E-Learning – VELS Specification

Project: E-Learning – VELS
Author(s): Andreas Platschek, Martin Mosbeck
Reviewer: Axel Jantsch
Version: 0.6
Date: December 29, 2016

Copyright: TU Wien,
Institute of Computer Technology

Contents

Version	Autor	Datum	Kommentar
0.1	Andreas Platschek	02.07.2015	First shot at the specification.
0.2	Andreas Platschek, Martin Mosbeck	10.07.2015	First Draft for Review.
0.3	Andreas Platschek, Martin Mosbeck	20.09.2015	Updates from the implementation.
0.4	Martin Mosbeck	28.05.2016	Updates for public release.
0.5	Martin Mosbeck	04.10.16	Updates about configurability.
0.6	Martin Mosbeck	16.10.16	Updates for skip and auto_advance.

1. Introduction

The following document specifies E-Learning platform for a Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL). This platform will first of all be used for the lectures Mikrocomputer and Digitale Integrierte Systeme. The main goal is to provide exercises to the students, that can be done from home, checked immediately and decrease the efforts of lecture (actually the lecturer will have an O(1) instead of an O(N)).

An important point is, that the tasks should be different for every student – a basic version of the example will be provided along with some parameters that are subject to change. These parameters are then randomized in order to generate as many different examples as possible.

For the students, the interface shall be a very well known one – it is based on E-Mail (which can be assumed to be known to all students). This also has the advantage, that no usernames/passwords have to be given to the students, only an e-mail address they can use to get examples from and send solutions to. All possible interactions that can be performed by students are described in Section 2.

The system itself is specified in Section 3.

NOTE: in the following *Structured Analysis* will be used to specify the system. If you are not familiar with this method, you can find details e.g. in [DeM81, Goo01, Coo03].

2. Interaction with the system

The VELS E-Learning system has 2 distinctive user groups: course operators and students. To satisfy the use cases for each of the 2 user groups the following interfaces have to be defined:

- VELS Email Interface
- VELS Web Interface
- VELS direct server access

The different use cases for students and operator and the responsible interface can be seen in Table 1 and Table 2.

Use Case	Responsible interface
register with the system	VELS Email Interface
get status in course	VELS Email Interface
get current task	VELS Email Interface
submit a task	VELS Email Interface
ask a question	VELS Email Interface

Table 1: Use cases for students

Use Case	Responsible interface
configure a course	VELS Web Interface
modify task generation or testbench generation	VELS direct server access
create a new task	VELS direct server access
view the progress for students	VELS Web Interface
view task statistics	VELS Web Interface
read log files	VELS direct server access

Table 2: Use cases for course operators

2.1. VELS Email Interface

The VELS Email Interface is the primary interface for students. Students are uniquely identified by their email address to interact with this interface. The student has to interact with a single, non-changing email address during the whole duration of a course. This email address should contain the student id ("Matrikelnummer") therefore only the generic University of Technology email address can be used by the students. These email addresses have to be added to the system's whitelist by a course operator in order to enable them to be authorized for interaction.

The different actions a student can take are defined via email subject and address. The following cases are possible:

- *Sender email address is not on whitelist:* The student is sent an e-mail that he should use his generic University of Technology email address to interact with the system. If that does not help, he should contact on of the course operators.
- *Sender email address is on whitelist, arbitrary Subject:* If the student is not registered with the system, registration will be initiated. Registration with the system creates the appropriate data entries in the database for the student. If adding the new user was successful, two e-mails are sent to the user: a welcome message with general information on how the system works and if already available the description of the first task. If the student is already registered he is sent an e-mail with the usage for this interface.
- *Subject contains the word "Question":* The student has a question about the course, this question is forwarded to all course operators. The student receives a confirmation that the question has been passed on.
- *Subject contains the phrase "Question Task N" where N is the task number:* The student has a question about the course, this question is forwarded to all task operators for task N. The student receives a confirmation that the question has been passed on.
- *Subject contains the phrase "Result Task N" where N is the task number:* The student wants to hand in a solution, for Task N. Students are allowed to hand in solutions for all tasks that have already been solved previously, as well as the one task that has not been solved yet – they are not allowed to hand in solutions for tasks they did not receive the task description yet. The solution is tested on the server. If the task solution was appropriate the student will receive the next task (or a congratulations message, if no more examples are available). If the tests were not successful, the student receives an error message that gives a hint on what may be wrong (output of the simulation tool, expected output vs. actual output, etc.)

- *Subject contains the word "Status":* The student is send an e-mail with the list of his completed tasks and his current task description.
- *Subject contains the word "Skip":* The student will be skipped to the task after his current task if this feature is activated. Otherwise he is sent a usage message.
- *default:* The student is sent an e-mail with the usage for this interface.

2.2. VELS Web Interface

The VELS Web Interface is a configuration and status tool for the course operators.

Course operators can create a course. A course consists of a task queue, which explicitly orders a selection of the available tasks. The following actions for course configuration shall be implemented:

- Configuration of the task queue (position of task in the queue, start and end date for the task, points for task completion.
- Configure course registration deadline
- Modify course registration email whitelist
- Set Number of tasks
- Modify Archive Directory
- Set Administrator E-Mail

Course operators can view the progress of each student and is presented with the following informations:

- Student name and student identification number ("Matrikelnummer").
- Task the student is working on and the tasks the student has completed.
- Score which was accumulated by the student.
- Number of submissions for each task the student has received.
- List of attached files for each task the student has received.

Course operators can view statistical informations about his course:

- Number of wrong and right submissions for each tasks.
- General statistical informations about his course (e.g number emails received/sent).

The VELS Web Interface may also be used by the course operator to read the results at the end of the semester, that is the points that have been scored by the individual students.

2.3. VELS direct server access

The VELS direct server access can be used by course operators to generate new task, modify code for task generation, description and testing of existing tasks, read log files or fix bugs of the VELS system. It also can be used to view every submission a student has done. These procedures will be described in the VELS Usermanual.

3. Submission System

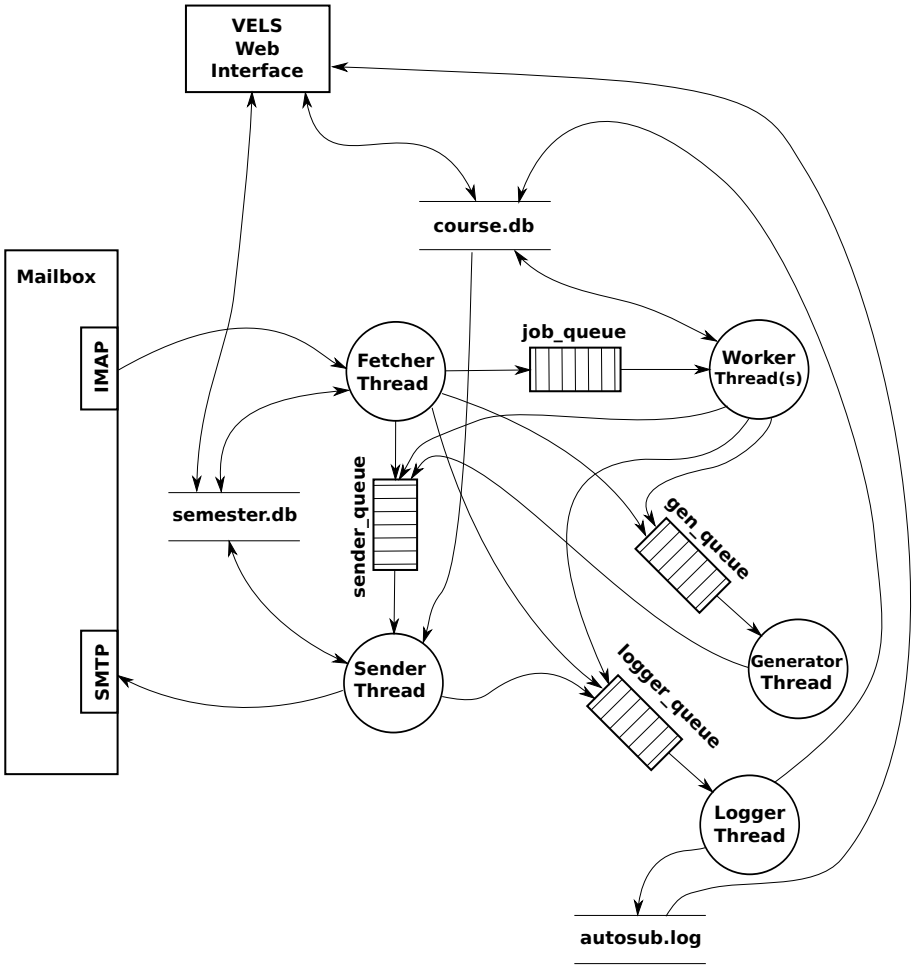

Figure 1: DFD0 – High-Level Data-Flows in the Submission System.

3.1. Description of Entities

MB Mailbox

Description	Mailbox to interface with students.
Rate	Asynchronously by students, once per minute by the E-Learning system.
Comment	The Mailbox is used to interface with the students. In depth description of this interface can be found in Section 2.1. The Mailbox is accessed via Internet Message Access Protocol (IMAP) for reading new e-mails and Simple Mail Transfer Protocol (SMTP) to send e-mails. After the action triggered by an received e-mail has been processed, the e-mail shall be archived into a configurable folder on the e-mail server.

1 Fetcher Thread

Description	Fetch E-Mails from the Mailbox.
Rate	Periodically once a minute (by default – period shall be configurable).
Comment	The Fetcher Thread periodically checks the Mailbox for new mails. These new mails are sorted in the fetcher thread an actions are triggered based on the messages. To decide which action is the correct one for a particular e-mail, the user E-mail address and the subject are evaluated. The following use cases are possible. • New user has to be created. • Task wants to be submitted. • Question wants to be asked. • User status wants to be known. Further Information concerning the email interface can be be seen in Section 2

2 Worker Thread(s)

Description	Test the task submissions of students.
Rate	Asynchronously triggered upon arrival of a new task submission by a student.
Comment	The worker threads are the ones which do the actual work of testing submissions by a student. If the Fetcher Thread receives an e-mail with a result submission, a new message is added to the job_queue. The worker threads are using a blocking read on the job_queue to wait for work, as soon as the new message is added by the Fetcher thread one of the workers will read it and start the tests for the users submission. The tests themselves are implemented externally in a bash script. This script is expected to return with 0 on success, and 1 if the tests failed. Additional information (e.g. error-messages) are expected in a file called `error_msg` in the unsuccessful case.

3 Sender Thread

Description	Sends E-mails to the Students
Rate	Triggered by Fetcher, Worker or Generator Thread.
Comment	If a message has to be returned to the student, the thread that wants to send the message just puts the necessary information into the sender_queue. The sender thread then takes that information, formats an e-mail and sends it out using SMTP. The unique message-id is passed from the fetcher (possibly via other threads) to the sender thread. If the sender thread has sent out the answering e-mails, the e-mail with the given message-id is moved into an archive folder on the mail server.

4 Logger Thread

Description	Format log messages and store them in the log file.
Rate	Triggered by all other threads.
Comment	In order to enforce a common logging format, and allow the setting of a common log-level, all threads just put their log-messages into the logger_queue. The logger thread decides (depending on the log-level) whether or not the message shall be logged or not. If so, the message is formatted and written to the log file. The following log-levels shall be implemented (from lowest to highest): **DEBUG:** Print all kind of information that might be helpful to debug problems. This includes, what user sent which e-mail, what action was taken, what was the result of that action. **INFO:** Information that might be of interest, even if not in debugging mode. **WARNING:** A problem occurred, but it did not lead to a long-term problem (e.g. could not connect to the mailserver, but after retrying it worked). **ERROR:** A problem that lead to a long-term malfunction of the submission system occurred, the system will not be able to recover from this problem.

5 Generator Thread

Description	Generate a (unique) example for a user.
Rate	Triggered by registration of a new user, and successful submission of a task.
Comment	The tasks are tailored for each student using randomized parameters. The generation of a task is done as soon as it is needed e.g. for the first task after the user registered, for all the other tasks, as soon as the previous task has successfully been submitted. The Generator Thread is blocking on the gen_queue, until the generation of a new example is requested. As soon as the example was generated, a request to send it out is put into the sender_queue.

3.2. Description of Message-Queues

job_queue The job_queue is used to trigger worker threads to start testing solutions that were received. The messages in the job_queue are comprised of the following fields:

- **UserId:** The unique User ID of the user who submitted this solution.
- **UserEmail:** The e-mail address of the user who submitted this solution.
- **taskNr:** Number of the task that solution has been submitted for.
- **MessageId:** The unique message-id of the e-mail on the server.

sender_queue If a thread wants to send an e-mail to a user, the sender thread is notified via this queue. The data needed by the sender thread is in the messages in the sender_queue. These messages consist of the following fields:

- **UserId:** The unique User ID of the user who shall receive this e-mail.
- **recipient** E-Mail address of the recipient (the content of the 'To' field in the e-mail).
- **message_type** The message type is used to decide on how to format the e-mail, and whether or not additional e-mails have to be sent out (e.g. when a question is handled). Possible message_types are:
  - `Task` – A task description.
  - `Success` – A Task has been tested successfully.
  - `Failed` – An error message for a failed test-run of a result submission.
  - `SecAlert` – Scanning the code revealed that this might be an attack on the system.
  - `TaskAlert` – An error message for failures in files that were created for tasks.
  - `InvalidTask` – A result submission for a non-existent task.
  - `Usage` – An e-mail with usage explanation shall be sent to the student.
  - `Question` – Confirm that a question was received.
  - `QFwd` – Forward a question to the administrator.
  - `Welcome` – Send a welcome message to a new student.
  - `NotAllowed` – A user who is not on the whitelist sent a mail to the system.

- `SkipNotPossible` – A user wants to skip to the next task, when he is either at the last task or the next task has not started yet.
- `TaskNotSubmittable` – A user wants to submit to a task which has not started yet.

- **Task** Number of the task that message concerns (e.g. the current one if the test failed, the next one if the task description shall be sent).
- **MessageId:** The unique message-id of the e-mail on the server.

logger_queue Trigger the logger to write a log message about an event that happened.

- **msg:** The text that describes the event that shall be logged.
- **type:** The log-level of this log message; available log-levels are DEBUG, INFO, WARNING and ERROR (see specification of Logger Thread in Section 3.1 for details).
- **loggername:** Name of the thread that reported the event that shall be logged.

gen_queue The gen_queue is used to trigger the Generator Thread to generate a new (variant of) a task. That means that certain parameters of the task are randomized, in order to assure that each student receives his/her very own example.

- **UserId:** The unique UserID of the User for whom the task is generated.
- **UserEmail:** E-Mail address of the User for whom the task is generated.
- **TaskNr:** Number of the task that shall be generated.
- **MessageId:** The unique message-id of the e-mail on the server.

3.3. Description of Datastores

3.3.1. autosub.log

The log file shall contain log-messages formatted as follows:

```
<date> <time> [<loggername>\t] <log-level:> <log-message>
```

Some examples for log-messages could look like this:

```
2015-06-20 20:36:32,813 [Worker1     ] INFO: Starting Worker1
2015-06-20 20:36:32,813 [Worker2     ] INFO: Starting Worker2
2015-06-20 20:36:32,814 [Worker3     ] INFO: Starting Worker3
...
2015-06-21 08:06:44,526 [fetcher     ] WARNING: Failed to select inbox
...
2015-06-22 16:05:51,255 [fetcher     ] DEBUG: closed connection to imap
server
```

3.3.2. .cfg File

A .cfg file may be passed to the submission system when the daemon is started. If no .cfg file is specified, the used file is default.cfg. The groups and fields in the config file are as follows:

Group	Field	Description
imapserver	servername	Hostname.domain of the IMAP server.
imapserver	serverport	Port to be used.
imapserver	security	Security protocol to be used when connecting Possible values: none ssl starttls
imapserver	username	Username to be used to login.
imapserver	password	Password to be used to login.
imapserver	email	Email address at the IMAP server.
smtpserver	servername	Hostname.domain of the SMTP server.
smtpserver	serverport	Port to be used.
smtpserver	security	Security protocol to be used when connecting. Possible values: none ssl starttls
smtpserver	username	Username to be used to login.
smtpserver	password	Password to be used to login.
smtpserver	email	Email address at the SMTP server.
general	num_workers	Number of Worker threads. This influences how many tests can conducted in parallel.
general	queue_size	Size of the thread communication queues.
general	poll_period	Period in seconds at which the mailbox is checked.
general	course_name	The name of course.
general	semesterdb	The name and path of the database that shall be used for data specific to this semester. (default: semester.db)
general	coursedb	The name and path of the database that shall be used for data specific to this course. (default: course.db)
general	logfile	The name and path of the autosub logfile. (default: autosub.log)
general	auto_advance	Decides if users get auto advanced to a task which is activated by its TaskStart. Possible values: yes no (default:no)
general	allow_skipping	Allows users to skip ahead to the next task with a skip email. Possible values: yes no (default:no)
challenge	num_tasks	Number of tasks in the course.
challenge	mode	The mode in which to run (exam or normal).

3.3.3. semester.db

The semester database is used to store all data that is specific to one semester. This includes, users, parametrization of tasks assigned for users, statistics, etc. .

The database semester.db contains the following tables (and entries in those tables):

Table Name	Users
Description	Used to collect all necessary information on the Students.
Table Entries	• **UserId**: A unique UserID given at registration time (when the first e-mail is received from the users e-mail address). • **Name**: The name of the user as specified in the "from" field of the e-mail. • **Email**: The e-mail address of the user as specified in the "from" field of the e-mail. • **FirstMail**: Timestamp of the first e-mail received by this user. • **LastDone**: Timestamp of the email that contained the successful solution of the last task (only if this user has already finished the last task). • **CurrentTask**: The task the user is currently working on.

Table Name	TaskStats
Description	Contains one entry for each available task, collecting statistics on the individual tasks.
Table Entries	• **TaskId**: Unique ID of the task. • **NrSubmissions**: Number of solutions received for the task with this TaskId. • **NrSuccessful**: Number of correct solutions received for the task with this TaskId.

Table Name	TaskCounters
Description	Implements counters for certain events, examples for such events are: e-mail received, e-mail sent, question received, new user.
Table Entries	• **CounterId**: Unique ID of the counter. • **Name**: Name of the counter. • **Value**: Value of the counter.

Table Name	UserTasks
Description	Used to map configured tasks to users – e.g. store the generated examples so they can be fetched later on for testing.
Table Entries	• **TaskNr**: Unique number of the task. • **UserId**: Unique ID of the user – the combination of TaskNr and UserId make the entry unique. • **TaskParameters**: Either the parameters that describe the setting for this particular student, or a value that can be used to derive the parameters from. • **TaskDescription**: Message that describes the task that was specifically generated for the student. and shall be sent to the student. • **TaskAttachments**: List of attachments that shall be sent to the student (path+filename). • **NrSubmissions**: Number of submissions the student has done for this task. • **FirstSuccessful**: Number of the first successful submission.

Table Name	UserWhiteList
Description	A Whitelist of E-Mail addresses that shall be authorized to interact with the system.
Table Entries	• **UniqueId**: Unique ID in the whitelist table. • **Email**: E-mail address that shall be authorized to interact with the system.

3.3.4. course.db

In contrast to semester.db, course.db consists configuration data that will very likely be reused in the next semester. This seperation makes it very easy to start a new semester: just backup semester.db to some safe location and remove it in the original location (a new empty one will be made automatically).

Then use the VELS web interface to perform the small modifcations (year/semester, deadlines, etc.) needed in course.db.

Table Name	SpecialMessages
Description	A collection of texts that shall be sent, in the case special events.

Table Entries	• **EventName**: Currently the following events are implemented:
	  • WELCOME – A welcome message that is sent upon first e-mail from a student and that gives an explanation of how the system works.
	  • USAGE – In case an e-mail that can not be interpreted is received, this message shall be sent to the user.
	  • QUESTION – A message that is sent as a confirmation if a question was received.
	  • INVALID – A message that is sent, in case a result for an invalid task has been received.
	  • CONGRATS – A message that congratulates the student upon solving the last task.
	  • REGOVER – A message sent in case a student tries to register after the registration deadline has passed.
	  • NOTALLOWED – A mesage sent in case the student is not on the whitelist.
	  • CURLAST – A mesage sent to inform the student that he has solved the current last task.
	  • DEADTASK – A mesage sent to inform the student that he submitted a task which's deadline is allready over.
	  • SKIPNOTPOSSIBLE – A message sent to the user that the skip he requested was not possible. Only sent if skipping is allowed.
	  • TASKNOTSUBMITTABLE – A message sent to the user to tell him that he cannot submit a solution to a task he has not received yet.
	• **Text**: The text that shall be sent in case of the event **EventName** occurs.

Table Name	TaskConfiguration
Description	Used to configure tasks, including their order, the scripts used to test submissions, etc.
Table Entries	• **TaskNr**: The unique number of the task – this number is used to establish the order of tasks as received by the students. • **TaskStart**: Timestamp of when the task shall be available for students (if any). • **TaskDeadline**: The deadline until which the task has to be successfully submitted (if any). • **PathToTask**: The root directory of the task (so it does not have to be specified for all the entries). • **GeneratorExecutable**: The executable (script, binary, etc.) used to generate unique examples for each student. If this is not set, all students will receive the same task. • **TestExecutable**: The executable (script, binary, etc.) used to test the results submitted by the students. • **Score:** The points the student scores by solving this task. • **TaskOperator:** The e-mail address of the course operator, who is responsible for this task.

Table Name	GeneralConfig
Description	Store some general configuration for the semester.

Table Entries	• **ConfigItem**: Name of the Configuration Item. The following configuration items are available:
	  • `registration_deadline` – the deadline for students to register.
	  • `num_tasks` – the number of tasks available in this semester.
	  • `archive_dir` – the name of the directory on the mail-sever into which the answered messages shall be moved.
	  • `admin_email` – the email of the admin for the course.
	  • `challenge_mode` – the mode for the course : normal or exam.
	• **Content**: Content of the Configuration Item.

Abbreviations

IMAP Internet Message Access Protocol. 7

SMTP Simple Mail Transfer Protocol. 7, 8

VHDL VHSIC Hardware Description Language. 1

VHSIC Very High Speed Integrated Circuit. 1

References

[Coo03] Jim Cooling. *Software Engineering for Real-Time Systems*. Addison Wesley, first edition edition, 2003.

[DeM81] Tom DeMarco. *Structured Analysis and System Specification*. Yourdon P.,U.S., april 1981 edition, 1981.

[Goo01] Hassan Gooma. *Software Design Methods for Concurrent and Real-Time Systems*. Addison Wesley, fifth printing edition, 2001.