Fachhochschule Nordwestschweiz
Hochschule für Technik

MAS-IT

Business frameworks

Masterthesis von

Etienne Rebetez

FHNW

Hochschule für Technik

Studiengang MAS-IT

Betreuender Dozent:

Dr. sc. techn. Ronald Tanner

Bern, 16. Juli 2012

Summary

Text

Vorwort

Text

Inhaltsverzeichnis

1 Introduction

Laboratory processes. What is that all about?

Usually it goals like this. A physical sample is brought to the quality control (QC) analytic laboratory. The laboratory determines certain physical property's and then tell the customer about it.

A simplified view looks like this. A sample of a certain substance comes into the QC lab. What analysis have to be made etc. is given by the Labor information and management system (LIMS). The LIMS also says when and what sample has to be analyzed. The lab performs then the analysis. All used solutions, instruments and steps have to be documented. That is traditionally done on paper. At this point it is important to emphasize that everything that is not documented isn't done from an audit point of view. Also the standard operating procedure (SOP) has to be followed exactly. After the analysis it is mostly used to have a calculation step where certain parameters are calculated to the final value. The final value is then transferred back to the LIMS. These are mainly manual steps. These manual steps can lead to errors and are time consuming. Since the operating of balances, creation of solution or other practical tasks cannot be automated, the data flow and the process can.

This report aims to look at the layer between LIMS and the instruments and the analysis method. In the past few years there has been a ton of programs that aim to solve that problem. Some try to achieve that with an almighty LIMS. But since most LIMS date from the 90's the implementation is not easy and you end up haven a not maintainable monolith. There was also the idea of electronic lab notebooks (ELN) these ELN where usually conceived for R&D laboratory. That means more freedom and less strict processes. Then there is the software for the instruments, that can perform more and more tasks. This looks like a good idea, but there are so many instruments, methods and manufactures that there is just no solution that fits all. Also it is still the case, that one solution of one manufacturer is in no way compatible to the solution of another manufacturer.

2 The Laboratory landscape

The laboratory knows many different analytic methods. They all have different requirements and provide different forms of results. That gives us an extremely

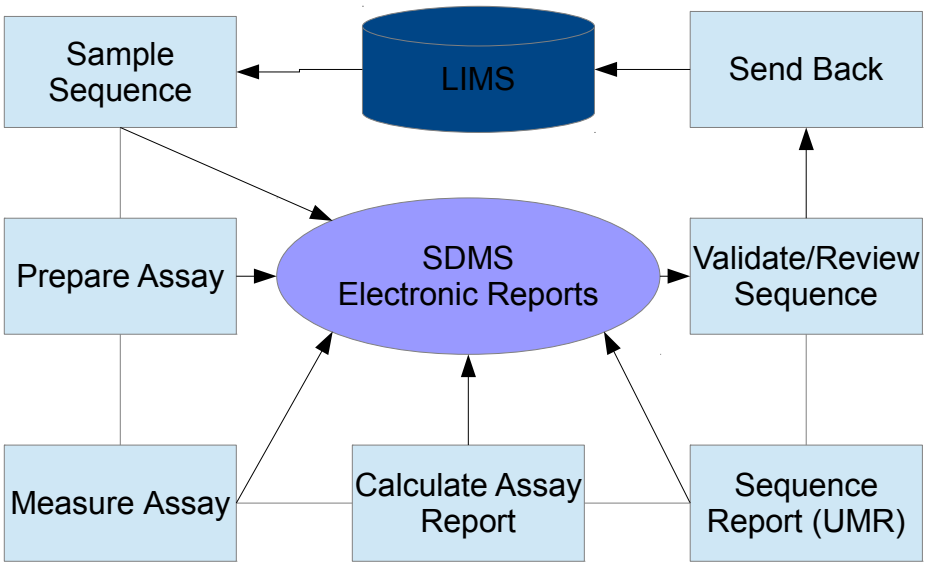

Abbildung 1: The current lab work flow

heterogeneous landscape.

First we need to look at the current laboratory work flow, with the given programs. Historically, and still so, the lab uses the LIMS to see what samples have do be done and to type the results back. The LIMS is a height level program that was designed to managed samples.

To normalize the data and provide an easy review ability of every step, a report, that can be viewed and signed in the SDMS database, is created. The problem with heterogeneous data and raw data reports is currently overcome with the SDMS database. It is possible to print reports into the database and tag them automatically. As a result the reports printed into that database can be parsed and data is so available. This currently is in place and works pretty well.

3 Concept

So the question came up, why not look at something completely different. The lab is not the only business that has processes. There could be other business frameworks that provide the needed functionality. This system should be able to integrate and interface these different islands of programs. It would be like a laboratory middle ware.

To make the automation in such a heterogeneous landscape possible a sort of state machine is needed. This state machine knows how process look like, which processes are currently run, in which state they are in and which is the next to execute. This pattern is basically the same in every business process. One of the therms is SOA (Service Oriented Architecture). The state machine engine would therefore be the heart of the hole automation process.

Since the lab is obviously not the only place in the world that has predefined process a look at generic state machines was imminent. To configure these state machine engines there are already several standards. Two process definition languages are already well established:

- BPEL
- BPMN

Both are open standards from OMG [2].

3.1 Requirements

Beside the central process engine, there are also other aspects to be considered. One is a web application that should provide forms and an overview of the tasks. Thes humand interface should also be available on mobile devices. Since the current implementation is based on excel sheets, it of importance that the current applications can be implemented in the new framework. For instance it should be possible to send data to an excel application and get the results back into the process stream.

4 BPEL

The BPEL (Business Process Execution Language) was an answer to the process problem that came from the it world. It is very technical and the process description is made with if's, loops that are also known in common programming languages.

5 BPMN

BPMN (Business Process Model and Notation) has it roots in the business process modulation. It first was only a description language to provide some unified task ore gateway definitions. These are the elements of the common flowcharts. With the versin 2 of the BPMN standard these flowcharts can now be interpreted by an engines and be run as a program.

6 Prozess Engines

The engines are implementations of these business description language. The engines provide all sort of logic that would be hard to implement every time from scratch.

- What processes are there
- which are in what state
- Processes are recoverable and transaction save. (crash)
- What happens if a new version of the process is deployed.
- Who did what and when.

The following chapters present some of these engines that are already available. The list is far from complete and in alphabetical order.

6.1 activeVOS

activeVOS [5] is a product from Active Endpoints, Inc. It uses open standards to implement its feature. The supported notations are BPEL and new also

BPMN. Everything is implemented in Java and XML. The editor you get looks and feels like an eclipse plugin. The application runs on an JavaEE application server.

The documentations look good. It has videos, tutorials and they also provide training.

In addition they have some additional tools to the BPEL engine like reporting.

The cost for the implementations will cost around 50'000 USD [6].

6.2 activiti

activiti [3] is an open source BPMN engine. It was forked from the jBPM [4] from JBOSS. It is therefore a very young project that has the goal to build a rock solid BPMN engine. Activiti is really only the engine that provide an Java api. There is also a example web application Activiti Explorer that shows and demonstrate the ability of the activiti engine.

Because it is open source it is free of charge. Although the effort to get a working product might be bigger.

There is also a company Signavio [12] behind activiti. Signavio provides consulting and BPMN creation tools.

6.3 Apache ODE

Apache ODE [7] is an open source BPEL engine. There is also an eclipse plugin, that helps create the BPEL XML definition.

6.4 ARIS

ARIS [8] is a product form software AG. It is a big enterprise BPM solution that brings many tools like process overviews, dashboards, reports or monitoring. ARIS was developed to work well with SAP. Therefore it is by definition an height level implementation.

You buy a solution and the code is not accessible. software AG provides training for employees to configure or work with the product.

The cost looks to be quit height and is at least 10 Million USD [9].

6.5 biztalk

Biztalk [10] is the BPEL implementation from Microsoft. It is sold as biztalk server 2010 and is praised as Business Proces Management (BPM) suite. Since it is a Microsoft product it is written and configured in the .NET programming language. Obviously the integration with other Microsoft products like Sharepoint should be good.

The enterprise edition costs around 44'000 USD [11]. For testing and development there is a free license.

6.6 Summary of products

Name	Notation	Language	License	Extras
activiti	BPMN	Java	Apache 2.0	only API
activVOS	BPMN/BPEL	Java	proprietary	uses open standars
Apache ODE	BPEL	Java	Apache 2.0	
ARIS	BPMN/BPEL	Java	proprietary	sold as fully feature product
biztalk	BPEL	.NET	proprietary	share point integration

Since either BPMN nor BPEL is in use at the moment the first thing to decide is what specification to use. It does not make sense to mix both.

BPMN 2.0 seems to be the better choice due to the close process view. BPEL exists longer but seems unnecessary complicated. Therefore BPMN is the better choice for the lab process definition.

If we now only look at the BPMN 2.0 engines only ActiVOS, activiti and ARIS remain from the list.

ActiVOS provides some extra GUI's to give ßimpleräccess to the configuration. It creates xml and configurations in the background. They can fortunately be seen and changed from hand. In the bottom line, it seems nice but does not provide a big advantage.

ARIS is the big all inclusive package. It is however impossible to work yourselves on the project they sell also the programming and support. Therefore there is no code ore xml visible in the hole package. So for each change the external company has to be contracted. They provide also some basic training courses.

Activiti has a total other approach. It has no gui what so ever. I provides an open programming interface that provides all functionality of the engine. The xml can be easily edited directly. It is therefore slim and sticks to the one task it tries to solve. The downside is that there is a steep learning curve since everything has to bee implemented in the code. Because it is open source there are already some example web application that can be used out of the box.

6.7 Conclusion

The monolithic concept of big systems has proven in the past to be inflexible and heavy on the maintenance. The decision was made to use activiti because of its simplicity and possibility to add other functions (i.e monitoring or reporting) in the future over other programs that also have one functionality. Activiti plays also well together with Java based web frameworks like vaadin.

6.8 Compatibility

Although BPMN is a standard and provides most keywords and rules the implementations still adds own keywords or extra functionality (i.e forms). So the BPMN files can't just be copied form one engine to another. If you chose one engine it is kinda final and you will stick to it.

If an engine change really has to be made it is however simpler to achieve than if you had the complete logic inside the code. If the new engines even uses the same programming language, it is very probable, that you can just reuse the business logic classes.

7 Example Application

The process takes an annual inventory item review. These items are reference samples of the substance that were sold.

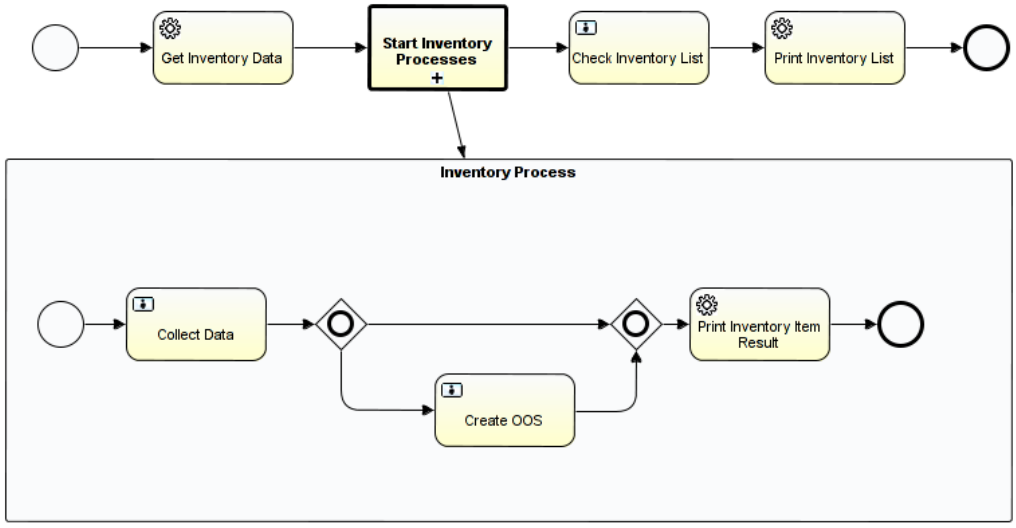

Abbildung 2: Example Application work flow

First a list of the reference samples that have to be checked is retrieved from the parent system (i.e LIMS). This is done with a SQL query on the database. The sample number and the location is then stored in a process variable as a data object list.

For each item in the list a sub process is started in parallel (callActiviy). The sub process is a separate BPMN process definition. The data object which represents a sample, is given to the newly created process.

Now the user can claim the tasks or in this case the samples they wish to check. Then they go to the location and provide the verdict of that sample. If the sample is fine a report is created otherwise a OOS (Out of specification) procedure has to be started first. The OOS process is not part of this demonstration application. It would however be very easy create and to call another sub activity in the future. To print the reports the existing infrastructure is used. The reports are stored in the SDMS and can be signed there.. Once all samples have been controlled, the parent process goes to the next task.

The laboratory supervisor now gets the task to check the sample list. An overview of the sample list and the verdicts is presented to him. The list with all samples is then again printed to the SDMS. That concludes the review process.

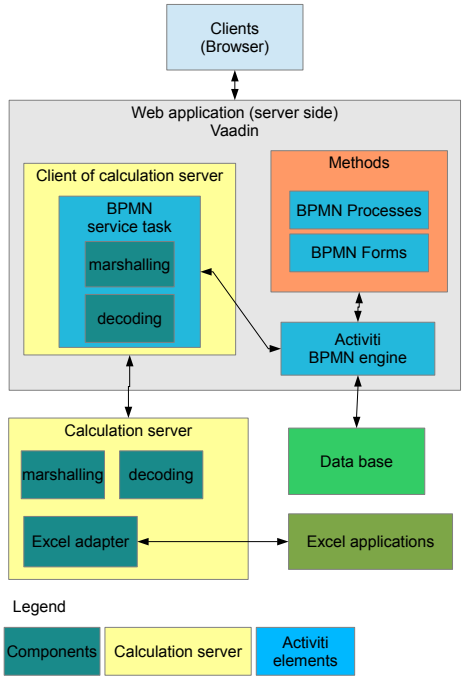

Abbildung 3: Overview of the architecture

This process is used to illustrate the further examples and references.

8 Software Architecture

webapp

activti

componetns

calc server

db

9 Webapplication

Besides the BPMN engine, the web application is a central cornerstone of this project. After all that is what the user gets to see. The decisions was made to use Vaadin as a web framework. It allows to program the web application completely in java on the server side. So there is no need to get in trouble with web technology.

As the back end activiti is used. So the user management, transaction safety and business logic is taken care of.

To glue all the frameworks together, spring is used. In spring the configuration is made in a xml file. As a result the code doesn't has to have any knowledge of the environment it is in. The code can therefore be untouched when deployed to a productive system.

9.1 class design

The web application is a simple GUI application. There is the main window that holds different widgets in a layout. Each widget shows a certain information. For example there is a process viewer that presents the available BPMN processes. The following widgets are implemented in the demo web application.

- process viewer: shows available main processes
- task viewer: unassigned tasks
- my task viewer: shows the assigned tasks of the current user
- start task by id: tasks can be started by a give id
- form viewer: shows the form of the task that is executed
- history view: shows information about finished tasks and processes

There is a listener interface that can be implemented by these widgets. The task viewer listens to the process viewer if a new task is started. The my task viewer listens to the task viewer to see if a task is being claimed. And the form viewer checks if a task is executed. When the task is finished the task viewers are updated to show the remaining tasks.

Since activiti is being configured over spring, the activiti instances can be provided as singletons to each widget that needs them. This way the process

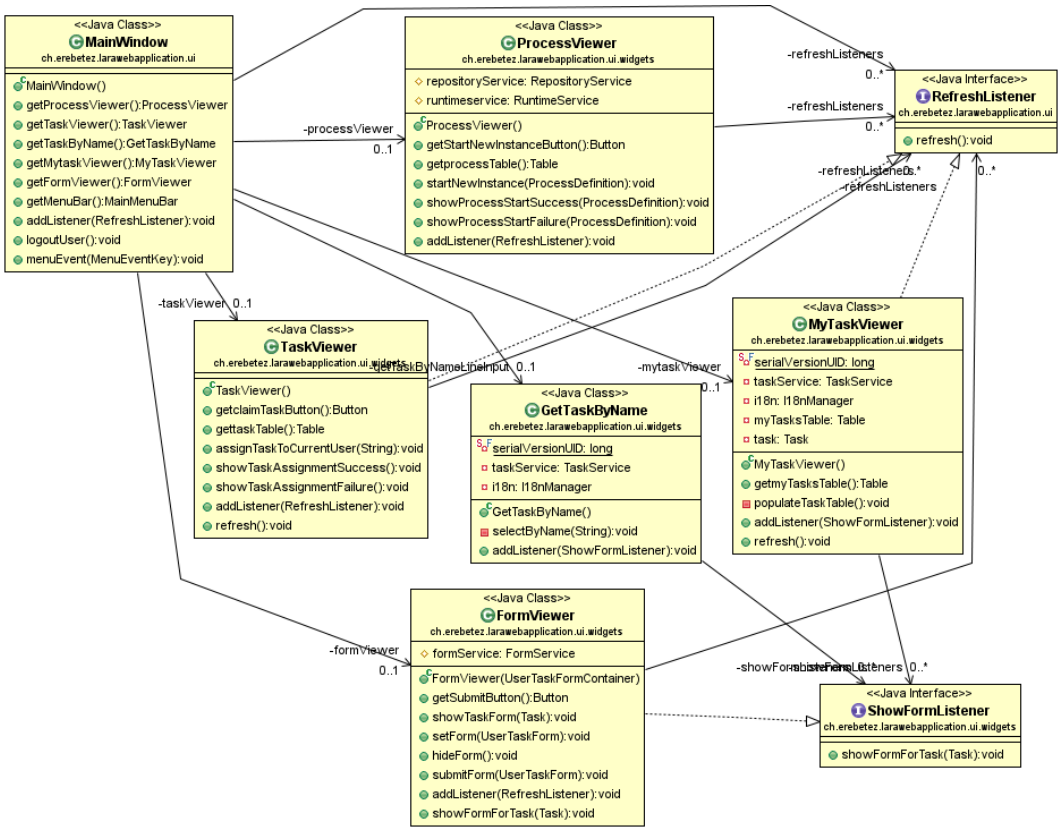

Abbildung 4: UML diagramm of main web application classes

logic is being separated from the user interface.

9.2 BPMN deployment

The demonstration web application deploys the BPMN processes to the activiti engine at start up. In the future and for productive use of activiti the BPMN process deployment should be done over a separate applications or as a function inside the web application.

9.3 Internationalization

Internationalization is only needed for the web application. Since the web application is written in basic java, the basic resource pattern can be used. That means each view able string is loaded from a resource file that holds the strings of the given local.

9.4 ldap

10 External application server

The basic concept is a socket client rpc server architecture. [14]

10.1 server

The server receives a request to execute a certain excel application. It then executes this application. If multiple request are received they get stored in an array and then the external excel applications are run in serial.

10.2 client

The client is a process on the BPMN engine server. This process can be called by a parent process (see callActiviy). The first task is a java service task. It creates a connection to the server and sends the data. It waits until the server sends the processed data back.

Abbildung 5: Work flow of external client call process

If there is an error a boundary event is fired and a user or e-mail task is started. When the problem is resolved the data is send again to the server.

10.3 protocol

For the communication between both parties a protocol has to be defined. First the client send a request to the server that has an UUID [15], the application to start and the marshalled data object. Each information is separated by two ::. The Information then looks like that.

The server is then responding, that it got the request by sending back the UUID with an !!!!ok.

When the server has processed the request it sends the new marhsall string back to the client. The UUID is still used as prefix to identify the request.

The client finally sends the ok to close the connection to the server.

11 Components

Components are reusable packages of functions and classes. In Java the classes are put together in a jar file. These jar files can be imported in an other project and so be easily used. Jar files can be tested separately and be provided in a maven repository.

For this project a few of these components where created. For one they would have at some point been needed anyway and for the other reason to demonstrate the work flow with the build tools and repository.

11.1 Data structure

At the moment a specified data structure to store all the needed laboratory data already exists. It was implemented in vba and just consists of arrays, dictionary's and values. These structures exists in every programming language. Therefore it is easy to port the existing structure for example to java. But since Java is object oriented the laboratory objects can be modeled even better than in plain old vba. A sequence object can be created that in turn hold assay objects that also holds result objects. Beside the basic data holding some logic can be implemented or some properties can be extended and customized to special objects. A further benefit of having customized data objects (beans), is the fact that vaadin provides tables that can be filled whit java beans.

11.2 Marshalling

One key feature of porting the existing excel framework to activiti and java is the marshalling and unmarshalling of the data objects. Being able to marshall an object into an xml file and sending it over a socket provides a powerful function to communicate with other programs that might even be written in an other programming language.

The ability to marshall objects is already implemented in the existing excel framework. So only the matching java code has to be implemented. With the existing XSD definition it is possible to automatically create java objects with the xjc [29] command line tool. These objects can than be used with the JAXB [30] classes to easily navigate through the XML data and creating the java objects.

11.3 Compression

Before sending the xml to another socket over the network it is usually good to compress the data. In our case the standard inflation and deflation algorithm is used (zlib) [27]. After the compression the data bytes are encoded into base64 [28]. That allows a save transmission over the TCP/IP stack.

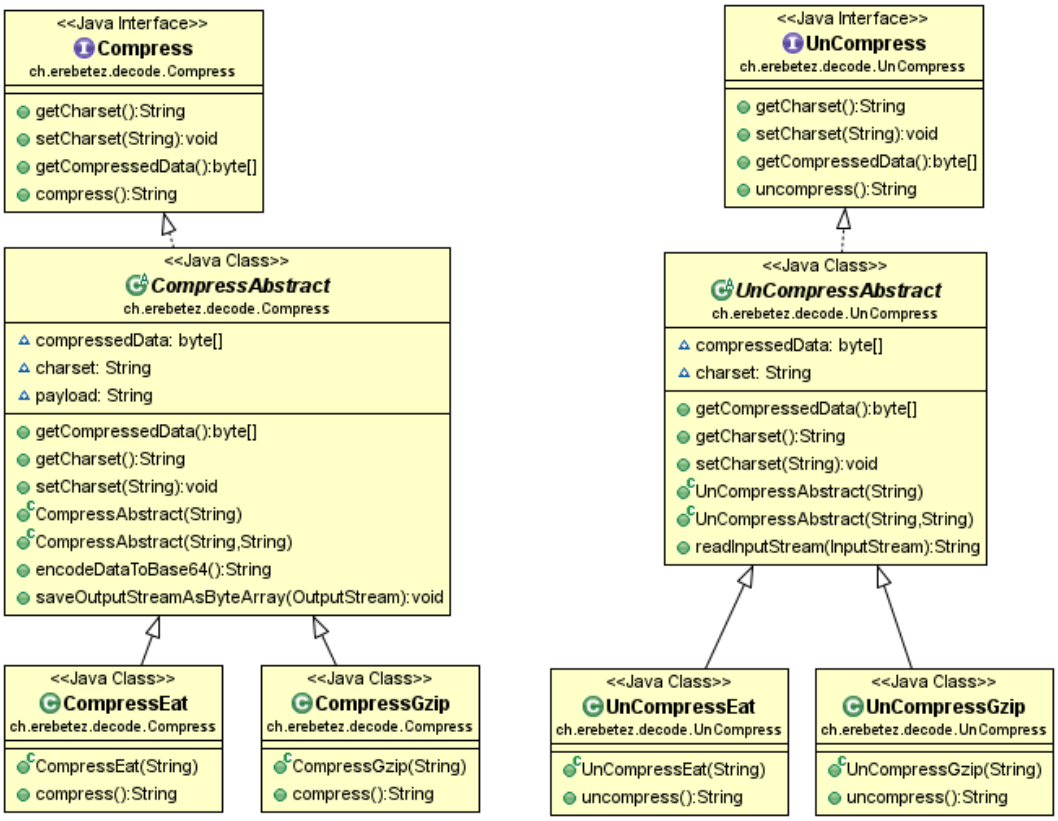

Abbildung 6: UML class structure of the compression classes

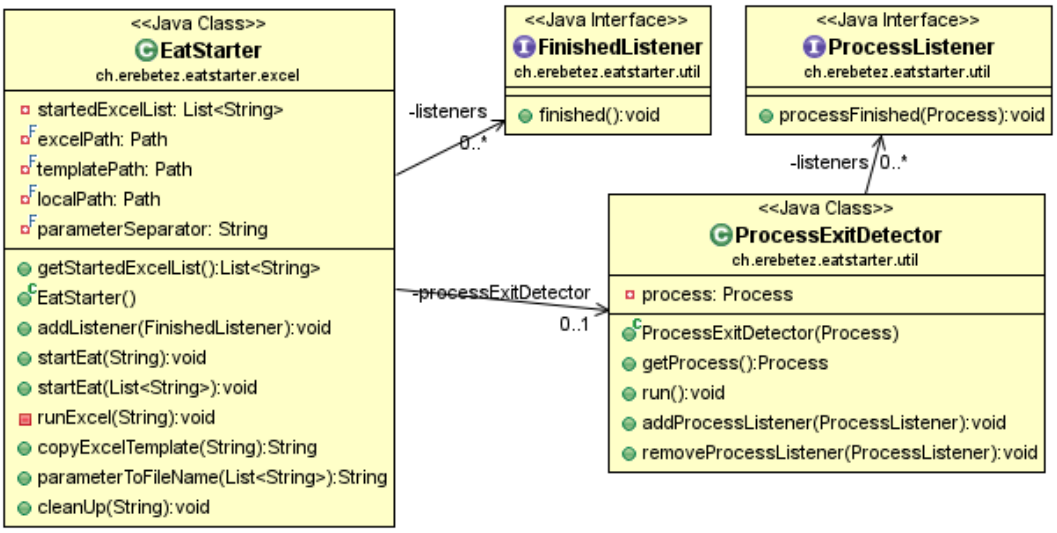

Abbildung 7: UML class structure of the excel starter

11.4 excel adapter

In order to use the existing application stack based on excel, it is important to be able to start excel files from java code. The excel adapter has two parts. One is the java package and the other is a special excel spreadsheet.

The java package takes some arguments starts the excel spreadsheet and gives a signal as soon as the excel process has finished.

The spreadsheet has an on open method, that starts the existing excel applications with the given parameter.

12 Environments

develop, validation, prod

13 Build tools

The question what build tool, if any, came pretty early in the project development. Just starting a Project in Eclipse i the easiest way. At least as long

as the project is not to big, is only one project and doesn't really change over time. That was all not the case. So lets have a look at few of them.

In the Java World there are the usual suspects:

- Ant
- Maven
- Gradle

13.1 Ant

Ant is the oldest of the three. It allows to define build scripts in xml. It has no own logic, so all tasks or targets have to be written from scratch. That allows a big flexibility, it is however also quit complex. There is also the fact, that scripting in xml is not really fun.

13.2 Maven

Maven also uses also xml for the configuration. Opposed to ant, you define how the project has to be build. The building is then made by Maven. Maven also resolves dependencies. Maven is widely used and therefore has a lot of plugins and additional tools like an eclipse plugin. For our case maven seemed to do the job.

13.3 Gradle

Gradle is the newest member in the Java build tool family. Gradle is not just a Java build tool so i can also handle other languages. The configuration file is written in groovy. That makes a much nicer looking project definition. Gradle also doesn't reinvent the world. It uses the same dependency resolvement as maven. So it can almost do everything that maven can. In addition, it is easy to create custom tasks like in ant.

The downside is, that gradle is still a very young project and has not as much plug-ins or examples as maven. It is however worth keeping in sight.

13.4 Usage

For the main web application it is the simplest way to use maven as the build tool. Since gradle seems very promising, the plan is to use it in the smaller sub projects. That way it can be easily evaluated and maybe used as the main build tool in the future.

14 Testing and Validation

Testing and Validation is an very important subject. It begins already with the developer and goes all the way to the qc personnel. There are several strategies to be followed.

14.1 Unit Tests

Unit tests are an important tool during and after the development of the code. If possible each function and each class should also have his unit test. Unit tests are low level tests that can be run automatically. By writing unit tests it also forces the developer to create modular and well designed classes. The best practice is, to write the unit test first with all the requirements and then implement functionality until all tests passes. This is of course repeated task that goes on forever. The big advantage of unit tests is now, that if a change in the code should brake something, at least one unit test will most likely fail. The developer then also know where to look and fix the problem right away.

14.2 Validation Tests

Once the all functionality is implemented the packages and applications are build and deployed to the validation environment. The validation environment is configured as close as possible the same as the productive environment.

Know the testers can execute the written tests plans to check the high level functionality. This is also a formal test to conform to the regulations. If a error occours the code gets modified in the development environment and redeployed. The tests are then repeated where needed.

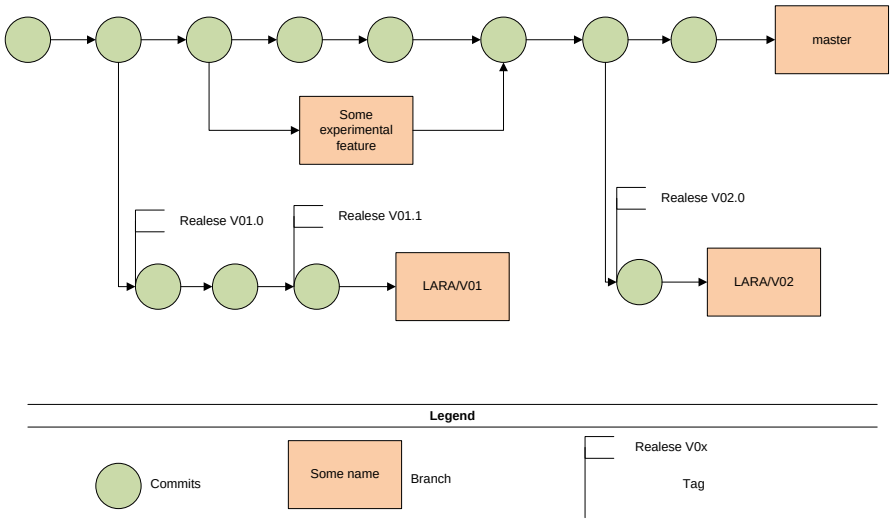

Abbildung 8: Concept of the git branches

15 Versioning

For the versioning of the source code there are several producs available. Just to name a few of them: SVN, mercurial, git. During the development of the example git was used. It is has a decentralised architecture. So every developer has always the full power of git at his disposal. It is also easy to create branches and to create labels for realises.

15.1 using git

In order that every user uses the branches the same way, a few ground rules have to be set. Each git repo will have at least three branches. productive, validation, and master.

Master is always the newest development branch. When the new feature is ready the changes are merged into the validation branch where they are tested. This validation branch gets the name of the new version. The only thing that has to be changed in the application configuration, is the database configura-

tion in the spring context files. The same has to be done when the productive package is about to be build. Since the configuration for the validation and productive environment can be in a separate file, the change is minimal. Only some imports in the spring context xml have to be changed.

Multiple git repository will be created. The following projects shall have a own repository.

- Web application
- External calculation Server
- The components

This way it is easier to manage branches and versions of each application.

16 Deployment

maven + repo + scripts + gradle

17 Infrastructure

This chapter shall give an overview how the infrastructure that supports the web applicatin and the development of it, has to look. All named server are planed to be virtual machines.

It would have four servers.

- Web application server
- Database server
- Calculation server (excel, r)
- Repository server

17.1 Webbapplication server

This server is a classic web application server that can run war Archives. So it runs an Jee environment like jBoss or tomcat. The operating system is therefore not so relevant.

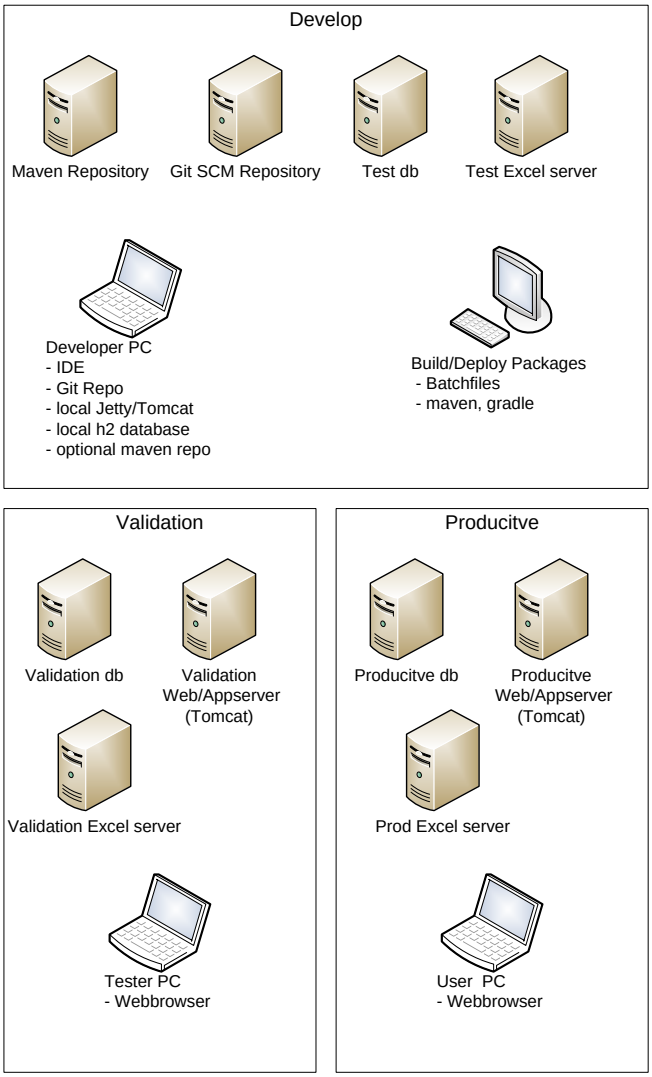

Abbildung 9: Infrastructure overview

To keep things simple linux was choosen for the example server.

17.1.1 configuration

17.2 Database server

De database server would be used to store the activiti data. The databases supported by activiti are mysql [21], oracle [23]. MSsql [24] is jet in experimental state.

For the example a linux server with a mysql database was setup. To manage the database phpmyadmin [22] is used.

Since the database is configured in the web application with spring, it is very easy to change the used database.

17.3 Calculation server

This is a rahter exotic configuration. This server should be able to run excel. Therefore windows is the only option al the operating system. On this server there will also run the calculation server (java) wich listens to the querrys of an activiti task.

17.4 Repository server

This server would not be used in the productiv setting. It is just there to facilitate the developers work. The server would provide a maven repository and a git repository.

The operating system would be preferably linux. That is because git is native to linux. For the maven repository nexus from (xxxx) would be a nice chois. It supports all neccesery functions and provides a nice web frontend for managing the repositoris.

18 Migration

Literatur

[1] http://en.wikipedia.org/wiki/Middleware

[2] http://www.omg.org/

[3] http://www.activiti.org/

[4] https://www.jboss.org/jbpm/

[5] http://www.activevos.com/

[6] http://www.activevos.com/content/blog/activevos2009butlergrouptechnicalaudit.pdf

[7] https://ode.apache.org/

[8] https://www.softwareag.com/ch/products/wm/bpm/default.asp

[9] http://gothamschools.org/2010/09/15/frustrated-with-citys-data-system-teachers-build-their-own/

[10] http://www.microsoft.com/biztalk

[11] http://www.microsoft.com/biztalk/en/us/pricing-licensing.aspx

[12] http://www.signavio.com/en.html

[13] http://www.eclipse.org

[14] http://en.wikipedia.org/wiki/Remote_Procedure_Call

[15] http://en.wikipedia.org/wiki/Universally_unique_identifier

[16] http://www.vaadin.com

[17] https://ant.apache.org/

[18] https://maven.apache.org/

[19] http://www.gradle.org/

[20] http://www.springsource.org/

[21] https://www.mysql.com/

[22] http://www.phpmyadmin.net/home_page/index.php

[23] http://www.oracle.com/de/products/database/index.html

[24] http://www.microsoft.com/sqlserver/en/us/default.aspx

[25] http://www.sonatype.org/nexus/

[26] http://git-scm.com/

[27] http://zlib.net/

[28] http://en.wikipedia.org/wiki/Base64

[29] http://docs.oracle.com/javase/6/docs/technotes/tools/share/xjc.html

[30] http://www.oracle.com/technetwork/articles/javase/index-140168.html