *What does Grit do?*

Grit assists you in running a programming course where it is required for students to hand in assignments done in code. The submissions can be automatically fetched from associated repositories and automatically tested. Grit delivers a well-structured and comprehensive document ready to be graded by a corrector as well as overall and detailed course statistics.

2. *What does Grit not do?*

Grit integrates between the steps assignment issuance, submission and correction. To be precise, it fills the formerly manual between submission and correction, as well as everything that comes after correction. Grit does not provide facilities to issue assignments, submit them, and, certainly cannot correct them.

3. *On what type of machine should I install Grit?*

It is possible to install Grit on different Linux systems (It is tested on Ubuntu, Debian and OpenSUSE)

4. *What is the default username and password?*

The default username is `username` and its password is `password`.

5. *Should I change the predefined passwords of the provided VM?*

Yes.

6 | Credits

6.1 Developers Team GRIT

Gabriel Einsdorf

Marvin Gülzow

Eike Heinz

Marcel Hiller

David Kolb

Fabian Marquart

Thomas Schmidt

Stefano Woerner

6.2 Developers Team VARCID

Verena Berg

(Verena.Berg@uni-konstanz.de)

Annelie Bruder

(Annelie.Bruder@uni-konstanz.de)

Robert Schmid

(Robert.Schmid@uni-konstanz.de)

Cedric Sehrer

(Cedric.Sehrer@uni-konstanz.de)

Iris Schmidt

(Iris.Schmidt@uni-konstanz.de)

Dirk Oehler

(Dirk.2.Oehler@uni-konstanz.de)

Onur Cakmak

(Onur.Cakmak@uni-konstanz.de)

6.3 Special Thanks

Arno Scharmann

Dr.-Ing. Ernst de Ridder

Sigmar Papendick

Simone Winkler

Tino Klingebiel

Stefan Brütsch