NTNU – Trondheim
Norwegian University of
Science and Technology

Department of
Computer and Information Science

TDT4290 Customer Driven Project

Development of Drilling Engineering Software

Authors:
Tomas Fagerbekk
Tintin Hoang
Pawan Chamling Rai
Tina Syversen

Customer Representatives:
Bjørn Brechan
Øyvind Pedersen
Sigbjørn Sangesland

Supervisor:
Soudabeh Khodambashi

January 13, 2014

Abstract

Petroleum activities have contributed significantly to economic growth in Norway, and to the financing of the Norwegian welfare state. Through over 40 years of operations, the industry has created values in excess of NOK 12 000 billion in current terms. In 2012, the petroleum sector accounted for 23 percent of value creation in the country [74]. However, oil companies in Norway are still using outdated software in their drill planning and operations.

The current market leading software is suite comprised of applications collected from different developers over the past years. The main issue with this solution is no intergration between the different applications and poor user friendliness. This causes major loss for the companies since they often have to outsource work to third parties that have higher competence in using this software. Furthermore, it introduces redundancy since the work done in one application have to be manually imported to the next.

Our solution is based on using combination of python and javascript to create a functioning prototype of a small part of the total system. We have chose to focus on the application that is responsible for the oil well planning process. One of our customers requirements was that the software should be able to run on all operating system, and also mobile devices like smart phones and tablets. Python is used to create a web interface which should be accessible on all units listed above. The framework we have worked on is called Django, and was selected because of its support for easily adding new modules to prototype in further development after the project's termination. Javascript was used to create 3D models of the wells by using open-source library three.js. This library showcase the capabilities of modern browsers and have extensive documentation.

During the development there were major changes in the requirements as a result of the demonstrations after each sprint. The main challenge was implementing a curvature formula that given two points would calculate the curve which would cause the least stress on the well. Under the project's duration we also came up with ideas for further development as a result of the experiences with our solution and the one currently in use, which is documented at the end of this report.

Preface

This report describe the complete process for the project "Development of Drilling Engineering Software" provided by Wellvis AS for the course TDT4290 Customer Driven Project held at NTNU. The project lasted from August 21st to November 21st 2013. The main purpose of this course was to give students a chance to experience how it is to carry out a large customer driven project.

We would like to thank our representatives from Wellvis AS: Bjørn Brechan, Øyvind Pedersen and Sigbjørn Sangesland for sharing their knowledge about the oil industry, their continuous feedback and overall being great customers. We could never have done this without your enthusiasm and support throughout the whole project.

We would also like to thank course responsible Babak A. Farshchian and all other members of the staff for arranging this course and supporting lectures. Last but not least, we appreciate the guidance and advice we were given by our supervisor Soudabeh Khodambashi.

Contents

List of Figures

List of Tables

Part I

Planning and Requirements

Chapter 1

Project Directive

This chapter will serve as a general introduction to our project. First we will explain the purpose of the project in section 1.1. We will then describe our customer in section 1.2. Afterwards, we will list the involved parties and what they hope to achieve in section 1.3. The project background is located in section 1.4, the objective in section 1.5 and the duration in section 1.6.

1.1 Project Mandate

The purpose of this project was to develop a more integrated and user friendly well drilling engineering system than those that are currently on the market. Because of the large complexity of developing a system of this scale, we were given permission to choose whichever part we felt the most interested in and develop a prototype with improvements suggested by the customer. We thereby reduced the scope of the project and decided to focus on the part of the system which was responsible of planning the well path. This is numbered as 1 (Well paths) on figure 1.1.

1.2 The Client

Wellvis is a company under establishment. It is founded and owned equally by Bjørn Brechan, Sigbjørn Sangesland and Øyvind Pedersen. The company has the same name as their main product, Wellvis, which is a concatenation of "Well Visualisation". This is an umbrella software that primarily modernize and improve the well integrity model available for the oil and gas industry. Well integrity is an area of focus in the industry as it reduces risk and avoids failures.

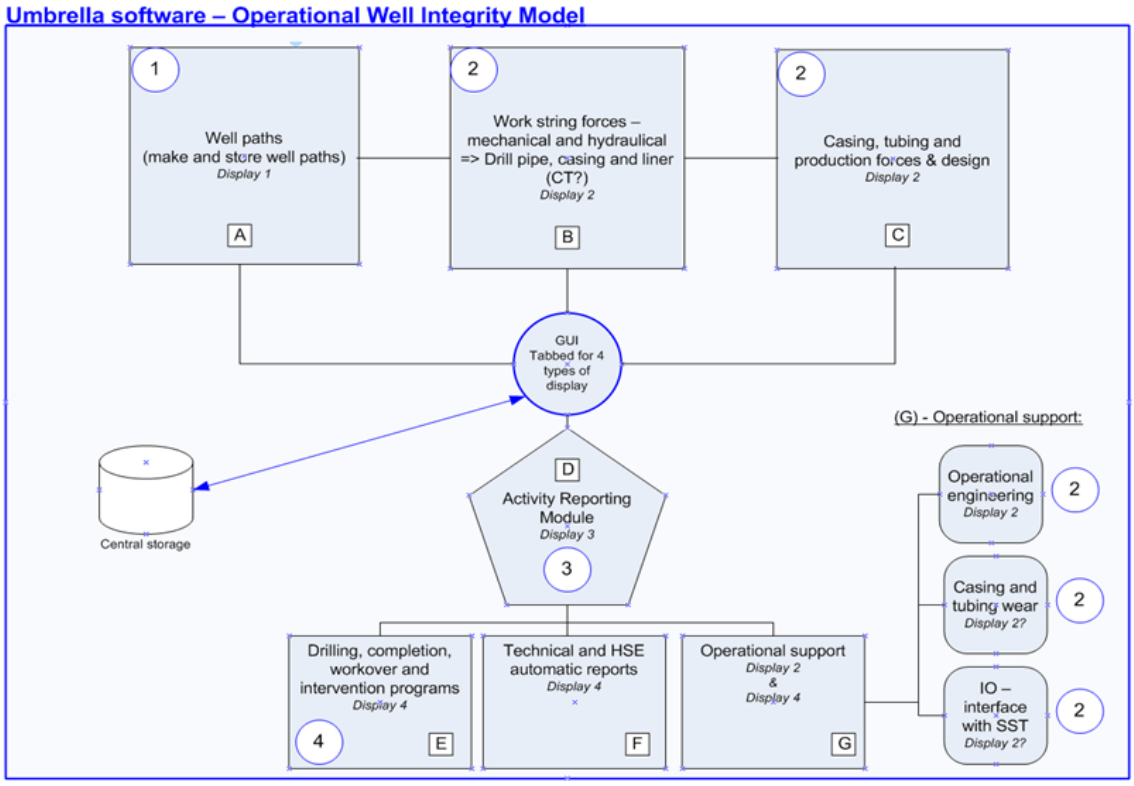


Figure 1.1: Operational Well Integrity Model

1.3 Involved Parties

These are the identified stakeholders who has an interest in our project.

Wellvis	They would like a new system that is more user friendly than the old and outdated system.
Project team	We would like to do a good job for our customer and get a good grade on our project
Course staff	Wants us to do a good job and want to know how to improve the course.
Users	Users of the current system want a more user friendly solution that is easy to use and learn

1.4 Project Background

Today the oil industry worldwide suffers from the lack of experienced engineers in all professions. One of the reasons for this is the work-intensive and demanding process of planning a well for production of hydrocarbons. The current workflow to establish a fully engineered well can be modernized and thereby reduce the amount of man hours needed. The existing well integrity models are complex and there is a high user threshold, resulting in several years of training before users are skilled and experienced enough to comfortably running it and making decisions based on the result.

A simplification of the work-flow can reduce required hours for engineering, but logical algorithms and integration could ensure that important tasks are not overseen, thereby increasing safety. With good visualisation of the status and remaining tasks, decisions can be made easier and sooner. This holds true for planning, but also during operations if they stray from the plan. Not only can the time required for proper engineering be reduced, but with lower threshold also the number of people involved.

At this moment there is only one complete software suite for the well drilling process. It consist of individual applications that have been bought and put together in a package. The problem is that these programs have been developed by different companies over a long time period. Consequently, the programs have none or little integration with each other and are very outdated in both graphics and programming paradigm. The programs are not very user friendly and require the user to have some tacit knowledge to operate them correctly. Because of a high user threshold, work is often delegated to external consultants. Lack of proper intergration between application also cause redundant work since data have to be manually transfered from one application to the next.

1.5 Project Directive

We have some objectives that we have set for ourselves and that the client wishes for:

- Create a working prototype of the Well Paths program with all the necessary functional requirements specified by the client
- Focus on the main areas of the client: that it is easy to use and learn
- Create a broad documentation of our work and the prototype
- Be able to present a final product

1.6 Project Duration

The duration of this project is three months, from the 21th of August until the 21th of November. For this course, each student is expected to work 325 hours. We are four people on this team so the maximum expected capacity will be 1300 hours.

Chapter 2

Requirements

This chapter describes our requirements for this project and how we have come to find them and use them. We start of in 2.1 where we list all of our requirements while how they have been evolutionized are in section 2.2. The end of this chapter is delegated to our use cases. In 2.3 we have both our diagrams and the textual descriptions for these.

2.1 List of requirements

This section is about our requirements and we discuss how they are prioritized in 2.1.1. Later we list all of our functional requirements in 2.1.2 both cancelled and implemented, sorted by when we got them from our customer.

2.1.1 Prioritization and Complexity

The team has prioritized the requirements in four categories:

- High: Core functionality of the utility that must be implemented.
- Medium: Requirements that will improve the value of the utility.
- Low: Requirements that will not add much value to the utility.
- Optional: Requirement that may be implemented depending on available time.

The team has estimated the complexity for each requirement. We use the following categories:

- High: Functionality that seems difficult or time consuming to create.
- Medium: Functionality that are somewhat time consuming, but quite straightforward.
- Low: Requirements that are trivial to implement.

2.1.2 Functional Requirements

ID	Description	Prioritiziation	Complexity
F1	Login to the web page	High	Medium
F2	Select software module in web page	High	Low
F3	See a list of existing well projects	High	Low
F4	Choosing a well project from list	High	Low
F5	Insert well path data through website	High	Medium
F6	Visualize well path data in a 3d graph	High	High
F7	Distinction between end- and help-targets	High	Medium
F8	Divide path into segments	Medium	High
F9	Select segment for a detailed view	Medium	High
F10	Make targets adjustable through GUI	Low	Medium
F11	Go-offline mode	High	High
F12	Insert geological data, along with no-go areas.	Medium	High
F13	Insert operational data, and show offset.	Medium	High
F14	Export coordinates function	High	Medium
F15	Import path data from other existing wells into 3D GUI	Medium	Medium
F16	Warning if system detect intersecting paths	Medium	High
F17	Configurable 3D view	High	High
F18	Automatic load and store of graph	Medium	High
F19	Create curvature for paths	High	High
F20	Version control of input	Medium	High
F21	Detachable well panel	Medium	Medium
F22	Fullscreen mode	Medium	Low

Table 2.1: Functional requirements

2.1.3 Quality Requirements

ID	Description	Prioritiziation	Complexity
M1	The module should be easy to integrate with other modules, and it should be easy to add more functionality. It should not take more than one working day to do this.	High	High
U2	The module should be intuitive and easy to understand and use. It should not take more than 3 hours to get through a tutorial or explain how to use it.	High	Medium

Table 2.2: Quality requirements

2.2 Requirements Evolution

The customer was very open minded to what kind of project this could develop into. During meetings in the pre-planning period, we were given a good understanding of the demand of a new software, by getting a first hand experience of the drawbacks. We were taken on a tour at the office area of a oil company and were put into the user's situation.

Due to the magnitude of the system, we had to choose specific parts and narrow down the scope for implementation. The general requirements ideas were made in collaboration with the customer representatives, but finalizing the details was mainly done internally in the team.

A summary of the requirements evolution and a graph illustrating the evolution is shown in 2.2.5

2.2.1 Pre-planning requirements

Prior to sprint 1, we had multiple meetings with the customer. During these meetings we came up the following possible requirements:

- F1: Login
- F2: Select module within page
- F3: List well projects
- F4: Choosing well
- F5: Insert well path
- F6: Visualize well path

- F7: Distinction to end point targets
- F8: Dividing path into segments
- F9: Segment selection
- F10: Adjustable targets
- F11: Offline mode
- F12: Insert geological data
- F13: Insert operational data and show offset
- M1: Modifiability
- U1: Usability

2.2.2 Sprint 1

Planned implementations

For sprint 1, we decided on the basic functionality. The software should be able to show a webpage where the user could log in and select an application, well and well path. This well path should then be displayed.

- F1: Login
- F2: Select module within page
- F3: List well projects
- F4: Choosing well
- F5: Insert well path
- F6: Visualize well path

New requirements

During sprint 1 as well with customer meeting after sprint 1, we came up with some new ideas:

- F14: Export coordinates function
- F15: Import path from other existing paths into GUI
- F16: Warning if other paths intersects with the current one
- F17: Configurable 3D view
- F18: Automatic store and load of graph
- F19: Create curvature for paths

Deliverance

All planned implementations were successfully implemented during sprint 1. We were lucky enough not to run into any problems that was not solved within a short time frame. The requirements were delivered just in time.

2.2.3 Sprint 2

Removed requirements

- F8: Dividing path into segments
  After seeing the usability of the zoom function, dividing path into segments seemed more obsolete, and was removed.
- F9: Detailed segment view
  Also removed because it was dependent on F8, which was removed.
- F10: Adjustable targets
  The realization that adjusting targets in a 3D space would not be very intuitive removed this requirement.
- F11: Go offline mode
  The workload associated with the requirement is too large to take on during this project. Instead we wanted to focus our efforts on making a good path graph.

Planned implementations

- F7: End target points

- F12: Insert geological data
- F15: Import existing paths into current
- F17: Configurable 3D view
- F18: Automatic load and store of graph
- F19: Path curvature

Cancelled requirements

Due to the underestimated complexity of implementing the high priority requirement F19: Path curvature, we had to cancel some lower priority requirements:

- F7: Distinction to end point targets
- F12: Insert geological data
- F15: Import existing paths into current
- F16: Warning if path intersects

New requirements

The following requirements emerged during sprint 2.

- F20: Version Control of path
  During the implementation of saving path to database, we realized how useful it would be with version control of well path compared to its complexity.
- F21: Detachable well panel
  Customer expressed this wish in the customer meeting after sprint 2.
- F22: Fullscreen mode
  Thought to be a visually beautiful feature with low complexity.

Deliverance

At the end of sprint 2, we had only delivered 3 out of the planned 7 requirements. We had underestimated the complexity of implementing the math functions involved in making an optimal

curvature, and the drawing of the points in 3D (F19). Also, struggling to implement storage of JSON-structures in the Django framework through AJAX-calls, caused further time to slip from our hands. In hindsight, estimations were closer to best case than average case scenarioes. This sprint turned out to be worse than a expected average case.

2.2.4 Sprint 3

Planned implementations

- F19: Create curvature for paths
- F20: Version Control of path
- F21: Detachable well panel
- F22: Fullscreen mode

Cancelled requirements

- F13
  Cancelled during pre-planning for sprint 3 due to time restrictions.
- F14
  Cancelled during pre-planning for sprint 3 due to time restrictions.
- F21: Detachable well panel
  Having two browser windows that communicates instantly is not possible with plain Javascript and non-persistant HTTP connections to the server. A workaround by polling the server every x second seemed like a suboptimal solution, and the proper solutions too technically heavy to implement. Thoughts on how to implement this requirement is presented in section 10

Deliverance

When we were finished with sprint 3 we had managed to deliver all of the requirements we were supposed to in this stage. We learned from sprint 2 and made sure to be able to deliver what we wanted. The curvature(F19) took a lot longer than we hoped, and got complicated by some bugs, so we prioritized this over the other two requirements that are simple to do. All of the requirements were delivered on time.

2.2.5 Summary

As a team, new ideas were emerging all during the project. Due to the loose bounderies on the project and agile method, we planned only one sprint at a time and allowed ourselves to drop certain requirements and focus on other ones in collaboration with the customer.

In hindsight, we realize that both sprint 1 and sprint 2 were estimated optimistically. This went fine in sprint 1, but came back to bite us once we ran into problems in sprint 2. We also noticed how not reaching goals can act as a motivational blowback, and found out how overestimating the thought time to some degree can get you closer to a realistic estimation.

Of the 22 requirements defined throughout the project, 12 were implemented and 10 removed. Some of the removed requirements are still good ideas for the software to-be, and we have therefore included thoughts and discussion on how they can be realized in chapter 10

For a visual presentation of the requirements evolution, see figure 2.1

2.3 Use Cases

This section is about our use cases. We start of with mentioning our actors in 2.3.1, and we continue with our diagrams in 2.3.2. In the end we have detailed textual use cases for these diagrams in section 2.3.3.

2.3.1 Actors

There are different kinds of users (actors) in a team: 2 - 4 drilling engineers and 2 – 4 completion engineers. There are two different access levels, one where everyone have access - read-only, and one where there is possible to edit. We have called these user and superuser respectively. A regular user has in other words a subset of a superuser's permissions.

Superuser:

- Company Representative (Advisor) – for all licences (fields). Could be a role like this per country
- Coordinator of the Directional Drilling service
- Drilling Engineer – super user
- Drilling Engineer

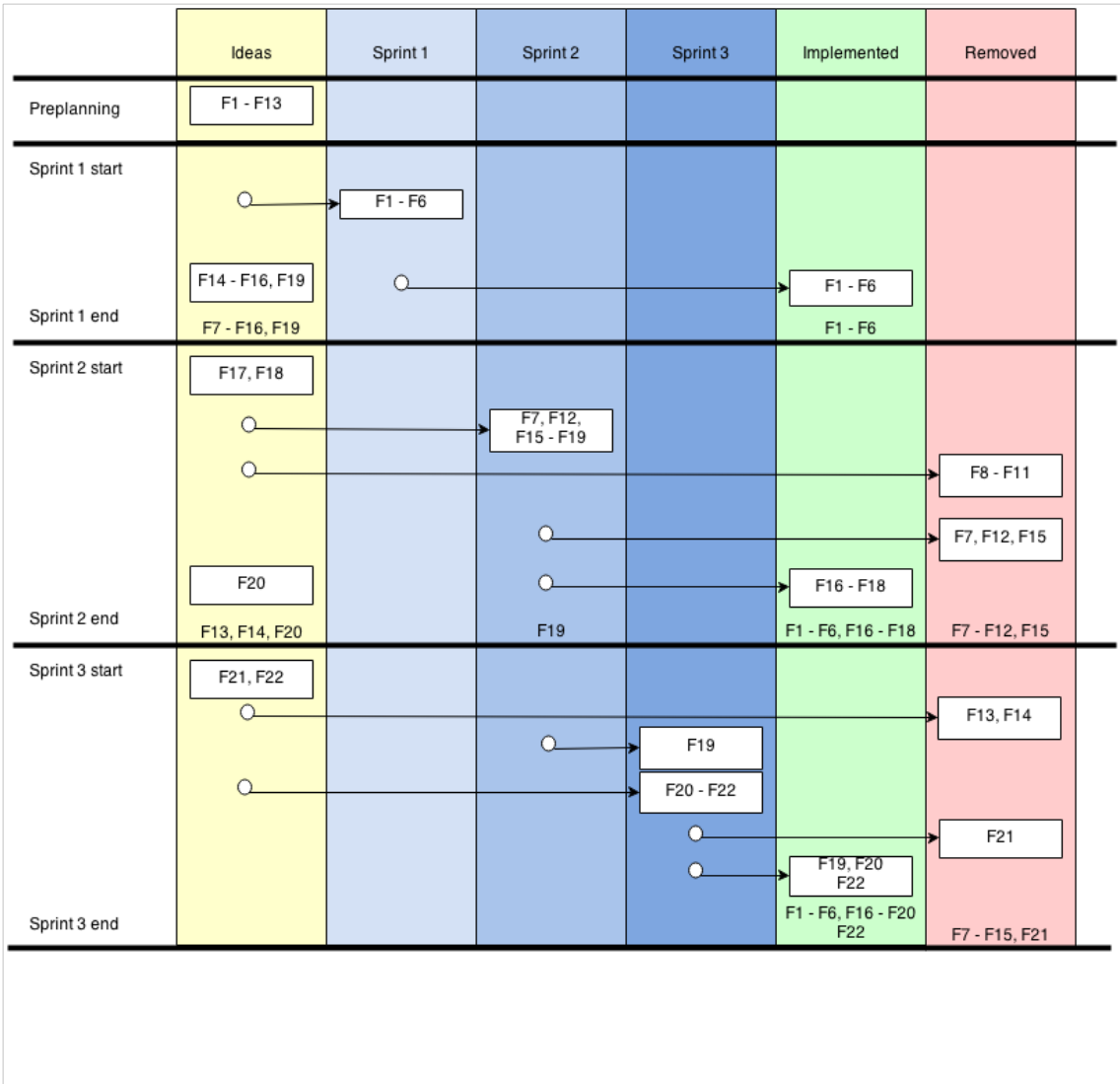


Figure 2.1: Requirements evolution. Requirements comes in to the Ideas column. From there they are moved to individual sprints, or decided not to be implemented (moved to removed). The bottom of each column within a sprint row tells what requirements they contained at the end of each sprint.

User:

- Completion Engineer
- Drilling Superintendent – read only access (normally) – not further described due to the low interface this role has towards the application(s)

We did not have the time to implement these two users, so in our system we only have one kind of user, and we are operating with that in the use case diagrams.

2.3.2 Use Case Diagrams

This section we will show our use case diagrams. We have only made use cases for the requirements that is done by a user.

Use case 1: Choose Well Path Project

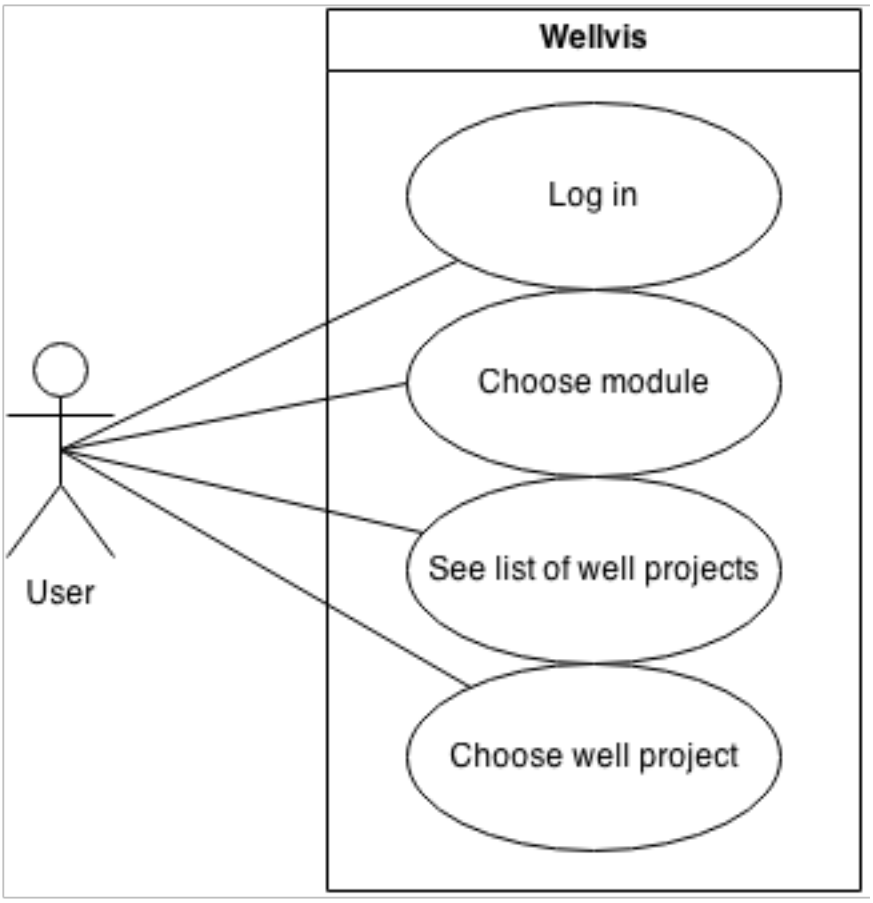


Figure 2.2: Use case 1

Use case 2: Well Path Data

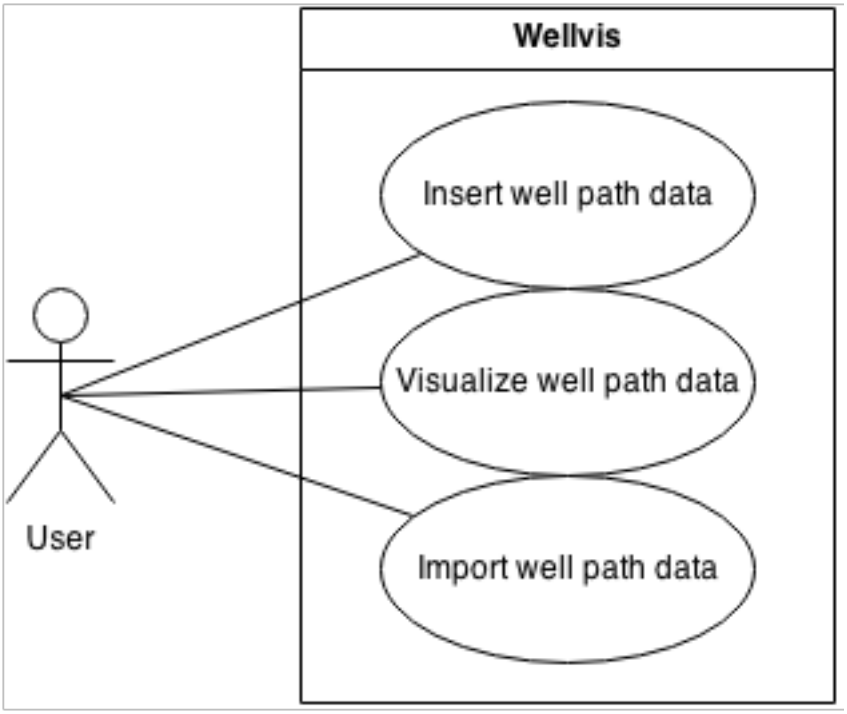


Figure 2.3: Use case 2

Use case 3: Choose Configurable

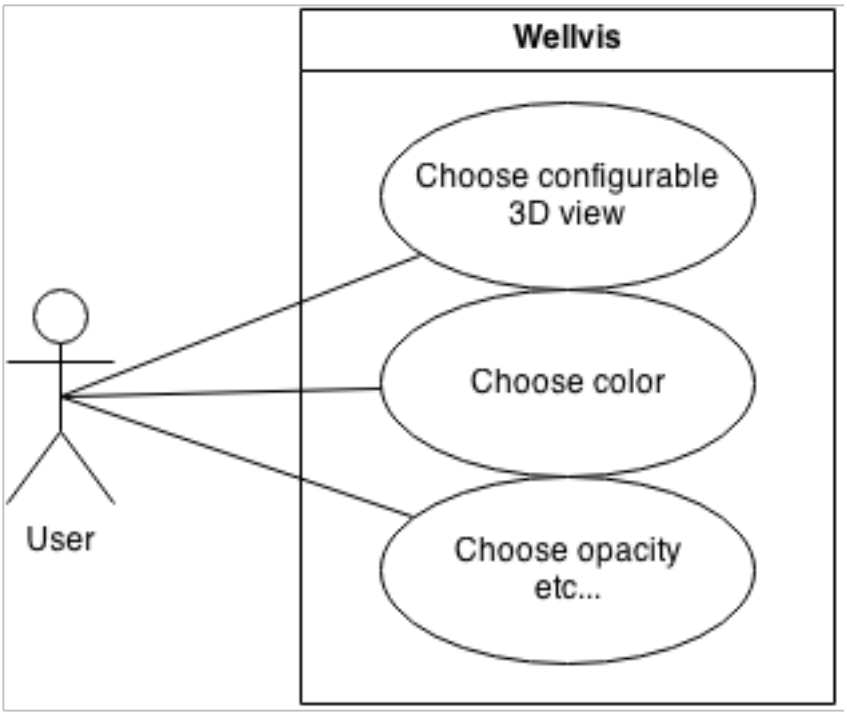


Figure 2.4: Use case 3

Use case 4: Choose Version Control

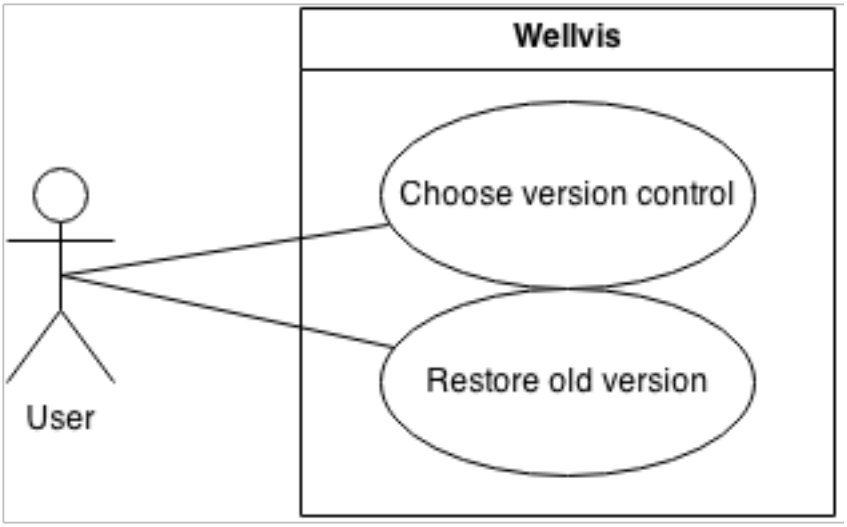


Figure 2.5: Use case 4

Use case 5: Choose Fullscreen

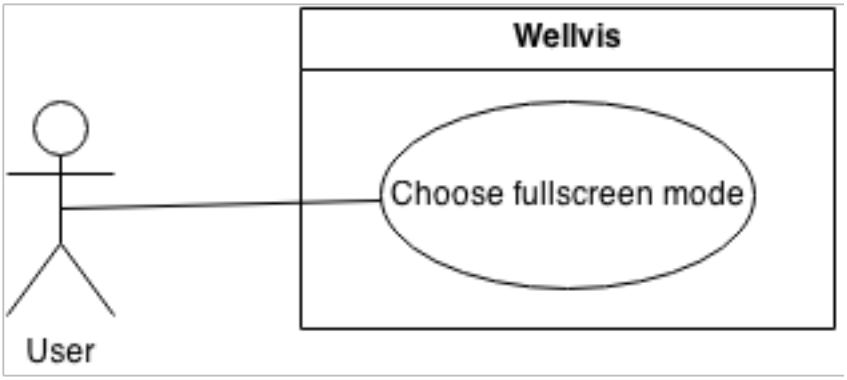


Figure 2.6: Use case 5

2.3.3 Textual Use Cases

This section describe textual use cases that comes from the use case diagrams in section 2.3.2.

Use Case 1: Choose Well Path Project

Purpose	So the user can choose a well project to see or work with
Precondition	Needs to have username and password
Step 1	Log in with username and password
Step 2	Choose the right module, Well path
Step 3	Choose to see a list of well projects
Step 4	Choose the right well project
Comments	

Table 2.3: Use case 1

Use case 2: Well Path Data

Purpose	So the user can insert, import or visualize well path data
Precondition	Needs to be logged in. There must also exist well path data to import for the import-part
Step 1	Insert well path data
Step 2	Import well path data
Step 3	Choose "Visualize well path"
Comments	Step 1 and 2 are interchangeable, and it is not necessary with both to visualize the well path. They can also be repeated several times and must be done by someone with permission to edit

Table 2.4: Use case 2

Use case 3: Choose Configurables

Purpose	So the user can choose colors and opacity themselves
Precondition	Needs to be logged in and be able to edit. Must have visualized the well path and imported well path data
Step 1	Choose Configurables
Step 2	Choose color
Step 3	Choose opacity
Comments	Step 2 and 3 are interchangeable and it is not necessary with both or any of them. Can be repeated several times.

Table 2.5: Use case 3

Use case 4: Choose Version Control

Purpose	So the user can go back and choose an earlier version of the well path
Precondition	Needs to be logged in and be able to edit. Must also have at least one previous version.
Step 1	Choose Version control
Step 2	Choose Version to restore
Comments	Step 2 is only necessary if they want to change version

Table 2.6: Use case 4

Use case 5: Choose Fullscreen

Purpose	So the user can choose to see in fullscreen
Precondition	Needs to be logged in and see a well path
Step 1	Choose Fullscreen
Comments	

Table 2.7: Use case 5

Chapter 3

Planning

This chapter we will go over the more administrative aspect of our project. We start off in 3.1 with our project plan and how we measure the success of our project. The project organization is laid out in 3.2. Afterwards, we have section 3.3 which contains how we assure quality and later in 3.4 we talk about the risks that can occur, and how to manage them.

3.1 Project Plan

This section is about how we have planned our project. Sub-section 3.1.1 discusses how we, and others, can measure the success of our project. Sub-section 3.1.2 talks about the limitations we have in our project, both technical and non-technical, while in 3.1.3 we have our schedule for when we should be finished with certain results.

3.1.1 Measurement of Project Effects

Customer

- *Functional prototype:* The customer wants a prototype with some of the most important functions to see if it is possible to make a better version of the current system.
- *Modifiability:* Since the project team is just developing a small part of the system it has to be well documented so the customer can expand with more functionality later. The modules should be easy to integrate.

- *Economy:* The system should also contribute to saving money by making it more user friendly and lowering the user threshold so they can do more of the work themselves instead of relying on specialists on the software. They are also hoping to increase productivity by making the system handle repetitive manual work.

- *Security*: The customer hope that by making the system more effective and user friendly less people skip or take "shortcuts" during the planning phase of the well drilling process and thereby reduce accidents or unforeseen circumstances during operation.

Course staff

- *Experience:* The goal with the project is to give the students practical experience in carrying all the phases in a larger customer driven IT-project. The focus is on group dynamics and customer negotiations.

Project team

- *Technology:* Besides learning how to work in a large project with real customers, the project team is also very interested in learning more about new technology.

3.1.2 Limitations

This section will describe the technical and non-technical limitations identified in the project. This are constraints that we have to workaround or find solutions to help mitigate their impact.

Technical Limitations

- *Authentic Test Data:* Data about oil wells are stored in a shared database that all oil companies are able to access. The goal of this is to keep an overview over where the wells are located so they can avoid drilling collision. Unfortunately we did not have access to this database.

Non-technical Limitations

- *Language Barrier:* While all team members are able make themselves understood in English, this is not the first language for any of us. Communication within the team and with the customer may therefore be slower and harder to interpret than usual. The sections in this report could also suffer where our intentions are not properly communicated to the reader.

- *Experience:* No team member have had any prior experience with 3D modeling, writing in LaTeX or working in the petroleum industry. A significant part of the project have to be delegated to learn and obtain the skills necessary.

- *Time Constraint:* The project lasts for a duration of 3 months. This deadline is a limitation we can not influence in any other way than to put in more hours if the progress is behind schedule.

3.1.3 Schedule of Results

21st August: Project start and first meeting with the customer
22nd August: Lecture "Presentation in agile software development and estimation techniques"
22nd August: Lecture "Presentation on ICT software architectures"
22nd August: Presentation on project management *5th August* Lecture "Group dynamics"
9th September - 30th September: Sprint 1
30th September 2013: First demonstration of prototype
1st October - 19th October: Sprint 2
19th October Second demonstration of prototype
8th October: Pre-delivery of report for advisor
14th October: Pre-delivery of report for examiner
14th October: Lecture "Technical writing course"
20th October - 10th November: Sprint 3
5th November: Lecture "How to sell in large application projects"
8th November: Lecture "Sales techniques"
20th November: Finalize report and make final presentation
21st November: Delivery of report and final presentation of product with customers and examinators

3.1.4 Concrete Work Plan

3.2 Project Organization

This section discuss how our project should be organized. In 3.2.1 we discuss the main responsibilities and roles that we have chosen and we also have an overview of the distribution of roles.

3.2.1 Project Organization

We have identified the main responsibilities and divided them into four different roles. The roles are responsible for making sure their area is under control, and being done. In other words, they are not expected to do everything within their area. The responsibility areas are as follows:

Task	Start Date	End Date	Est. Effort (hours)	Act. Effort (hours)
Pre-study	**21.08.13**	**08.09.13**	**255**	**238**
Research	21.08.13	01.09.13	50	8
Documentation	21.08.13	08.09.13	75	57
Implementation	N/A	N/A	0	0
Meeting	21.08.13	08.09.13	85	119
Lecture	21.08.13	08.09.13	36	54
Sprint 1	**09.09.13**	**30.09.13**	**313**	**333**
Research	09.09.13	30.09.13	40	61
Documentation	09.09.13	30.09.13	60	95
Implementation	09.09.13	30.09.13	118	125
Meeting	09.09.13	30.09.13	95	52
Lecture	N/A	N/A	0	0
Sprint 2	**01.10.13**	**19.10.13**	**313**	**265**
Research	01.10.13	19.10.13	36	23
Documentation	01.10.13	19.10.13	36	35
Implementation	01.10.13	19.10.13	142	157
Meeting	01.10.13	19.10.13	95	51
Lecture	N/A	N/A	0	12
Sprint 3	**20.10.13**	**10.11.13**	**313**	**247**
Research	N/A	N/A	10	6
Documentation	20.10.13	10.11.13	157	89
Implementation	20.10.13	10.11.13	42	65
Meeting	20.10.13	10.11.13	95	85
Lecture	N/A	N/A	10	4
Finishing report	**11.11.13**	**21.11.13**	**206**	**254**
Research	N/A	N/A	0	0
Documentation	11.11.13	21.11.13	126	132
Implementation	N/A	N/A	0	18
Meeting	11.11.13	21.11.13	80	104
Lecture	N/A	N/A	0	0
Total	**21.08.13**	**21.11.13**	**1400**	**1337**

Table 3.1: Concrete Work Plan

Advisor and customer contact

This person is responsible for maintaining contact with the advisor and customer. He/she will send the advisor our agenda before each meeting, and write a short resumé afterwards. He/she have a good overview of the customers needs and wishes and is our primary contact with regards to arranging a meeting place and time. He/she will also write an agenda before, and a short resumé after, the meeting with the customer.

Task manager

This person should have an overview over what tasks is defined, and making sure the appropriate information can be found in the project management tool. The purpose is to make it easier for others to pick up new tasks and do them with little overhead. The task manager should also have the tasks prioritized, and put in the designated sprints.

Project leader

This person have overview of all important deadlines. He/she is supposed to make sure every member of the team is contributing and is on time. He/she must have knowledge of our current progress and make sure we are on schedule.

Documentation manager

This person is responsible for that all information in the documentation is updated and conform with the current version of the product. He/she is also responsible that all necessary information is being documented. This does not mean that he/she have to write all documentation, just make sure it is getting done.

Scrum master

Scrum is facilitated by a Scrum master, who is accountable for removing impediments to the ability of the team to deliver the sprint goal/deliverables. The Scrum master is not the team leader, but acts as a buffer between the team and any distracting influences. The Scrum Master ensures that the Scrum process is used as intended. The Scrum master is the enforcer of the rules of Scrum, often chairs key meetings, and challenges the team to improve.

Branch master

The branch master is responsible for version control by merging different branches into the master branch. If an incompatible merging has occurred, he/she has the task to solve it.

Meeting leader

The meeting leader is responsible for that the agenda for the meeting is being followed and that everyone is able to express their thoughts or concerns.

Software architect

The software architect is responsible for making an architecture that satisfies both the functional and quality requirements for the software. This includes identifying architectural drivers and choosing appropriate patterns and tactics. He/she should be able to explain the rationale behind the design and make views that illustrate the structure and flow in the system.

Lead programmer

The lead programmer is responsible for overseeing the work being done by the other team members working on the implementation. A lead programmer will typically also act as a mentor for new or lower-level software developers or programmers, as well as for all the members on the development team.

Test manager

The test manager is responsible for making sure the tests are covering all the relevant aspects of the program. This includes both black-box and white-box testing of the software.

Role	Team Member
Advisor and Customer Contact	Tina Christin Syversen
Task Manager	Tomas Albertsen Fagerbekk
Project Leader	Pawan Chamling Rai
Documentation Manager	Tintin Trong Hoang
Scrum Master	Everyone
Branch Master	Everyone
Meeting Leader	Tina Christin Syversen
Software Architect	Tintin Trong Hoang
Lead Programmer	Pawan Chamling Rai
Test Manager	Tomas Albertsen Fagerbekk

Table 3.2: Role distribution

3.3 Quality Assurance

This chapter discuss how to ensure the quality. In 3.3.1 we mention routines for ensuring the quality internally while we in 3.3.2 discuss how we will approve a phase. In 3.3.3 and 3.3.4 we discuss our procedure for customer and advisor meetings. In 3.3.5 we talk about internal reports.

3.3.1 Routines for ensuring quality internally

To ensure that all phases of the project has the best quality possible we made sure to get another team member to double-check our work when we were finished. That way we could get someone to proof-read to reduce the risk of typos and get feedback about improvements. We also talked about what we were going to do today, what problems we may have faced and what we are going to do till next time during each scrum meeting to ensure that everyone had the same updated overview of the project progress.

Each team member was also assigned at least one responsibility area where they were supposed to have a good overview of and could report the current progress.

3.3.2 Phase Result Approval

We tried to make our progress very visible to the advisor and customer by demonstrating our results at the end of each sprint. They could then point out areas where they wanted an improvement and clear misunderstandings. If the customer was satisfied with the current result we could take another look at the requirements and agree on those we were going to work on next.

3.3.3 Procedure for Customer Meetings

After each sprint, the *customer contact* looked for a suitable time and place for the meeting with the customer. And also at times when when we found it necessary to discuss important topics. Minutes of meeting were written after each meeting.

3.3.4 Procedure for Advisor Meetings

To ensure effective and efficient meetings with the advisor, the agenda and current documentation was supplied to the advisor before 14:00 each friday. We had weekly scheduled meetings at 10:15, later 10:30, each monday. Minutes of meeting was written and supplied to both the advisor and customer on a weekly basis.

3.3.5 Internal Reports

A short summary was written each time the team members held an internal meeting or worked together. They also logged their hours with a short description of what area they had worked on.

3.4 Risk Management

ID	Risk	Description	Likelihood	Significance	Affected Area	Avoidance Plan	Mitigation Plan
R1	Sickness or no-show	Member is not available at critical time	High	Moderate	Deadlines and/or software functions	Stay healthy, good work-environment, give status updates	Delegate to another or work from home
R2	Member quits	Member leaves before project is finished	Low	High	Deadlines and/or sofware functions	Good work-environment, encourage members	Delegate to other member
R3	Arguments among team-members	Team memebers disagree	Moderate	High	Group, deadlines, software functions	Talk about decisions	Let them work with someone else
R4	Low motivation	Members do not feel that they are learning and doing enough and that makes them lose interest and work even less	Moderate	High	Group-environment, deadlines, software functions	Be encouraging, give them manageable tasks and responsibilities	Give other tasks or do a task together
R5	Underestimating project-size	The project and the assignments are bigger than expected	High	High	Deadlines and software functions	Divide an assignment into smaller parts and expect assignements to be harder than they might be	Divide assignment into smaller parts and do the easiest parts or save the part and see if there is time in the end
R6	Missing Skills	A member does not know how to do a task they have gotten	High	Moderate	Deadline and software function	Read relevant material before starting	Google and read up and get help from other member
R7	Lost Work	Some or all of the work has been lost	Low	High	Deadline and software functions	Remember to save/load/push	Start over and get help
R8	Customer does not involve themselves	The customer shows little interest and does not get as involved as they should	Moderate	Moderate	The outcome may not be as the customer expects	Have meetings after each sprint, keep contact, ask questions, keep them interested	Contact more or talk to the advisor
R9	Bad advisor	The advisor does not do their job which makes it harder to do the project	Low	Moderate	Result and deadlines	Weekly meetings and asking questions	Ask other advisors
R10	Wrong choice	Finding out that there has been made a wrong choice in the middle of the process	Moderate	Moderate	Deadline and software functions	Look at the drawbacks and check other possibilities	Start over
R11	Task not possible	A task that is given is not possible to do	Low	Moderate	Cannot give what customer wants	Do not promise anything before it is checked up, and divide task into smaller doable parts	Talk to customer and maybe change the task so it becomes possible

Table 3.4: Risk management

Chapter 4

Preliminary Study

This chapter present the preliminary study for the project. Since we had never done a project like this before there were a lot of things we needed to discuss. Naturally, majority of these discussions were on what we were to use and how to use it. We start with section 4.1 where we discuss similar solutions to our project. Then we discuss about different software development technologies that we can use in section 4.2. Later, in section 4.3 we discuss which tools we can use in our project, and in section 4.4 we discuss on different drawing tools that can be used.

We also needed to discuss on how to make sure we never edited in an old version, and how we can ensure this is mentioned in section 4.5 where we discuss different ways to do version control. Then we are over to the most important thing when creating a software, and that is choosing the programming language. In section 4.6 we discuss different popular programming languages and how they work. Parsers and libraries is also important and are then discussed in section 4.7. There are a lot of frameworks out there, and the web-frameworks are mentioned in section 4.8 while we have unit testing frameworks in section 4.9. User documentation tools is mentioned as well in section 4.10 and we have different integrated development environments in section 4.11.

In the end of this chapter, 4.12, we evaluate on everything mentioned in this chapter and conclude with what is best suited for us and for our project. We also discuss intellectual property rights and license for what we have chosen in 4.13.

4.1 Similar solutions

One of the main motivational factors for the customer to create a software like Wellvis was the lack of proper software solutions that fulfilled the requirements needed in the industry. During our brief research into similar solutions, we only found one potential competitor, TechDrill.

4.1.1 TechDrill

TechDrill [102] is all-in-one integrated drilling software. It has four different parts namely DSP-One, DDR-One, Octopus and AFE-One that are concentrated on well planning & drilling operations, daily reporting, collaborative field development planning and automated calculations respectively. We attempted, but failed in retrieving a demo or test software from this company. We strongly recommend the customer to investigate further into this software, before making large decisions regarding the Wellvis project.

4.2 Software Development Methodology

In this section we will describe about two of the most popular software development methodologies. In subsection 4.2.1 we will discuss the waterfall model and in sub-section 4.2.2 we will look at scrum.

4.2.1 Waterfall

Waterfall model has been the most popular software development since 1970 when it was first defined. It is a sequential model where the output of each phase acts as the input for the next phase, allowing the process to move downstream like waterfall hence called waterfall model. The different phases, in the sequential order, are as follows

Requirement Specification

It involves gathering the information about what the customer needs and defining, as clearly as possible, the problem the solution is expected to solve. In addition it also includes the non-functional requirements.

Design

The requirements gathered in the requirement specification phase are evaluated and a proper implementation strategy is formulated according to the software environment. The goal here is to define the hardware and software architecture that would dictate most to the implementation phase. The design phase is further categorized into two sections, i.e. system design and component design. The system design contains details and specifications of the whole system and explains how each component of the system will interact with each other. The component design contains specifications as to how each component will work separately and how results from one component will travel to another. For designing different software components different modeling languages/diagrams are used.

Implementation

Implementation phase is the time to actually start creating the components. Actual working parts of the system are created in this phase by applying the information gathered in the first two phases. The design generated in the design phase is converted into machine language that the computers can actually understand and process.

Integration

In this phase the different components from the Implementation phase are integrated to form a complete solution.

Testing

In this phase the software is checked for any errors or discrepancies. It actually starts right after the end of the implementation phase when the coding part is completed. Various different tools, software and strategies are used for testing the solution in order to make sure that it is error free.

Deployment

After the software is tested, the software is prepared for the installation/deployment. If the customer accepts the resulting software the product is installed into the customers' system. This is the phase where the customer can either accept or reject the resulting software.

Maintenance

No software is completely bug free so a certain amount of time is required for the maintenance. Furthermore, with the passage of time, customers' requirements will also change and modification or additions will also be required. So maintenance is an ongoing process which may stretch from a month to several months or even more.

4.2.2 Scrum

Scrum is a agile project management framework which is often used when the requirements are developed incrementally and exist in a rapidly changing enviroment. The solution is to deliver functional software at an early stage to the customer and evolve them as new requirements appear.

Scrum consists of three main phases. The first is an outline planning phase where you establish the general objectives for the project and design the software architecture. This is followed by a series of sprint cycles, where each cycle develops an increment of the system. Finally, the project closure phase wraps up the project, completes required documentation (such as system help frames and user manuals), and assesses the lessons learned from the project.

The innovative feature of Scrum is the sprint cycles in the central phase. A sprint is a planning unit in which the work to be done is assessed, features are selected for development and the software is implemented. At the end of a sprint, the completed functionality is delivered to stakeholders.

The main idea is to empower the team and let them make decisions as a group. Scrum deliberatly avoid using the term 'project manager'. Instead a 'Scrum master' is used to co-ordinate the development and have the responsibility to arrange daily meetings, track the backlog of work to be done, record decisions, measure progress against the backlog, and communicate with customers and management outside the team.

The whole team attends daily meetings, which are somethimes 'stand-up' meetings to keep them short and focused. During the meeting, all team members share information, describe their progress, problems that have arisen, and their plan for the following day. This means that everyone on the team have a good overview of the current situation, and if problems arise, can replan short-term work to solve them. [97, p. 72-74]

4.3 Project task management tools

Managing a software project involves many different aspects. We break down the requirements to smaller and more manageable tasks, both to track progress and to make the work easier to do. A method of doing this, is making a Work Breakdown Structure (WBS)[8002] that is a tree of the tasks that shows a hierarchical overview over the project[8001], and defines its scope. After creating a WBS, the tasks are then further specified with a time estimation and previous tasks it requires before it can be started, often in a project management tool. Having these, a Gant diagram[8003] is often made to find timing and ordering of the different tasks.

The purpose of doing project planning is manyfold. It makes it easier to know what to do, and that the requirements are covered. It simplifies task delegation and it provides the ability to say that we're on, or behind schedule.

In this section we will discuss popular project planning tools. We will evaluate the different candidates for project management tool for tasks, while in 4.3.2 we talk about other project tools for creating Gant diagrams, and WBS diagram.

We found our candidates by searching for (scrum) project management tools. We noticed that the market for this software hasn't completely matured yet, and felt that the alternatives were almost too many, without clearly better candidates. The degree of payment methods, methods, and maturity were wide.

Since this project is only over a few months, we wanted something that would be easy to learn, simple to use and get a status overview, and preferably free. Some functionality to be able to include it in our report was also wanted.

We came up with the requirements that follows.

4.3.1 Requirements

- Free
- Group board for at least four users
- Task can be assigned a status
- Task can be assigned a priority
- Task can be assigned to a sprint / period
- Task can be assigned to user
- Task can have attached files
- Task can be assigned a weight or time estimation
- Task can be assigned to a user story
- Task can have acceptance critera
- Can generate report to show burndown
- Can export what has been done

4.3.2 Trello

Trello [111] is a free and very simple task manager. You can create as many boards as you like, share them, and also add as many lists as you like in the boards.

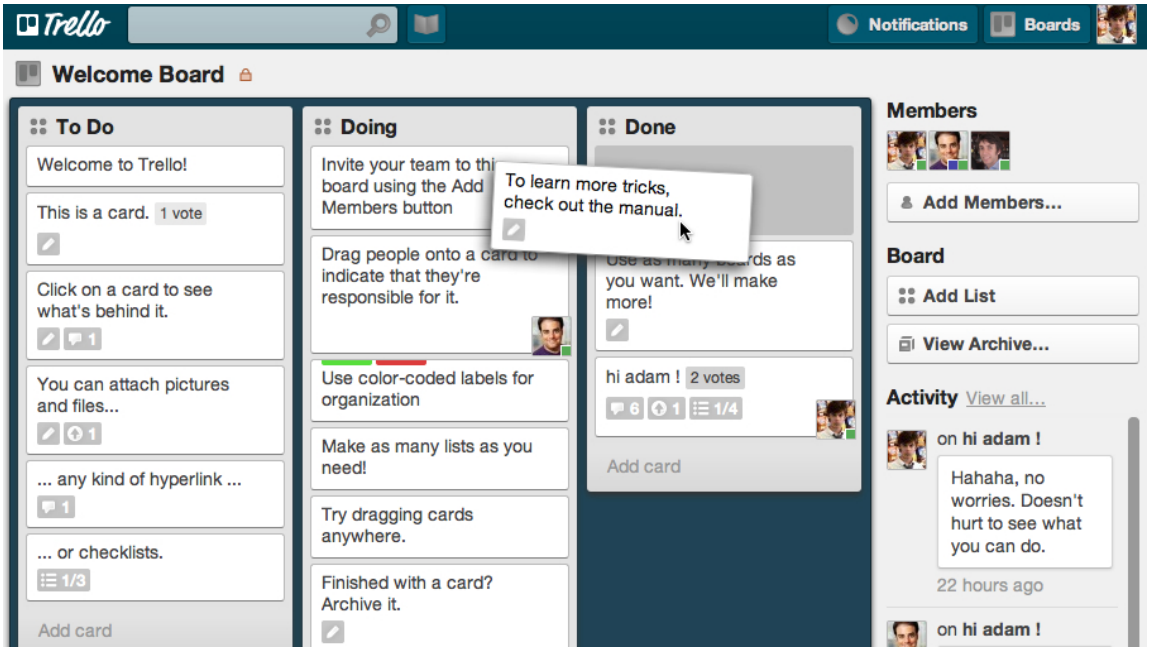


Figure 4.1: Trello is an easy to use task management tool.

Advantages

- Very easy and simple to use

Drawbacks

- Does not have time estimation
- Does not have a sprint or project overview
- Does not have any report tools
- Can not have subtasks (assign tasks to user stories)
- Does not support priority (other than moving tasks up on list)

4.3.3 EasyBacklog

EasyBacklog [39] fulfill most of the requirements. You can't directly assign a task to a user, but you can assign a color to it. It also can't attach files to tasks. On the other hand it has extra functionality such as creating a snapshot and comparing them. It can export the backlog to an excel file. It is easy to use and learn and is fitting for a small development team.

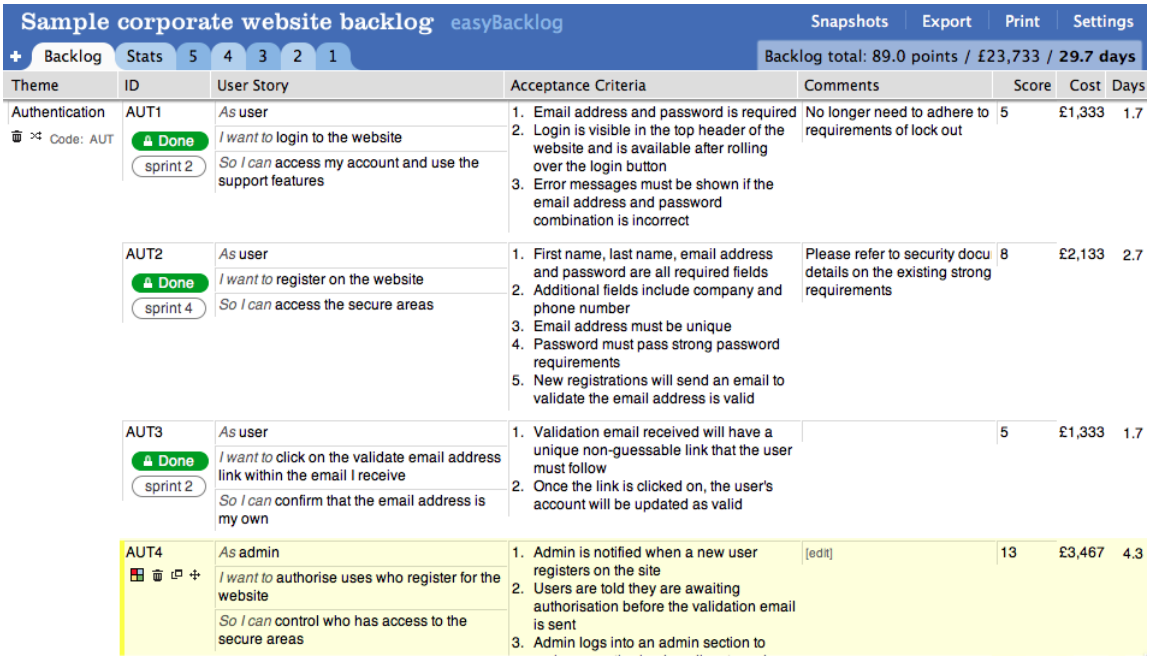


Figure 4.2: EasyBacklog's backlog is rich in information, but lacks ability to assign task to a person

Advantages

- Can export backlog to excel file for reporting
- Can create a snapshot and compare between different statuses
- Easy to learn

Drawbacks

- Cannot assign task to user (assign color)

- Cannot assign tasks to stories
- Cannot add files

4.3.4 RallyDev

Rallydev [95] is free to use for projects less than 10 people. It's a highly customizable tool, and has more functions than we need. Luckily, the views are simple enough to get a decent overview for the project. It satisfies all our requirements, though the reporting tool available in the free version is a bit weak.

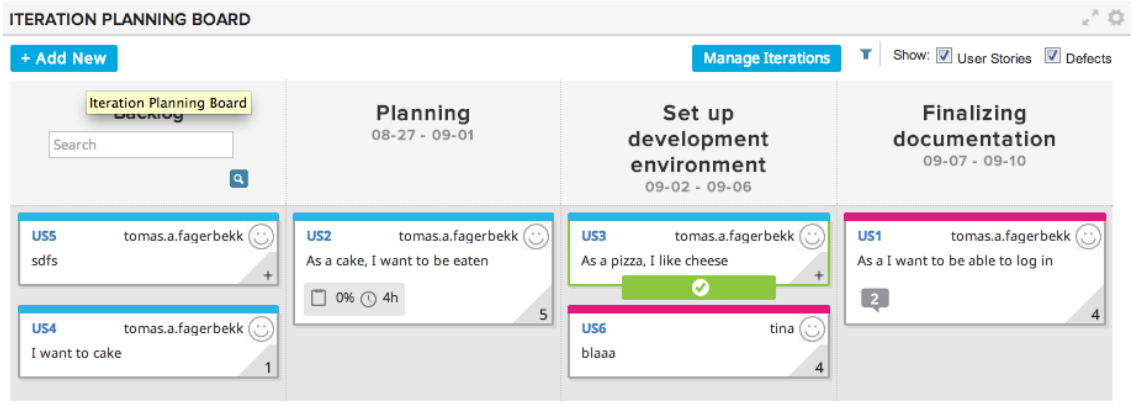


Figure 4.3: Rallydev provides a lot of information in an intuitive and user friendly way. Unfortunately the reporting tool is not available in the free version of the software.

Advantages

- User stories are easily moved between statuses and sprints
- The status board quickly lets you know the time estimate, status, assigned person and whether it's blocked or discussed.
- Quick edit of multiple stories in the backlog
- Dashboard shows stories ready for acceptance and also blocks

Drawbacks

- Not a good overview over individual tasks within user stories

4.3.5 Pivotal Tracker

Pivotaltracker [110] is free for public projects. It has nice reviews, and satisfies all the requirements we have set up. The use is simple and intuitive, and it has good reporting tools for showing what has been done within a given period. Epics show the progress on individual sprints in a nice way. Estimated hours, assigned person and status of individual tasks can be seen in the overview panel.

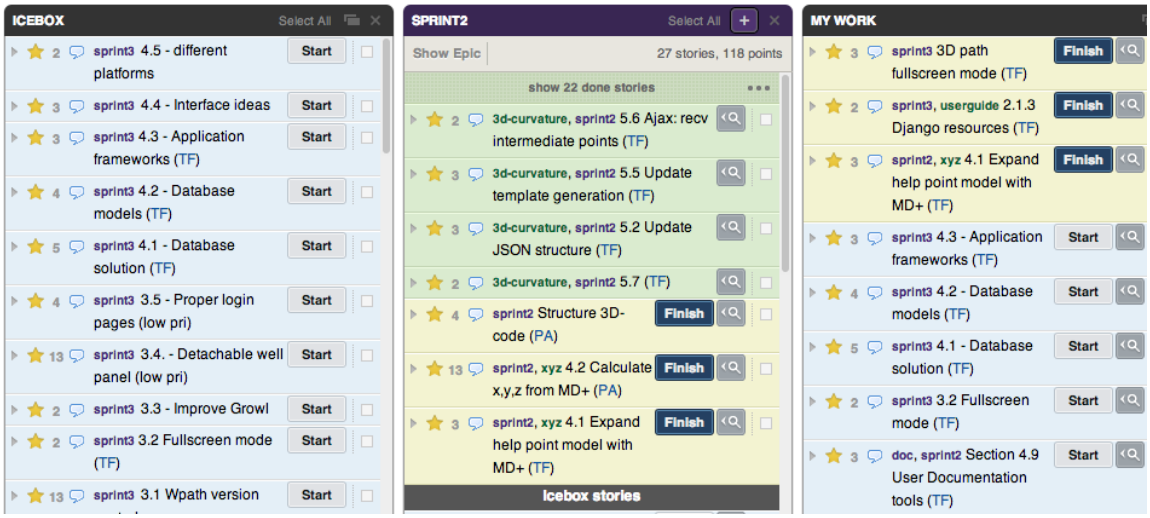


Figure 4.4: Pivotal Tracker allows integration with github, and has decent reporting tools

Advantages

- The status boards are easy to understand and use.
- Easy overview over sprint status
- Reporting tools are the best we've seen.
- Possibilities for integration with github

Drawbacks

- Requires project to be public in order to be free.
- Not a good overview over individual tasks within user stories

4.4 Illustrative Drawing Tools

In a software development projects, drawing tools are commonly used to illustrate the planned system through diagrams, models and graphs. These can be ER-models for databases, UML-diagrams (class-diagrams and float charts) to describe how a system works, mockups or wireframing to draw prototype interfaces. While most drawing tools can do this with enough time and effort, we wanted to use as few tools as possible. It should be easy to complete these tasks, and not cost much to get access to all functionality. The differences are summarized in table 4.4.4.

4.4.1 Microsoft Visio

We chose to first look at Microsoft Visio [120]. With a price range starting at 3 099 kr, it is not really a viable option, but it does provide us with a good standard to measure against. Microsoft Visio was introduced in 1992 [134], and is well established with great reviews [89] [101]. Figure 4.5 shows an example chart from the program.

4.4.2 LucidChart

LucidChart [67] was referenced in a lot of articles and user comments as the best free alternative to Visio [103]. Our impression from their demo page gave us the impression that this was a good all-round software that would satisfy our needs. It also have the possiblity to collaborate with others in real time. Unfortunately, the product has become a pay-to-use software ranging from $5 to $25 per month.

4.4.3 Balsamiq

Balsamiq [7] is a wireframing tool to mockup prototypes. It is profesionally made, and has a much richer variety of figures for interface mockups than draw.io. However, the scope for this software is noticeably smaller. It is clearly not meant to be used for other areas than for interface mockups. The desktop version costs $79, while the online version has a monthly fee.

Figure 4.6 shows an example chart in the program.

Name	UML	ER	Mockup	Price
Visio	Yes (great)	Yes (great)	Some	3 099 kr
LucidChart	Very (great)	Very good	Yes	9.95 & / month
Balsamiq	No	No	Very good	79 &
draw.io	Yes	Some	Some	Free

Table 4.1: Summary of diagram type support and price for different drawing tools

4.4.4 draw.io

draw.io [37] is a free web application that deliveres an intuitive drawing tool. It has a rich span of different figures that makes it eligible for making class diagrams, flow charts, use-case diagrams, UML diagrams, ER-diagrams and interface mockups. The user interface is very simplistic, but at the same time has functionality for integrating with Atlassian Wiki, Google Drive and Google Apps for business. Figure 4.7 shows an example chart in the program.

4.5 Version Control

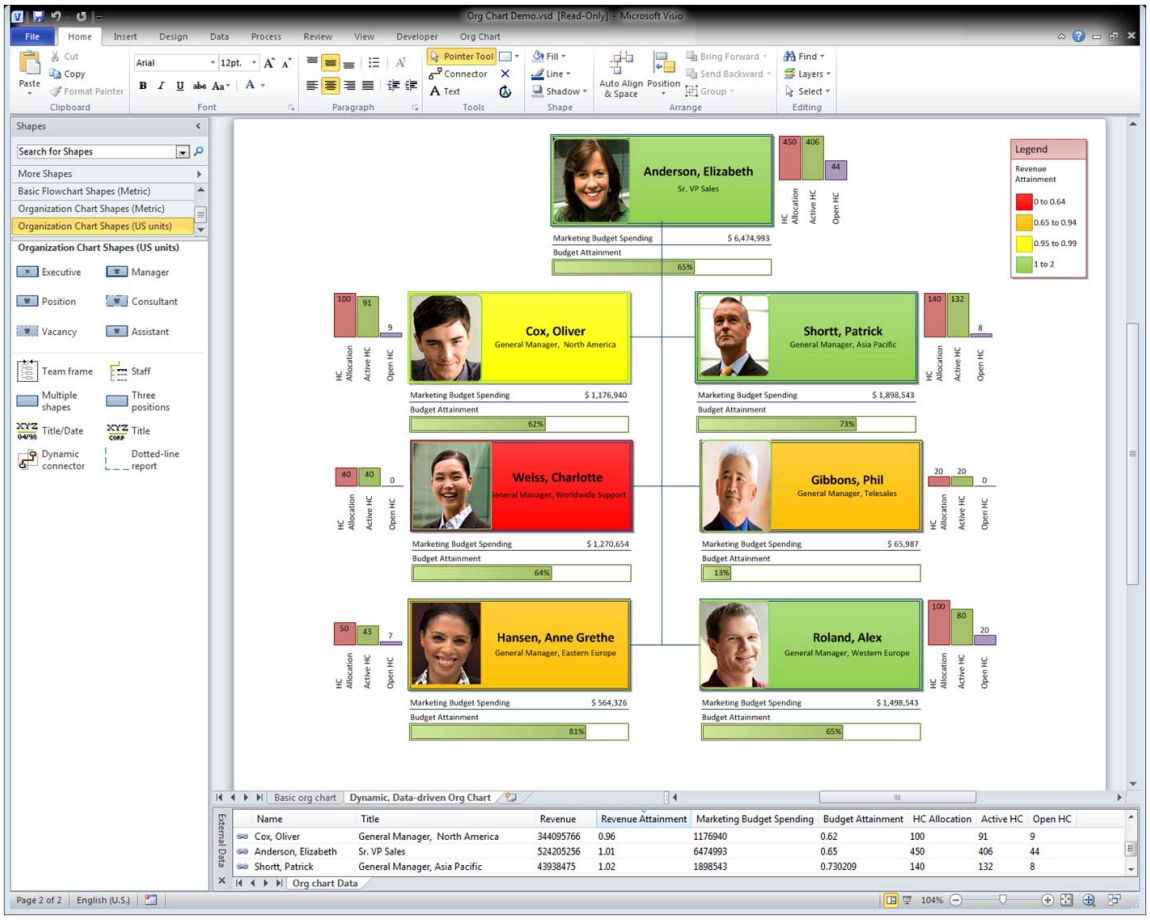
Version control systems (VCS) are software that records changes to a set of files over time. This allows you to both see the speed of the project, see who made what changes and restore specific versions at a later point. Compared to copying files into a backup directory (which is a common alternative to using a VCS), version control systems are faster, less space consuming and easier to handle, and makes it simpler to ensure consistency over a number of different computers [47].

One of the main arguments for using VCS comes up when one collaborates in a distributed team. Say Anna pulls down files A to her computer and changes them to A1. While Anna changes these files, Bob downloads the same files A, and changes them to A2. If Anna now updates the server with A1, we have a risk that Bob will override these changes when uploading A2 to the server. VCS prevents this by forcing Bob to incorporate Annas changes A1 into his files before being allowed to upload them to the server.

This chapter discuss the three most popular VCS. Section 4.5.1, we will look at an older and more traditional VCS, SVN. We will look at the currently most used one, Git, in section 4.5.2, before we look at a comparable alternative, Mercurial, in section 4.5.3.

4.5.1 SVN

SVN or Apache Subversion [1] which is its full name, is a centralized version control system initially released in 2000 [130] based on Concurrent Versions System (1990) [132]. SVN saves space by saving deltas (changes) between versions instead of whole files. However, different versions and *branches* of the repository is stored as files and folders, making a SVN tree potentially messy.



Name	Title	Revenue	Revenue Attainment	Marketing Budget Spending	Budget Attainment	HC Allocation	Active HC	Open HC
Cox, Oliver	General Manager, North America	344095766	0.96	1176940	0.62	100	91	9
Anderson, Elizabeth	Sr. VP Sales	524205256	1.01	6474993	0.65	450	406	44
Shortt, Patrick	General Manager, Asia Pacific	43938475	1.02	1898543	0.730209	140	132	8

Figure 4.5: Microsoft Visio can be applied to many types of charts and diagrams. Above an example for organization chart.

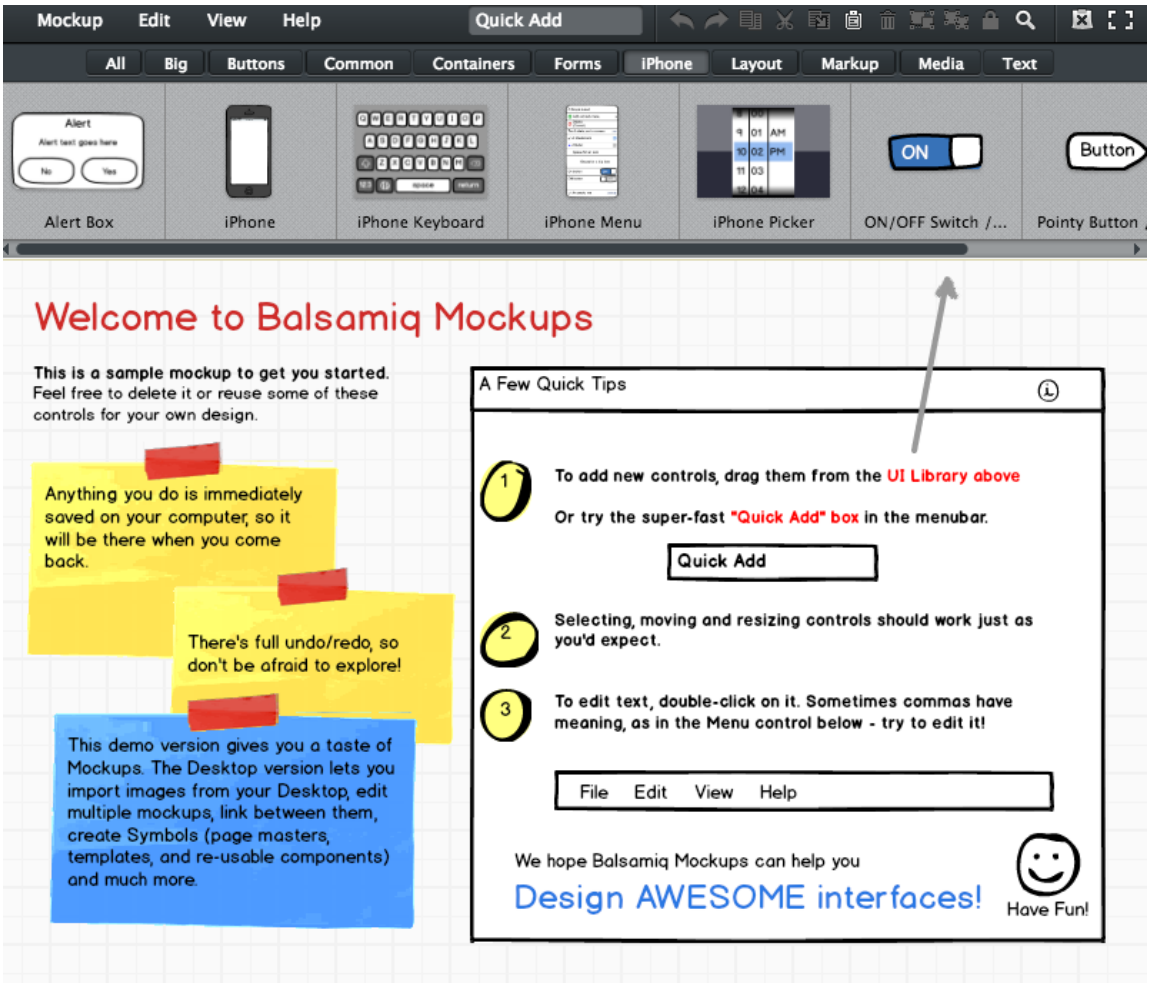


Figure 4.6: Balsamiq is a great program for making interface mockups. Unfortunately, it lacks good figures for drawing other types of diagrams.

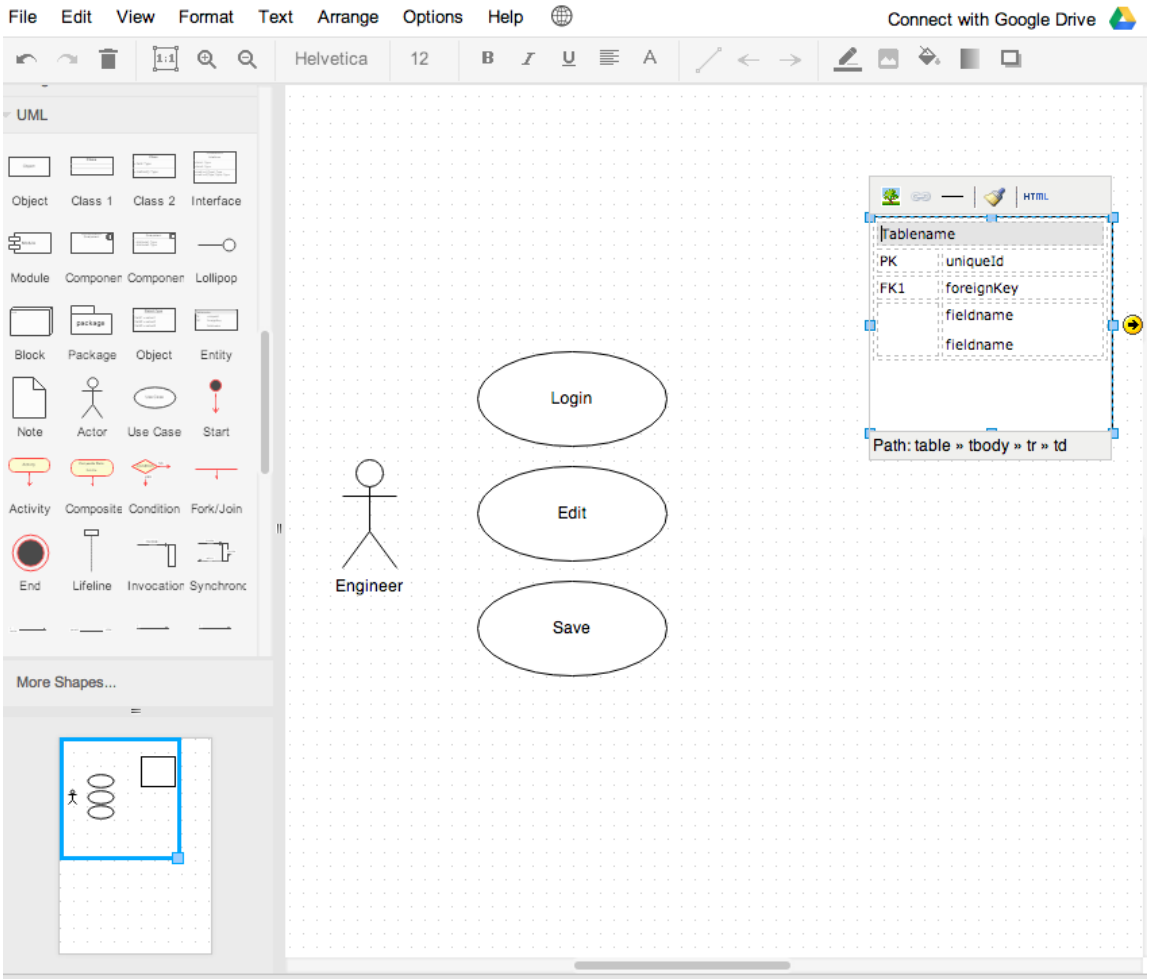


Figure 4.7: The free online drawing tool draw.io [37] does a great job competing against the pay-to-use solutions

SVN key points

- Full revision history of files
- Centralized
- Reduced space usage
- One point of failure (server)
- Messy file structure

4.5.2 Git

Developed by Linus Torvalds, the distributed version control system Git [52] has become a rising star after its release in 2005 [50]. Compared to SVN, it is more complex and can take a little longer to get comfortable with. However, the web interface tool Github [54] has both guides and community that should help you getting started. The advantages compared to SVN are several. Not only is Git faster and significantly smaller in size, but each copy is a redundant standalone with complete history making it much less vulnerable to data loss [51]. In addition, git has a branching feature that makes it easy to implement continuous integration [43]

Github also speaks in favour of Git, by providing *service hooks* to update different project management tools or programs [49], as well as a great visual navigation of a repository.

Git key points

- Complete version control
- Distributed
- Quick
- Very efficient space usage
- Integration with a range of project management tools
- Great complementing tools (Github)
- Flexible

4.5.3 Mercurial

Like Git, Mercurial (2005) is a distributed version control system with major improvements over SVN [92]. Where Git is purely written in C, Mercurial is implemented in Python with a C extension for the heavy lifting [105]. This has a negative effect on performance, though quite small. The main difference compared to git is its different philosophy when it comes to history. Mercury philosophy is that "History is permanent and sacred" and as such only allow you to undo the last pull or commit [5]. Git on the other hand, allows you to do `anything` to the history, including changing history as well as reverting to the point before you changed the history [4]. As such, Mercury is more safe to "play around with" if you're not quite sure what you're doing, while Git allows you to do more if you know what you're doing. This user friendliness is also emphasized by Mercurials GUI options [5].

Mercurial key points

- Complete version control
- Distributed
- Quick
- Efficient space usage
- Easy to learn, SVN-like commands
- Better GUI support
- Rigid

4.6 Programming languages

In order to select candidates for which programming language to base our project on, we looked at the current experience of our group members (see table 4.6.6). In addition to having some prior experience with the programming language, we have decided to evaluate a programming language in these different ways.

- Performance: How quickly does a program in the given language run?
- Development time: How quickly can you develop a program?
- Tools: Is there developed good, mature tools in the languages?

- Popularity: Is the language rising or declining in popularity?
- Web suitability: Is the language suitable for web development?

Performance and development time

Performance, while never being a bad thing, is often negatively correlated with development time [68]. Figure 4.8 illustrates this by showing the runtime and code size of 13 different implementations in Python, C, C++, C#, Java, Ruby, PHP and Javascript. In our project, we will focus on illustrating the possiblities of well path 3d modelling, which is will not be especially computionally heavy, as it is only serving relatively small amount of data through a webpage. Hence, short development time is significantly more important for us than performance. Further thoughts on possible performance bottlenecks for the future product can be found in section 10.

Tools and Popularity

The popularity of and community around a programming language is of vital importance. There are many reasons to this: bugs usually do not lay dorment. Tools and different uses of the language is being developed. The ease of which to find good resources online, as well as programmers to further develop your project. We are willing to take the bold statement that popularity is of more importance than the qualitative qualities of the language itself. Since the Wellvis project has a long time horizon, we have used the TIOBE Programming Community Index [96] (see figure) and LangPop [65] to evaluate the development of language popularity over time. This could be a good indication of how relevant the languages are in 5-10 years time.

Web suitability

The customers wish for a platform independant software, and the trend for putting applications in the cloud makes web suitability an important aspect. Different programming languages usually have different areas where they excel, and as such might not be equally suited for developing web applications.

4.6.1 C and C++

C is probably the most influential and commonly used programming languages of all time [65] [96]. It is a low-level language with high performance, meaning it should be considered for computionally heavy operations.

Developed around 1980 C++ is basically an extended version of C [10]. It is similarily very fast,

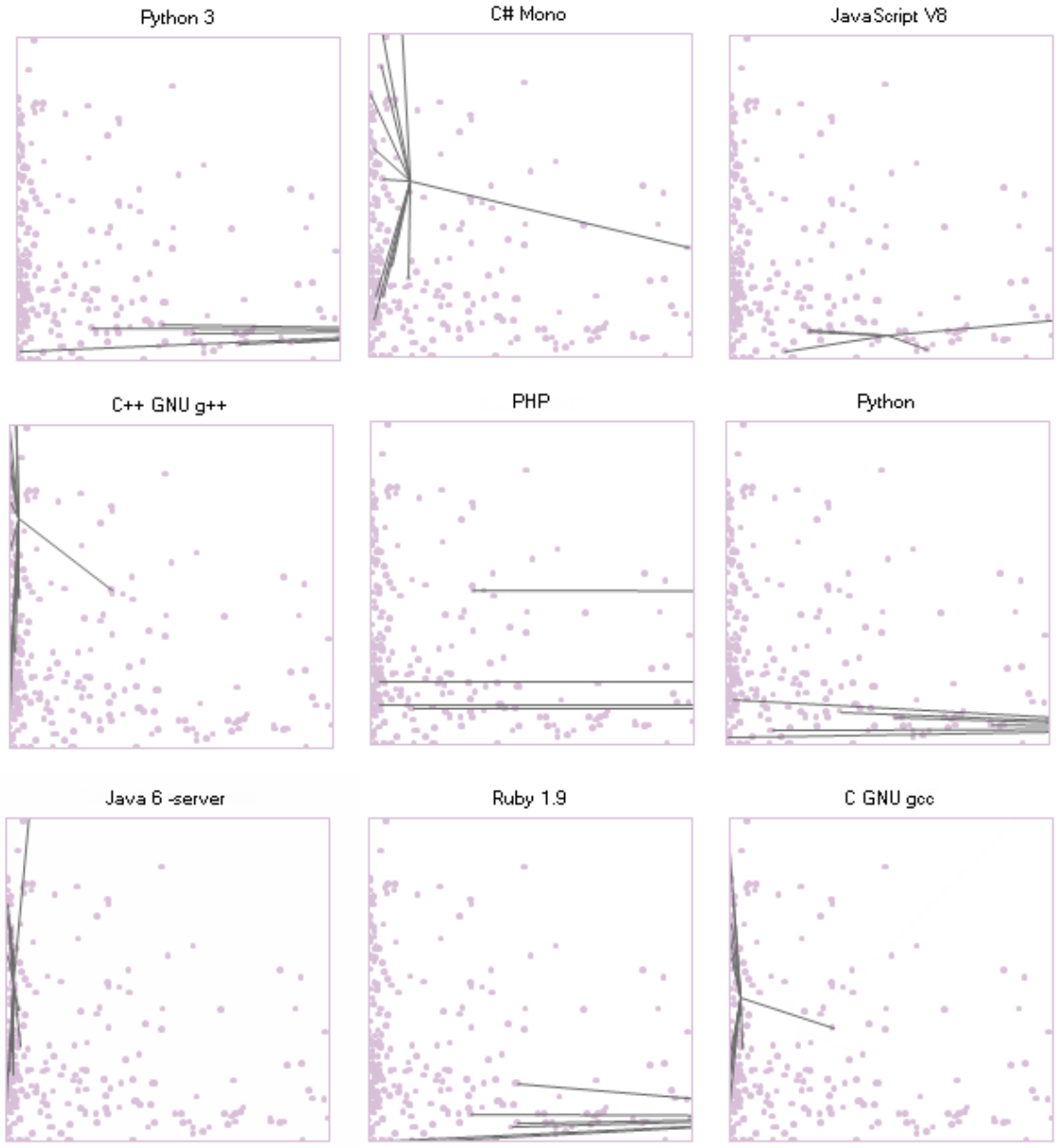


Figure 4.8: Speed and size of different programming languages [15]. Higher value on x-axis relates to longer runtime (lower performance), while y-axis relates to code size.

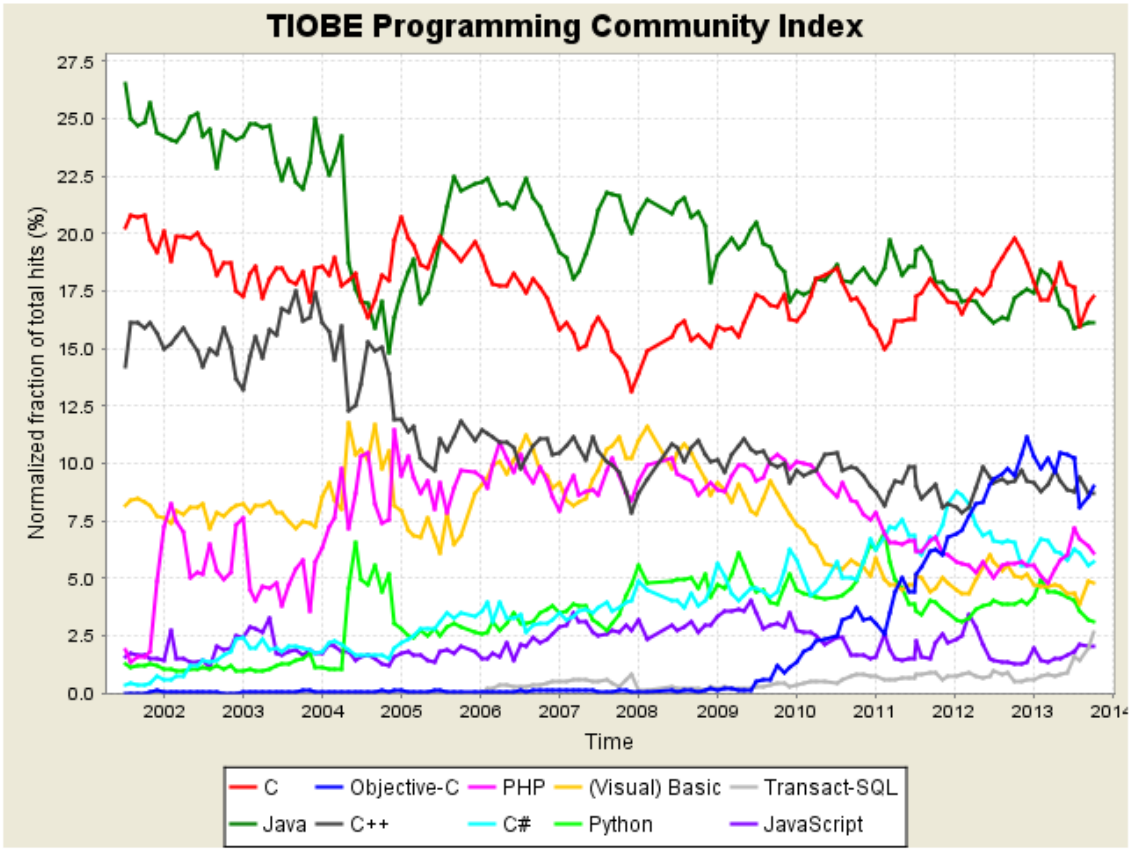


Figure 4.9: The TIOBE index [96] shows the popularity of different programming languages based on web search results over time.

though also comprises of high-level language features such as classes and exception handling. This shortens the implementation time compared to C, but is still somewhat slow to program in. The benchmark (fig 4.8) illustrates this for both C and C++ by having a consistenty low run time (far left on x-axis) with the price of large code size (far up on the y-axis).

Due to the fact that these are low level languages, the tools are usually related to computionally heavy tasks and crunching numbers. There exists C++ web framework, but they are not plentyfold, and the community around them are rather small. Popularity for these languages are declining, from a combined 35 % to 26 % since 2002.

- Great performance
- Long development time
- Very popular, but declining
- Great for computionally heavy tasks
- Limited web relevance
- Low experience within group

4.6.2 C# and Java

Strongly influenced by C and C++, C# is a popular programming language developed by Microsoft. Naturally, it is easily embedded in Microsoft environments. C# is very similar to Java, both in performance, tools, usage and especially syntax. Figure 4.8 shows C# falling a bit behind in performance, but not by a large margin. It would be probable that this is due to the long standing popularity of Java, that gives Java a head start in optimalizing compiler performance. Development time is slightly shorter compared with C and C++, and performance somewhat lower. Popularity is decreasing for Java, from 26 % to 16 % since 2002, while C# has had a steady increase from 1 % to 6 %.

- Good performance
- Long development time
- Java: Very popular, but declining
- C#: Popular, increasing
- Java: Much experience within group

4.6.3 Python and Ruby

Ruby and Python are two interpreted, dynamically typed languages. The popularity of the Ruby language increased greatly with the release of Ruby on Rails, a popular web framework released in 2008 and is currently in use by large sites such as Twitter and Github [75]. Similarily, Python has a popular web frameworks [88]. They both have short development time, but pays for this with their low performance due to their dynamic nature (as indicated by figure 4.8). Both languages can though be extended with C, C++ and Java and as such eliminate bottlenecks in computionally heavy parts by rewriting code in the appropriate language. Both Python and Ruby have had a decent increase in popularity since 2002, from 0.25 % and 1.3% to 1.5 % and 3.5 % respectively.

- Lower performance
- Open source
- Popular advanced frameworks
- Extendable to C, C++, Java

4.6.4 JavaScript

According to w3schools [124], JavaScript is the worlds most popular programming language. Whether or not that is true, we will not say, but it is certainly the case that it is being used on almost every single modern web page. It has also gained a lot of attraction after the popularization of HTML5 and Apple subsequently posting their Thoughts on Flash [2] in 2010, describing Flash as lackful and stating that several Apple devices would no longer support it. While the TIOBE index does not show a significant increase in JavaScript popularity, it is widely acknowleged that JavaScript is the web programming language for the decade to come, and will replace Java Applets and Flash in the web browser [14] [76] [108] [38].

- Platform independant
- Can be used for client and server side purposes
- Access to hardware graphics through HTML5

4.6.5 PHP

PHP is by far the most popular language for web development [115]. Many user-friendly and popular content management systems, like Wordspress, Drupal and Joomla has been developed in it. That being said, PHP has been criticized for difficult debugging, and a non-conform way of doing things, claiming it to be potentially messy when the projects reach a certain size and hard

to pick up others' projects [73]

- Potentially messy
- Lower performance
- Open source
- Extendable to C, C++, Java

4.6.6 Other languages

Even though there is familiarity with the languages within the group, ActionScript, objective-C, JSP and SQL has not been evaluated as thoroughly. The reason for this is the different nature of these languages. SQL is a database query language only egible for database related tasks. Actionscript (shortned AS in table 4.6.6) is a dialect of ECMAscript focused towards Adobe Flash, and is in our opinion a non-viable option (any longer) in client-side graphics emulation. JSP is a lightweight version of Java to dynamically create webpages, and has been excluded due to its simplicity. Objective-C has simply been excluded due to the lack of experience with it within the group. It should be noted that it is a good option, ranking high on TIOBE index, gaining much popularity the last few years with web frameworks such as Typhoon [114] and Frothkit [46].

	Java	Python	C#	C/C++	obj-C	AS	JS	JSP	PHP	SQL	Ruby
Tomas											
Tintin											
Tina											
Pawan											

Know well or have used recently
Have some limited knowledge of
Have never used before

Table 4.2: The groups familiarity with different programming languages.

4.7 Parsers and libraries

Here we will mention the different parsers and libraries that could be relevant in our project. We start of with WebGL frameworks in 4.7.1, and continue on with jQuery, 4.7.2, and CSS and visual libraries in 4.7.3

4.7.1 WebGL Frameworks

Making the application interface in a web browser is a very strong candidate for this project. If we end up doing this, using WebGL to render the well graphics is implicit simply due to the standardisation of web technologies. WebGL is a low level API for creating 3D graphics in the browser [128]. Due to this, we would use a Javascript WebGL framework to generate the code. Compared to the alternatives, using WebGL with a Javascript framework has several advantages:

- A framework gives much faster development.
- All major browsers support JavaScript and WebGL (platform independent).
- The technologies has become popular the last few years.
- WebGL is expected to be a popular standard for decently long period of time (longevity).

With WebGL we can develop graphically rich programs easily with JavaScript as our client-side scripting language. The one major drawback with using WebGL frameworks s that the technology is immature. The current shift going from Flash and Java Applets to WebGL for web-based applications is in its childhood. There is therefore a lot of experimenting going on, which means that there is a great possibility that the code we write will be outdated within only a couple of years. We do not consider this to be a huge flaw, as the pure WebGL code can be extracted from the code in the framework, however we would like to minimize the risk of this being necessary. To do this, we wanted the best established framework, with the most stable version available. We also considered that a large community around the framework would be a good indicator of it being continuously developed, and therefore be a good choice in the long run.

Requirements

- Detailed documentation
- Good tutorials
- Variety of demos/examples
- Large support groups/communities

The candidates for frameworks are plentyfold. As per november 2013, 36 different frameworks were listed on the official wiki [129]. We chose the following 3 candidates for comparison:

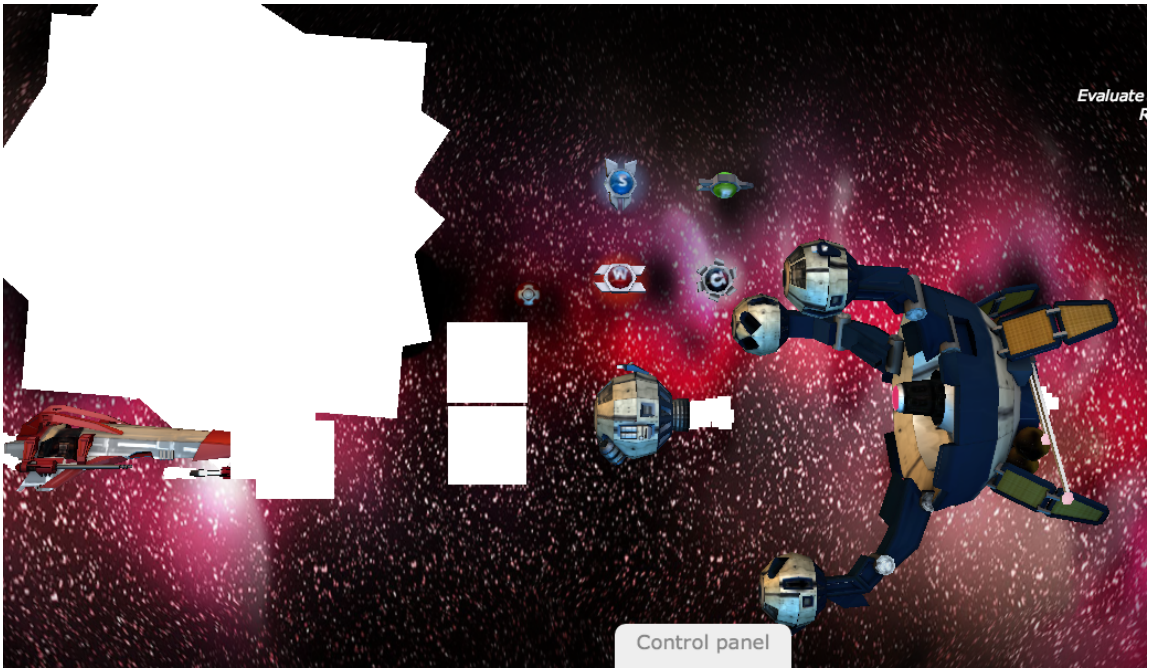


Figure 4.10: One of the demos at the main page of Babylon.js seemed bugged.

Candidate: Babylon.js

Babylon.js [6] has a really nice demos with very rich graphics. The examples are nice and interactive. However, the tutorials and documentation seems lacking, and there is limited community activity around it. In addition, one of the demos seems bugged in Chrome on OS X 10.7 (see figure 4.10)

Googling "babylon.js" yields about 10 000 results as per september 2013

Candidate: Canvas 3D JS Library (c3dapi.js)

Canvas 3D JS [61] is another framework for 3D modelling which lacked proper tutorials. In fact the tutorial it provides is the only tutorial so far and it's not very easy to understand. It got complicated on the tutorial on how to use the camera settings and lighting effects. Even though it is not a special requirement for our project, most demos seemed to have some performance issues on one of our computers, even on simple demos (see figure 4.11. We consider this a flaw that might prevent it from gaining popularity and as such not be in development a few years from now.

Googling "canvas 3d js" yields about 28 000 results as per september 2013

Figure 4.11: A simple astroid demo used on the homepage of Canvas 3D JS. This one, and other demos, had performance issues even though they seemed relatively simple.

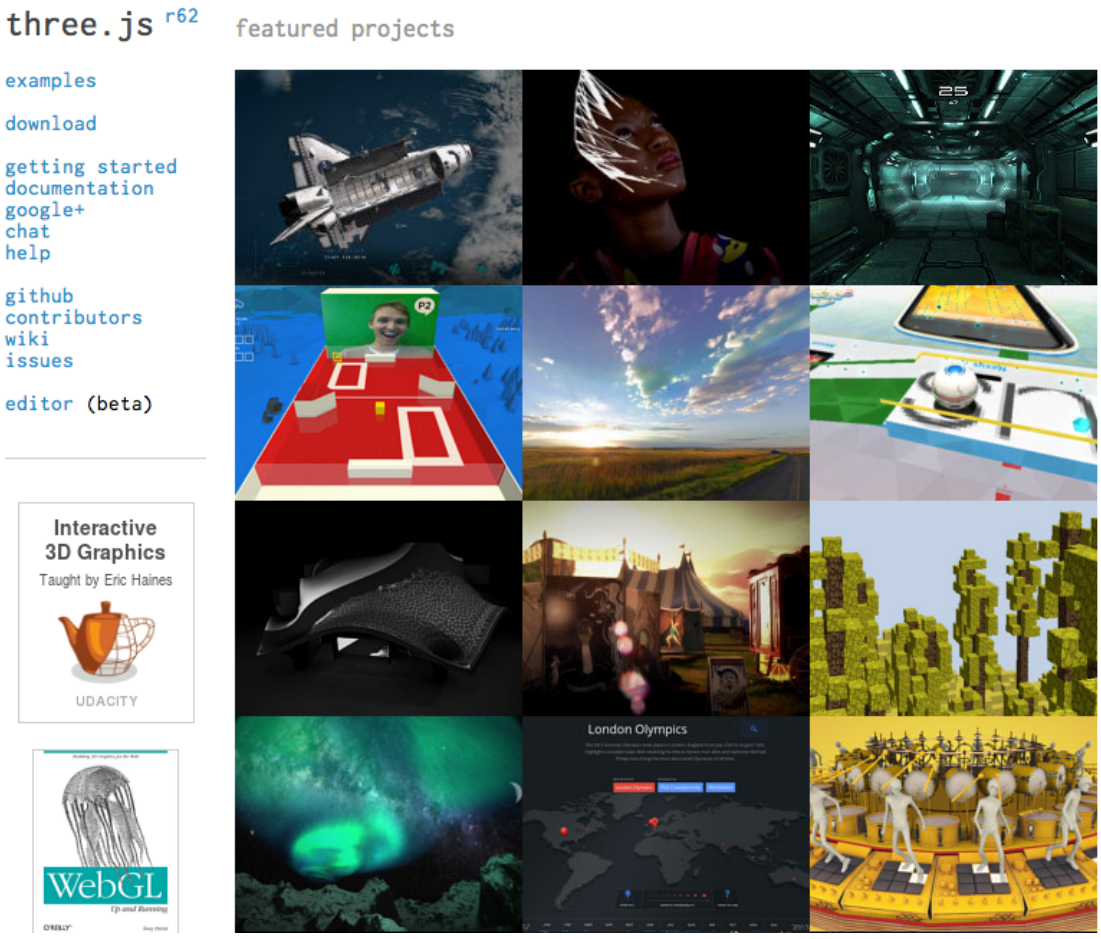


Figure 4.12: THREE had a variety of great demos than ran smoothly on their homepage.

Candidate: THREE.js

Three.js [107] had a really large set of examples (see figure 4.12) and a lot of tutorials. Documentation was great, and it's widely used. The livelihood in community around it and amount of help offered on the internet, makes us think that this library will still be used in many years to come.

Googling "three.js" yields about 430 000 results as per september 2013.

4.7.2 jQuery

Since its apperance in 1995 [72], JavaScript has become *the* programming language for web pages and browsers [69]. While it is in itself a high level language, the amount it is being used in has created fertile grounds for Javascript libraries to spire. In this section, we will present a library to make JavaScript even simpler and faster to write, namely jQuery. jQuery [58] has become a defacto library to simplify client-side JavaScript. It masks the different handling of different browsers, and simplifies writing JavaScript by a large margin. It's used by almost 70 percent of the top 10k webpages [12], and team members are already familiar with the it. The library is a bit heavy, and when making small or lightweight web applications (for example for mobile phones), alternatives should be considered. However, for this project, we're confident in choosing this tool without looking into alternatives.

4.7.3 CSS and visual libraries

In this section we will look at different tools to simplify front end development. The purpose of these tools is to provide simple, beautiful elements, css-styles and small javascript snippets to help you make a mobile-friendly and nice looking webpage with minimal amount of effort. These sort of tools has not been common for long, but had a rapid increase in usage the last year or so.

Foundation

Foundation is a frontend visual framework for enabling quick and consistent creation of web page layout [42]. Other than standard features like forms, buttons and dropdowns it has cool features like enhanced tooltips, image galleries, form validation and a neat little thing called Joyride, which gives you a tour of the site. Foundationss usage is still small, but has been increasing from about 0.01 % to 0.1 % in the last 6 months [11].

Twitter Bootstrap

Twitter Bootstrap is the *big brother*, or the least small brother rather, in this category of tools. While it does not really have any functionality outside the basic elements, the community around Bootstrap provides several hundred extensions so you can create many types of beautiful elements without being a designer. It is well tested and the most popular tool of this type we could find. Twitter Bootstrap has a 500 % increased usage the last 6 months, now being used by about 4 % of the top 10 000 most visited web sites [13].

HTML Kickstart

HTML Kickstart [62] has more or less the same visual elements as Twitter Bootstrap, perhaps with a slightly smaller size of options. It also got a few helper features such as adding .first and .last to list items. In our opinion, HTML Kickstart elements are especially beautiful.

4.8 Web frameworks

Unless one is creating a very small or very special application, frameworks are in most cases a must-have. They provide a general skeleton that can simplify security, user- and access-management, databases, even GUI. It also often gives you a rough sketch for the file structure of the application, which makes it easier for other to take over the project when you're done. It's a jungle of frameworks out there, that can handle near-to-nothing to nearly everything.

Using a Web Framework for this project is an obvious choice. It allows us to focus our time on the functionality of the application, not the potentially large overhead of server configuration, protection against misuse, database choices and setup. Also, our codebase at the end of the project will be easier to understand and continue to use. Under are the candidates we considered for python web development.

4.8.1 Django

Django calls itself "the web framework for perfectionists with deadlines". It was originally made in 2004 for newspaper sites, and make pages quickly [32]. This is reflected in that Django has a complete user-friendly admin interface delivered right of the bat. It is also centered around the idea of separated applications within the whole web page. It is a rather heavy framework, which allows programmers to easily get into it, but not as easily run the show. Django is centered around the logic and backend (model and controller in MVC), but does not impose any restrictions, nor help with frontend like HTML5 and JavaScript. Django has a dynamic database API that allows you to easily migrate between different DB solutions and integrated test framework that allows you to easily run tests. It also has really good documentation and a large community.

4.8.2 Flask

Flask is a very simple and lightweight python framework [41]. Its core is small, but there is possibilities to extend this to enhance its features. Flask doesn't come bundled with other libraries, and may such be seen as both more configurable and free, but at the same time more time consuming to start up with (before you get the good stuff).

4.8.3 Pylons

Pylons is a collection of web related technologies currently in development. It has released the Pyramid web framework, which uses python language [80]. It makes extensive use of WSGI(web server gateway interface) standards to promote reusability and to separate functionality into distinct modules. It is also strongly influenced by Ruby on Rails. Pylons have multiple database support and is very good for complex sites. Pylons makes it easy to pick components like URL router, ORM, template system etc, and plug them together. But then there is not so much flexibility to reuse applications. Pylons has a high learning curve and it takes time to get something to work well in pylons. Unfortunately, there is not so much documentation on pylons as there is on other python frameworks.

4.9 Unit testing frameworks

Using unit testing is not something that goes without saying that we would use in a project as small as ours. It requires a good chunk of time, and can feel like a waste of time. Especially for relatively inexperienced programmers like ourselves, which has yet to feel the crackles of a large project about to fall over our heads. However, since the aim of this project is to become a start, or if we're being ambitious, the first cornerstone of a rather comprehensive software suite, we think it's beneficial for the customer that this first stone be a solid one. It it also a valuable experience for the team members that have not been through this before.

4.9.1 Unittest

For Python version 2.1 and newer, the unittest library is the standard testing module [87]. It works pretty much the same as testing in any other programming language: Classes are derived from unittest.TestCase, and tests are run as assert variable == expected_value.

Django is already delivered with unittest set up, so that one easily can run all test cases within the project (or individual application) with a single line of code. Django will then set up a temporary test-database for the test run, making sure no data is actually created or deleted from the original database. This allows tests to be run in production environment without much hassle.

4.9.2 Pytest

Pytest [81] is a library thats simple, great and makes it easy to write tests. No need to extend from TestClasses and has no external dependencies. This seems like a very "clean" way of testing, with possibility to use test-databases. All in all, it seems like a slightly tweaked or better version

of unittest.py.

4.9.3 Selenium

Selenium [93] is a type of test tool that runs the (web) application interactively in the web browser. Where the other test tools verifies that program methods works as they are suppose too, by sending python parameters in and evaluating it's output, Selenium has a more abstract/grand testing. It will open a given webpage with a given browser, and do the tests within the browser. This allows you to do tests that both check JavaScript and other frontend elements, compatibility against browser version and basically do automatic, reliable user testing, without the actual user. It checks, clicks and interacts with DOM element based on xpath. A possible result is that the tests from early development can be prone to break later, if the HTML elements change too much [23].

4.9.4 Doctest

Doctest [83] is a very simple form of writing simple tests within the inline documentation in the python code. This can make the code look more messy, especially with test methods that require a lot of setup. On the other hand, this will connect documentation and testing along with the code, and work as a reminder to do these things. It can also make it easier to understand the code for future developers, since they're all located at the same place. Just like unittest, doctest is also integrated with django, making it easier to run and verify the tests.

4.10 User documentation tools

We wish to use a documentation tool in order to extract inline documentation from the sourcecode out into a more readable format. The motivation for this is being able to ensure consistency between those two locations of documentation, as well as saving time.

4.10.1 Djangos admindocs

Django comes integrated with an app called admindocs [20] that automatically pulls documentation from docstrings of models, view, template tags and filters from application within the framework and dumps this to an HTML file. Simple markup within the documentation can create hyperlinks to other parts within django.

- Simple
- Automatically integrated with Django

4.10.2 PyDoc

PyDoc is a simple doc-generator included by standard with Python installations [86]. Creates a simple, clean HTML file for document, based on doc-strings within the code.

- Simple
- Python standard

4.10.3 Doxygen

Doxygen is a multi-language doc generator, with a large set of possible outputs-formats [35]. Can even create a graph to graphically represent the interactivity between the different classes.

- Requires more time to use
- Advanced features
- Graphical representation of class interaction

4.10.4 Sphinx

Sphinx [100] is a tool that makes it easy to create intelligent and beautiful documentation, and is the tool used for documenting Python v3.3.3. It can export documentation to many formats: PDF, LaTeX, HTML, man pages and plain text.

- Requires more time to use
- Advanced features
- Several export formats

4.11 Integrated Development Environments

In this section we mention different integrated development environments that it is possible to use for our project. We start of with discussing VIM in 4.11.1 and continue with Sublime Text in 4.11.2. We also discuss Eclipse, which we have all used previously in section 4.11.3 and we also discuss PyCharm in the end of this section, 4.11.4.

4.11.1 VIM

Vim [117] is a simple, old school, yet powerful text editor that can be set up as IDE. Primarily used from the command line, but GUI versions are also available. Popular in some circles, has a lot of community around it, but requires a lot of time to master. We do not recommend any members that does not have previous experience with vim to start with it during this project, due to the limited time frame and high user threshold.

4.11.2 Sublime Text

Sublime Text [104] is a simple, yet powerful graphical text editor. Popular, with a lot of available plugins to help or simplify programming. For example, SublimeLinter can help you write PEP8. Is in it's original form just a simple text editor, and it might take some time to enjoy it's full potential in it's vast amounts of additional modules.

4.11.3 Eclipse

Eclipse [40] is a full IDE that most team members are familiar with from previous subjects. It requires some setup to integrate with Git, but provides a very rich environment. Hated by some, loved by others.

4.11.4 PyCharm

PyCharm [57] is a full IDE in the sense of Eclipse and similar, with autocompletion and refactoring. It is somewhat simpler than Eclipse, so that the setup is less hassle. It is already delivered with easy-to-use git integration, but is unfortunately a pay-to-use product. Can be used as a 30 day free trial for those that are interested, but is pay to use after that.

4.12 Evaluation and Conclusions

In this section we will evaluate and conclude with everything we have mentioned in each section of this chapter. This means that we start of with software development methodology in 4.12.1 and end with integrated development environments in 4.12.10.

4.12.1 Software development methodology

The project is very loosely defined and the customer has no hard requirements. To the contriary, we were encouraged to come up with ideas as we went along and do what we thought would be best. This dynamic sort of developing is a very good fit for an agile methodology like Scrum, and similarily a poor one for the waterfall method.

For this reason, we went for scrum.

4.12.2 Project task management tools

For project task management tool, we decided to use Pivotal tracker. There was two main selling points. It's automatic generation of work reports from a given time period minimizes our time spent documenting the progress of the project, and lets us focus on work with the software itself. The fact that it also supported service hook integration with github simplifies the connection between task and code, and serves as another form for online documentation. The only potential obstacle with the tool, is that it has to be a public project. However, this was not an issue with our customer.

4.12.3 Illustrative drawing tools

We chose to use draw.io for illustration purposes. The main argument for this was that is was free, and had decent all-round capabilities that served all of our needs. Since one of our group members already had Microsoft Visio installed, he also used that for some illustrations.

4.12.4 Version control

Git was easily superior to SVN due to both functionality, and quantitative measures such as space usage and speed. Compared to Mercurial, we have not found hard objective arguments for choosing one or the other. Instead we went for Git due to our prior, positive experiences with git and the complementing web service, github.

4.12.5 Programming languages

We got our eyes on Python and Ruby with complementing JavaScript quite early. Javascript for the exciting development that happens in that area at this time, and the high probability that it will continue to be the dominent technology for web graphics. For Ruby and Python, a reason was the project directive to get our ideas out of our heads into prototypes and as such wanting low implementation time, which eliminated C, C++ and to some degree Java and C#. Also, it

was important that our code was easy to understand, so the product could be developed further by others. This made PHP a less lucrative choice. The choice came down between Ruby and Python. Our choice here was mainly influenced by the group members prior experience with the programming languages.

4.12.6 Parsers and libraries

WebGL frameworks

The technical possibilities of all the candidates exceeded our need in this project. We therefore chose based on the ease with which to learn it. Since none of us had prior experience with 3D modelling in JavaScript, the documentation and community around it weighed the most on whether or not we wished to use it.

Three.js is without doubt the library with the largest support groups and community contributing to it. In addition, it had a good amount of tutorials to learn from. The documentations were detailed and had a large set of demos – from simple to advance examples. It supports HTML5 Canvas, WebGL and can even fallback to SVG setting for older browsers.

jQuery and visual libraries

jQuery was chosen to use due to its simplification, time saving and increased readability compared to plain JavaScript. Among the visual libraries, Twitter Bootstrap was chosen due to a prior knowledge within the group and well established community that allows us to extend it if deemed necessary.

4.12.7 Web frameworks

The Django framework was chosen due to the impeccible documentation available online. It has also been used in high traffic pages like The Guardian and Washington Post for a small decade [136], and as such proved reliable. Django simplifies a lot of things through powerful built-in tools. In addition, we think Djangos clean and simple structure makes it easy for Wellvis to further use and develop the programs we develop in it.

4.12.8 Unit testing frameworks

Regarding testing, we will initially try to use the doctests, i.e. unit testing within the actual code. If this becomes too messy, we will make a transition to unittest.py. This transition shouldn't take

much time, nor any configuration on the django side of things. Even though pytest.py is a bit sleeker in our opinion, the difference is too small to make up for the Django integration. Since testing with Selenium in it's basic sense is a programmatic way of testing the user test cases, there is no necessity to implement this early. Depending on our progress and development later in the project, we can consider to use Selenium.

4.12.9 User documentation tools

In true Scrum style, we start out with the simplest parts, and expand with other tools if it turns out they are needed. In such a way of thinking, starting with the integrated Django tools seemed both logical, and timesaving. The nice thing about this is that Djangos admindocs bases itself of the docstrings within the code. The more advanced option, Sphinx, can also do this, making a potential transition in the future not causing double-up documentation.

4.12.10 Integrated Development Environments

There is no obvious reason to force anyone to use a specific IDE or editor. In the short time span of this project, we see it as most beneficial for the work progress if no one has to learn a larger range of tools than necessary. The choice of IDE is also a matter of taste, as much as a matter of what's objectively best. The four softwares mentioned are among the most popular environments to program in, and are all good candidates.

4.13 Intellectual Property Rights and License

Below are the licenses used in different parts of our solution. Section 4.13.1 describes parts that are integrated in the delivery itself. An example of this is the three.js library used for rendering WebGL graphics. Section 4.13.2 describes parts that have been used to simplify the process, such as virtualenv. These are not strictly a part of the solution, but used in setup. Section 4.13.3 describes tools that have been used outside the technical solution, such as Pivotal Tracker for tracking project progress. We have had a focus in the project to use open source and free products as far as we can. For a simple overview, colors are used to indicate licenses. Green indicates it is free to use and change for commercial purposes, commonly provided that the *parts themselves* are delivered with the same license. The color yellow indicates is free to use, under some constrictions while red indicates it is propietary and cannot be used without a license.

4.13.1 Integrated parts in solution

Program	License
Python	PSF [85]
Django	BSD [19]
South	Apache license [98]
jQuery	MIT [59]
Twitter Bootstrap	GPLv2 [8]
Three.js	GPLv2 [106]
Spectrum	Simplified MIT Copyright [99]
Growl	Simplified MIT Copyright [60]

Table 4.3: Intergrated parts in solution

4.13.2 Libraries used

Program	License
virtualenv	MIT [118]
Git	GPLv2 [48]
Pip	BSD [79]

Table 4.4: Libraries used

4.13.3 Tools used

Program	License
Draw	Copyright [36]
vsftpd	GPL [122]
netatalk	GPL [70]
denyhosts	GPLv2 [17]
vim	GPL [116]
screen	GPL [91]
opensshserver	BSD [78]
Microsoft Visio	Copyright [121]

Table 4.5: Tools used

Chapter 5

Test Plan

This chapter is all about testing and how we are doing it. The purpose with this chapter is to have a structured way to test our solution and to make sure that we follow our plan. We start off with different testing methodologies in the first section 5.1 where we discuss how it is possible to test a system. Then we discuss our non-functional requirements and how it is possible to test these in 5.2. We show our testing templates and what they include in 5.5.

5.1 Testing Methodology

When testing a software system, we have three different types of tests available. In 5.1.1 we will talk about how to do white box testing while 5.1.2 will talk about black box testing and how to do this. There is also grey box testing, which is a combination of the other two, but we will not discuss this.

5.1.1 White box

The method of software testing with the white box method is where you test internal structures or modules of an application, as opposed to its functions. White box testing requires the tester to have an internal perspective of the system as well as sufficient programming skills. Because of these reasons we will do the whitebox testing ourselves. This is done by using the test frameworks mentioned in section 4.9.

5.1.2 Black box

The method of software testing with the Black box method is where you test the functionality of the system, as opposed to its internal structures. Black box testing does not generally require the tester to have an intimate knowledge about the system or any of the programming logic that went into making it. It is primarily interested in the relationship between the input and output of the system. Not having any knowledge about how the system is built can help detect errors that the developers did not foresee. We want the customers to do the black box testing since they are the ones who knows how it is supposed to be used.

5.2 Non-functional requirements

Our non-functional requirements is about usability and modifiability.

Usability is easy to test. When showing the prototype to the customer we have showed them how everything works and let them tell us if something is not as intuitive as we would like it. We have also given them the possibility to test the system and come with comments during the project. The usability of the old software is really bad. So improving this has been no problem since we have been thinking through this whole process if what we have done is easy enough to understand. And we also tested this on the team members who have not worked so much on the code. If we were to make a completely finished product we would have done a full usability test to see if the system works good enough for a user to use in their workplace. They would then fill out a questionaire, with a scale from 1-10, where we would say that the usability is good enough if we have an average score on 7.

It is harder to test the modifiability requirement. This mostly is about that this module should be easy together with all the other modules, but since those are not made this is difficult to test. We have however changed requirements and such over the process which has made us test the modifiability of our system by adding new functionalities and such. So is is modifiable in itself, but since the other modules does not exist we can not test if it is modifiable with those.

5.3 Tests

Since we only are making a part of a software, and because it is only a prototype, we will not do all types of testing. This is because we feel that most of the tests like: performance, acceptance and system tests is more appropriate for a finished product, and not a prototype that is not finished.

User-tests

This project is not about making a finished software for our clients, but it is a prototype for them to keep working on later. Since we will not have a completely finished product with all requirements, we do not feel the necessity of doing user-testing. We feel that this is more suitable for a finished product that we know that the customer will use later. Also we have had meetings after the first sprints, where we have gone through what the customer liked, and what they felt were missing and could be better, and we have had close contact during this project so that is why we have not had any user-testing.

5.4 Test Criteria

An item will be considered to have passed a test if the actual result from the test matches the expected result from the test. An item will be considered to have failed the test if the output varies from the expected result. If there are some specific reason why the test failed it will also be documented.

5.5 Testing Templates

We have made two templates for our testing. The first template has info about the test case, what execution steps we will make and what the expected result should be. In the other template we have the test results. Here we can see when we tested what test case, and what the results are.

ID	ID of the test
Description	Description/title of the test
Precondition	Precondition that needs to be fulfilled to perform the test
Feature	What feature of the project does this belong to
Execution	Execution steps
Expected Result	Expected result

Table 5.1: Test case template

ID	ID of the test
Description	Description/title of the test
Tester	Name of tester
Date	Date of the test
Result	Result of the test

Table 5.2: Test result template

5.6 Testing Responsibilities

We had a few ideas about hot to divivde the responsibilities when testing, but because of little time together we decided to keep it simple. Each programmer has most knowledge about their own code, so they are responsible for making unit testing for their own part. The test manager however, is responsible for getting feedback from the customer when they do black box testing.

Chapter 6

Architectural Description

There are many ways to define software architecture and we have chosen the following definision: "The software architecture of a system is the set of structures needed to reason about a system, which comprise software elements, relations among them, and properties of both" [66, p. 4].

This chapter will introduce the final architecture of the product. It will describe the qualities in the system we wanted to achive and how it was structured to accomplish them. In section 6.1 we discuss which quality attributes had the biggest impact on the software structure. In section 6.2 we describe the tactics implemented to fulfill the architectural drivers. In section 6.3 we go through the architectural patterns that is included in our software. We look at the architecture from different views in section 6.3.3. Finally, in section 6.4 we give a walkthrough of our implemenation.

6.1 Architectural Drivers

After the first customer meeting, we identified the main architectural drivers as modifiability and usability.

Modifiability was important because the project only resulted in a small module which was part of a much larger system. We therefore focused on making our system easy to intergrate since it would have to work with other modules. This would reduce some of the main problems with the old system, where the modules had poor co-operation and introduced redundant work. Extensibility was something else we wanted to achive. The module was only a early prototype and would have to be able expand with more functionality at a later date.

Another large issue with the current system is that it has a high skill ceiling and users often struggle to complete their tasks. The focus was put on making the system more forgiving and intuitive.

6.2 Architectural Tactics

A tactic is a technique that can be used to achive a required quality attribute [66, p. 70].

6.2.1 Modifiablity Tactics

Modifiability deals with change and the cost in time or money of making a change, including the extent to which this modification affects other functions or quality attributes. [66, p. 121]

There is a cost of preparing for change as well as a cost for making a change. The modifiabbility tactics are designed to prepare for subsequent changes. Changes can come from developers, installers, or end users. [66, p. 128]

Split Module

The modification costs of a module with large capability is usually high. Refining the module into several smaller modules should reduce the average cost of future changes. [66, p. 123]

Increase Semantic Coherence

If the responsibilites in a module do not serve the same purpose, they should be placed in different modules. The purpose of moving a responsibility from one module to another is to minimize the chance of side effects affecting other responsibilities in the original module. [66, p. 123]

Encapsulate

Encapsulation introduces an explicit interface to a module. This interface includes an application programming interface (API) and its associated responsibilites. Encapsulation reduces the likelihood for changes from one module to propagates to other modules. [66, p. 123]

Restrict Dependencies

Restrict dependencies is tactic that restricts the modules that are given module interacts with or depends on and restricting access to only authorized modules. [66, p. 124]

Refactor

Refactor is a tactic used when two modules are affected by the same change because they are (at least partial) duplicates of each other. Code refactoring is a mainstay practice of agile development projects, as a cleanup step to make sure that teams have not produced duplicative or overly complex code; however, the concept applies to architectural elements as well. Common responsibilities (and the code that implements them) are "factored out" of the modules where they exist and assigned an appropriate home of their own. [66, p. 124]

Abstract Common Services

In the case where two modules provide not quite the same, but similar services, it may be cost-effective to implement the services just once in a more general (abstract) form. Any modification to the common service would then need to occur in just one place, reducing modification costs. [66, p. 124]

6.2.2 Usability Tactics

Usability is concerned with how easy it is for the user to accomplish a desired task and the kind of user support the system provides [66, p. 175]. Once a system is executing, usability is enhanced by giving the user feedback as to what the system is doing and by doing and by allowing the user to make appropriate responses. [66, p. 179]

Cancel

The system must be built so it is able to listen for when the user issues a cancel command. There must be a constant listener that is not blocked by the action that is being canceled. Furthermore, the command being canceled must be terminated and free any resources reserved. It is also important to inform any components that where collaborating with the canceled action, since they may have to react to this event. [66, p. 179]

Undo and Version Control

To support the ability to undo, the system must maintain a sufficient amount of information about the system state so that it can be rolled back to a previous one. This may be in the form of "snapshots" of a state or a set of reversible operations. [66, p. 179]

6.3 Architectural Patterns

Architectural patterns are packages of design decisions that is found repetedly in practice, has known properties that premits reuse, and describes a class of architectures. Patterns can be seen as "packages" of tactics. [66, p. 204]

6.3.1 Client-Server

There are shared resources and services that large numbers of distributed clients wish to access, and for which we wish to control access or quality of service.

By managing a set of shared resources and services, we can promote modifiability and reuse, by factoring out common services and having to modify these in a single location, or a small number of locations.

The client-server pattern separates client applications from the services they use. This pattern simplifies systems by factoring out common services, which are reuseable. Because servers can be accessed by any number of clients, it is easy to add new clients to the system.

Clients interact by requesting services of servers, which provide a set of services. Some components may act as both clients and servers. We have chosen to use one central server, instead of multiple distributed ones.

Some disadvantages of the client-server pattern are that the server can be a performance bottleneck and it can be a single point of failure. Also, decisions about where to locate functionality (in the client or server) are often complex and costly to change after the system has been built. [66, p. 217]

6.3.2 Model-View-Controller

User interface software os typically the most frequently modified portion of an interactive application. For this reason it is important to keep modifications to the user interface software seperate from the rest of the system. Users often wish to look at data from different perspectives, which should all reflect the current state of the data. [66, p. 212]

The model-view-controller pattern seperates application functionality into three kinds of components:

- A model, which contains the application's data
- A view, which displays some portion of the underlying data and interacts with the userr

- A controller, which mediates between the model and the view and manages the notification of state changes

6.3.3 Architectural Views

We have chosen to use the 4+1 architectural view model. The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers and project managers. The four views of the model are logical, development, process and physical view. In addition selected use cases or scenarios are utilized to illustrate the architecture serving as the 'plus one' view. [64]

Logical view: The logical view is concerned with the functionality that the system provides to end-users. This view is represented by the class diagram, entity relationship diagram and sequence diagram.

Development view: The development view illustrates a system from a programmer's perspective and is concerned with software management. We have chosen to not include this view because our system is only consisting of one module and does not have many dependencies.

Process view: The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the runtime behavior of the system. The process view addresses concurrency, distribution, integrators, performance, and scalability, etc. We feel this is not necessary for our project, because there are no requirement for concurrency and distribution on multiple servers is outside our scope.

Physical view: The physical view depicts the system from a system engineer's point-of-view. It is concerned with the topology of software components on the physical layer, as well as the physical connections between these components. We have not included this view because our system is software based and does not have hardware requirements.

Scenarios: The description of an architecture is illustrated using a small set of use cases, or scenarios which become a fifth view. The scenarios describe sequences of interactions between objects, and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype. This view is being represented by use cases in section 2.3.

6.3.4 Architectural Rationale

The client-server pattern was used because of the nature of the clients. The customer wanted the system to be able to run on any operating system and being able to run on mobile devices. We have therefore used a web interface to support these requirements. This means that the server is hosting the application and doing the heavy calculations, while the clients only need to display a

screen of the results.

The model-view-controller pattern was a good way to organize our system. The system had to be able to handle multiple users with different roles and degree of access. By seperating the view from the model the system is able to display only parts of the model to the user.

6.4 Implementation of Architecture

This section will describe the structure and inner workings of our solution. A simplified class diagram describing the general structure of the solution can be found in figure 6.4.7. Figure 6.4.7 describes how these classes interact with eachother, while 6.5 gives a simple overview of how a user request is being processed in the application. The Django structure fits well with our requirement of modifiability. In essence, the solution consist of a simple base application and subsequent and separate subapplications or "modules". Each module consists of the five main parts shown within wPath container in figure 6.4.7 and under "Application" in figure 6.5 (tests.py is here excluded). These five parts are templates, urls, models, view and tests. We will describe these in sections 6.4.3, 6.4.1, 6.4.4, 6.4.2 and 6.4.5 respectively. In addition, there are locations for referenced folders shown with color green in figure 6.4.7. These are explained in section 6.4.6. A substansial part of our solution is the 3D graphic rendering located in the js folder. This part works very seperately from the django part, and is described in section 6.4.7. For illustration purposes, the link `www.wellvis.com/wpath/edit/1` will follow us through this section by providing an example of what happens at the end of relevant sub-sections.

6.4.1 Urls

Urls are located in `module/urls.py`. The urls.py files for the django application is usually the first modifiable part of django that responds to a user request. Looking at figure 6.5, a user enters a URL (labeled 1), which is handled by the base applications urls file. This sends the request to the application specific url file, illustrated with the label 2. The urls file in that application matches the url request with a corresponding method (labeled 3). This matching is also the single purpose for urls files, namely to define *what methods should reply to which urls*. We will describe the simple urls file for wpath module, which is shown in listing [6.4.1] and reference the more advanced url file for the main application (listing [6.4.1]) where applicable.

Meta

Looking at [6.4.1] listing, the first two lines import libraries used to simplify the syntax, by introducing patterns and url methods. Lines 4, 5 and 12 automatically discovers relevant information, like enabled applications. For more information on this, see *Django - The Django admin site* [21]. The same lines repeat themselves in listing [6.4.1].

Regex matching

The regex matching first happens in the main urls file. For illustration Line 7 to 9[6.4.1] are the interesting parts where we define valid urls for the application and corresponding response. The valid urls are defined on line 8 and 9[6.4.1] and 8-16, 28, 31[6.4.1], with a url method that takes three arguments. The first argument is a regex string the match the url the user enters. We see that line 8[6.4.1] responds to an empty string (to understand regex, see regex tutorial [112]). To understand what url this refers to, think of the complete url of any parts in three section. First, there is the url for the webstie, ala. `http://www.wellvis.com/`. Second is the one that defines the applications, for example `wpath/` which is defined by the main urls[6.4.1], on line 18-25. The third and last part is the specific url, which in this case is empty. Put together, this means that ie. `http://www.wellvis.com/wpath/` will respond to this entry.

Corresponding methods

While line 8[6.4.1] responded to the empty 3rd part, line 9[6.4.1] handles a slightly more advanced query. The `(?P<wellpk>[0-9]+)` takes in any string that is a number, labels the argument with a short-name "wellpk", and then sends it to the second argument, `wpath.views.edit_well`, a method named `edit_well` located within the `wpath` application, in the `view.py` file.

Names

The last parameter to the urls method is a name for that specific entry. This can be used to reverse lookup urls. This is not something we have used in this project. For more information on this specific utility, please see *Django - on URL dispatching* [31].

urls.py for wpath module

```
1  from django.conf.urls import patterns, include, url
2  from django.contrib.staticfiles.urls import staticfiles_urlpatterns
3
4  from django.contrib import admin
5  admin.autodiscover()
6
7  urlpatterns = patterns('',
8      url(r'^$', 'wpath.views.home', name='home'),
9      url(r'^edit/(?P<wellpk>[0-9]+)$', 'wpath.views.edit_well',
10       name='view-wellpath'),)
11
12 urlpatterns += staticfiles_urlpatterns()
```

urls.py for main module

```
1  from django.conf.urls import patterns, include, url
2  from django.contrib.staticfiles.urls import staticfiles_urlpatterns
3
4
5  from django.contrib import admin
6  admin.autodiscover()
7
8  urlpatterns = patterns('',
9      url(r'^$', 'wellvis.views.home', name='home'),
11     url(r'^dashboard/', 'wellvis.views.dashboard', name='dashboard'),
11
12     # Hierarchy navigation
13     url(r'^country/(?P<countryid>[0-9]+)$', 'wellvis.views.view_country',
14        name='view-country'),
15     url(r'^field/(?P<fieldid>[0-9]+)$', 'wellvis.views.view_field',
16        name='view-field'),
17     url(r'^platform/(?P<platformid>[0-9]+)$', 'wellvis.views.view_platform',
18        name='view-platform'),
19     url(r'^well/(?P<wellid>[0-9]+)$', 'wellvis.views.view_well',
20        name='view-well'),
21
22     # Modules
23     (r'^wpath/', include('wpath.urls')),
24     (r'^wcp/', include('wcp.urls')),
25     (r'^ajax/', include('wellvis.ajaxurls')),
26     (r'^wdesign/', include('wdesign.urls')),
27     (r'^wplan/', include('wplan.urls')),
28     (r'^wreport/', include('wreport.urls')),
29     (r'^wsupport/', include('wsupport.urls')),
30
31     # Admin documentation
32     url(r'^admin/doc/', include('django.contrib.admindocs.urls')),
33
34     # Admin-panel
35     url(r'^admin/', include(admin.site.urls)),
36 )
37
38 urlpatterns += staticfiles_urlpatterns()
```

Example

Using a user request `www.wellvis.com/wpath/edit/1` to illustrate the workings of the url parts of the application.

1. The user request `www.wellvis.com/wpath/edit/1` first responds to `/wellvis/urls.py`[6.4.1] on line 19.

2. `/wellvis/urls.py`[6.4.1] redirects the remaining part of the request (`/edit/1`) to `/wpath/urls.py`[6.4.1].

3. `/wpath/urls.py`[6.4.1] matches the remaining request on line 9 to method located in `wpath.views.edit_well` and send with `1` as the value of the parameter wellpk.

The request is then further processed by the view files, described in section 6.4.2.

6.4.2 views

Views are located in `module/views.py`.

Querying database

Django has its own layer of database querying on top of the relevant, native query language. This gives us an abstraction layer that allows the same code to be reused when moving from one solution to another. Querying database is used in most views files and their subsequent methods. This is done by calling methods *on* the Class from which the object is defined (models, section 6.4.4) For example in `wellvis/views.py` on line 78:

```
78 selected_country = Country.objects.filter(pk=id)
```

Where the `selected\_country` would become a QuerySet list variable of existing countries that satisfy the condition that variable `pk` equals `id`. More on Django querying can be found in the *Django - QuerySet API reference* [28].

Using external methods

Views are in its basic form a python file, and hence can classes and methods from external files in the same folder be included using:

```
from external_filename include methodname
```

However, application folders in Django are also made to be python modules automatically, where the application name becomes the module name. Hence one could from the wpath view file write

```
from wellvis.ajax include ajax_method
```

which would then include the ajax_method located in a manually created ajax.py file in the wellvis application folder into the relevant python file. As such, including external methods either from other django applications native django files or from manually created libraries is easily done.

Returning template with context

Datafields in the templates are filled with a context-dictionary from the view. For more information on this, see *Django - The Django template language* [25]

Returning other MIME types

Views does not have to return a HTML page to the user. It can also return any other MIME type [45]. For more information on this, see *Django - Django response objects* [29].

Returning additional messages

The solution is equipped with a Growl JavaScript snippet that enables the sending of additional messages to the user. For usage on this from the view side, see line 22 in 6.4.2. For display in the template side, see line 47-58 in `wellvis/templates/base/main.html` and the relevant javascript snippet, located at `static/js/jquery.growl.js`.

views.py's edit_well for wpath module

Listing below show the edit_well method in views.py file for the wpath module located in `wpath/views.py`.

```
1  @login_required(login_url='/')
2  def edit_well(request, wellpk):
3    """
4    Page for editing existing well.
5    If: no project exists. Send to create project page.
6    if: no path exists. Send to create path page.
7    else: Send to edit page with graph
8    """
9    context = {'messages':[]}
10
11   if not request.user.is_authenticated():
12     context['messages'].append(``403 Error: You are not authenticated'')
13     return render(
14       request,
15       'wellvis/home.html',
```

```
16        generate_sidepanel(context, request)
17      )
18
19    well = Well.objects.filter(pk=wellpk)
20
21    if not well:
22      context['messages'].append(``404 Error: Well '' + wellpk + `` not found'')
23      return render(
24        request,
25        'wellvis/home.html',
26        generate_sidepanel(context, request)
27      )
28
29    well = well[0]
30    project = Project.objects.filter(well=well).order_by('-created_date')
31    if not project:
32      project = Project.objects.create(
33        name = ``Untitled'',
34        responsible = request.user,
35        start_date = get_null_date(),
36        end_date = get_null_date(),
37        well = well)
38      context['messages'].append(``Empty project created as none existed'')
39    else:
40      project = project[0]
41    path = WellPath.objects.filter(project=project).order_by('-date')
42    if not path:
43      path = WellPath.objects.create(
44        project=project,
45        creator=request.user,
46        path=get_default_path())
47      context['messages'].append(``Default path created as none existed.'')
48      path.save()
49    else:
50      path = path[0]
51    context['wellpk'] = well.pk
52    context['page_title'] = ``wPath''
53    return render(request, 'wpath/3D_View.html', context)
```

Example

Using a user request `www.wellvis.com/wpath/edit/1` to illustrate the workings of the view parts of the application. The previous step in the procedure is described in section 6.4.1. The user is authenticated, the well exists, and a project and path is already attached to it.

1. The method is called with request details (automatic) and `1` as `wellpk` parameter.

2. Line 11 checks for authentication, and validates the user request.

3. Line 19 queries the database for a well with primary key 1, and returns a QuerySet [28] with the well object (illustrated as part 4 in 6.5).

4. Line 21 checks that a well with that primary key exists.

5. Line 29 transforms the `well` variable from a QuerySet to a Well object

6. Line 30 retrieves a QuerySet of Project objects related to that specific well, sorted with the most recent one first.

7. If else block on line 31 to 40 checks that a project exist, triggers `else` and converts to `project` from QuerySet to Project object.

8. Line 41 to 50 works identically to the Project section.

9. Line 51 adds wellpk as a key to a a dictionary [82], `context`. The value corresponding to this key is the primary key of the relevant well object

10. Line 52 sets "wPath" as value for the key `page_title` in context.

11. Line 53 returns the method (to the user) as a rendering of the template `wpath/3D_View.html` with `request` and `context` (illustrated as part 5 in 6.5).

The request is then further processed by the relevant template file, `wpath/3D_View.html` described in section 6.4.3.

6.4.3 Templates

Templates are located in `module/templates/type-name/name.html`. Templates are basically html files that are extendable with python similar django template syntax. Its functionality is simple and similar to basic PHP. This works in collaboration with the context sent from views-file when rendering a HTML response file to the user.

Extending a base template

A base template is in our project located in `wellvis/templates/base/main.html`. It is a HTML file that is reused in other templates to simplify creation, maintainance and updates to other template files. In order to extend a template, the parenting template needs to define blocks, where

children can input their text/data. An example of this is on line 16 in this parent template, which names a block and defines its position:

```
16 {% block styles %}{% endblock %}
```

The child templates of this needs to define content to go in this block using

```
{% block styles %}
    content goes here
{% endblock %}
```

Including partial templates

Subparts of the webpage, for example sidepanel and topmenu can be included in both parent and child templates through the syntax used on line 25 in our parent template. The included file is then copied in to the host template.

```
25 {% include "base/menu.html" %}
```

Using values from view context

Values from the python dictionary can be subject to simple procedures such as evaluation, iteration, attribute retrieval and textual representation in the template. Evaluation can be seen in `wellvis/templates/base/menu.html` from line 12:

```
12 {% if sidepanel_hidden == "1" %}
13 Context if evaluation is true
14 {% else %}
15 Context if evaluation is false
16 {% endif %}
```

Iteration, attribute retrieval and textual representation can be seen in `wellvis/templates/base/sidepanel.html` from line 28:

```
28 {% for country in countries %}
29   <li class=''nav_country''>
30     <a href=''/country/{{ country.pk }}''>
31       <span class=''badge pull-right''>3</span>
32       {{ country.name }}
33     </a>
34   </li>
35 {% endfor %}
```

Above, countries is a list of Country-objects which among other things have an attribute called name and also primary key. These attributes are extracted on line 29 and 32. The syntax on line 28 and 35 defines what to iterate (countries), and what to produce for each containing element.

views.py's edit_well for wpath module

Listing below show usage of non-HTML/CSS/JavaScript django template language within the HTML template file `wpath/templates/wpath/3D_View.html`.

```
1  {% extends "base/full_view.html" %}
2
3 {% block content %}
..
341 <link rel='stylesheet' href='{{ STATIC_URL }}css/spectrum.css' />
..
758 {% endblock %}
759
760 {% block scripts %}
761
762 <script src=``{{ STATIC_URL }}/js/3D/wellvis_model.js''></script>
...
816 var theWellJSONdata = get_wellpath({{ wellpk }});
```

Example

1. The template is rendered with a python dictionary containing the key `wellpk`.
2. Line 1: `base/full_view.html` is retrieved.
3. Line 3: indicates that the following lines (4-757) should be put into a block named `content` within the extended template, `base/full_view.html`.
4. Line 341: `{{ STATIC_URL }}` is being replaced by a valid url defining the folder for static content, retrieved from statics file [24].
5. Line 758: indicates the end of the content-block
6. Line 760: indicates that all lines up to next endblock should be put into a block named `scripts` within the extended template, `base/full_view.html`.
7. Line 816: Django replaces `{{ wellpk }}` with the key wellpk specified when calling the rendering of the file

For more information on this template syntax, see The Django template language section of django documentation [25].

6.4.4 models

Models are located in `module/models.py`. They define the objects in the database. They are written in a django-specific python-like syntax, and will populate a database defined in the settings [24].

Field model

The following code snippet is from `main/models.py`, defining the Field class. Lines 83 to 86 documents the class. This *docstring* will be extracted using Django admindocs (see section 4.10.1). Line 87 to 89 defines string fields with a limited amount of characters, while line 90 is a foreign key to another class. Line 92 and 93 defines how the class should be represented in for example the admin panel of Django. For more information on how to write Django models, see the Django documention on Models [27].

```
82 class Field(models.Model):
83     """
84     Every oil field belongs to a :model:`main.Country`, and contains
85     multiple :model:`main.Platform`.
86     """
87     name = models.CharField(max_length=30)
88     short_name = models.CharField(max_length=6)
89     description = models.CharField(max_length=100)
90     country = models.ForeignKey(Country)
91
92     def __unicode__(self):
93         return self.name
```

6.4.5 Unit Tests

Unit tests are located in `module/tests.py`. For this project, testing has not been deemed worth the time, as the wellwis project is far from production and what we have done is mostly a prototype. For more information on testing, see Testing in Django [30]

6.4.6 Folders

The folders are shown in figure 6.1. The main django folder (1) contains only the manage.py file, which is used for many management operations. For more information on that, see Django documentation [22].

Subapplication structure

Each of the created modules (2) has the core functionality of views, models, urls and tests as mentioned. In addition, an admin.py file controls interaction from the app to certain administrative tools. For more information on that, see Chapter 6 in the Djangobook [33].

Main application structure

The main applications (3) uses tests, urls and views in the same way as other application (see sections above). In addition there is a settings.py file which defines settings for the server. For more information on this file, see the *Django documentation - settings* [24]. We have also created some custom ajax files to separate out methods that responds to AJAX requests, which are called from different views files. Templates which are rendered from views files are located under the templates.

Static

The static folder (4) is a folder that is references from templates using `STATICURL` tag. This contains static files like images, css and javascript files.

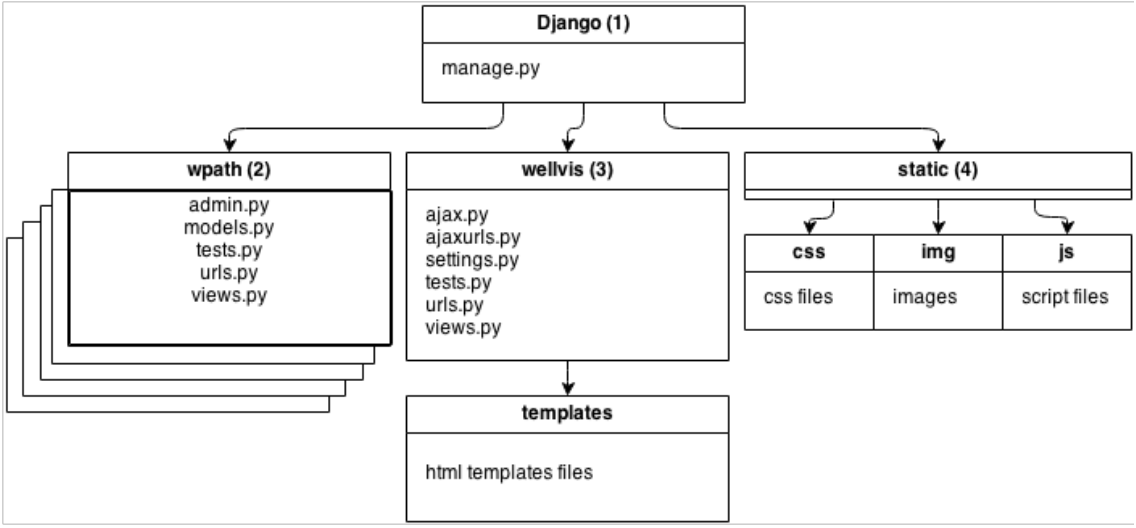


Figure 6.1: The folder and file structure within the solution

6.4.7 WebGL elements

The WebGL elements are shown and used while planning Well-path. The template file representing this ability is located in `wpath/templates/wpath/3D_View.html`. `3D_View.html` contains the main HTML DOM structure and its supporting CSS directives for the Well-Path module. It contains some JavaScript methods that would directly interact with the HTML DOM structure. It includes JavaScript libraries and files from `wellvis/static/js` and `wellvis/static/js/3D`. Files under `wellvis/static/js` are different open source JavaScript libraries that are used for building good User Interface and help in easier interation with the HTML DOM structure. While files under `wellvis/static/js/3D` are open source 3D libraries and files, plus files that are part of Well-Path module. `Three.js` is the main library for the 3D graphics and has the support for WebGL. `Detector.js` basically detects if the graphic card and the web-browser supports WebGL.

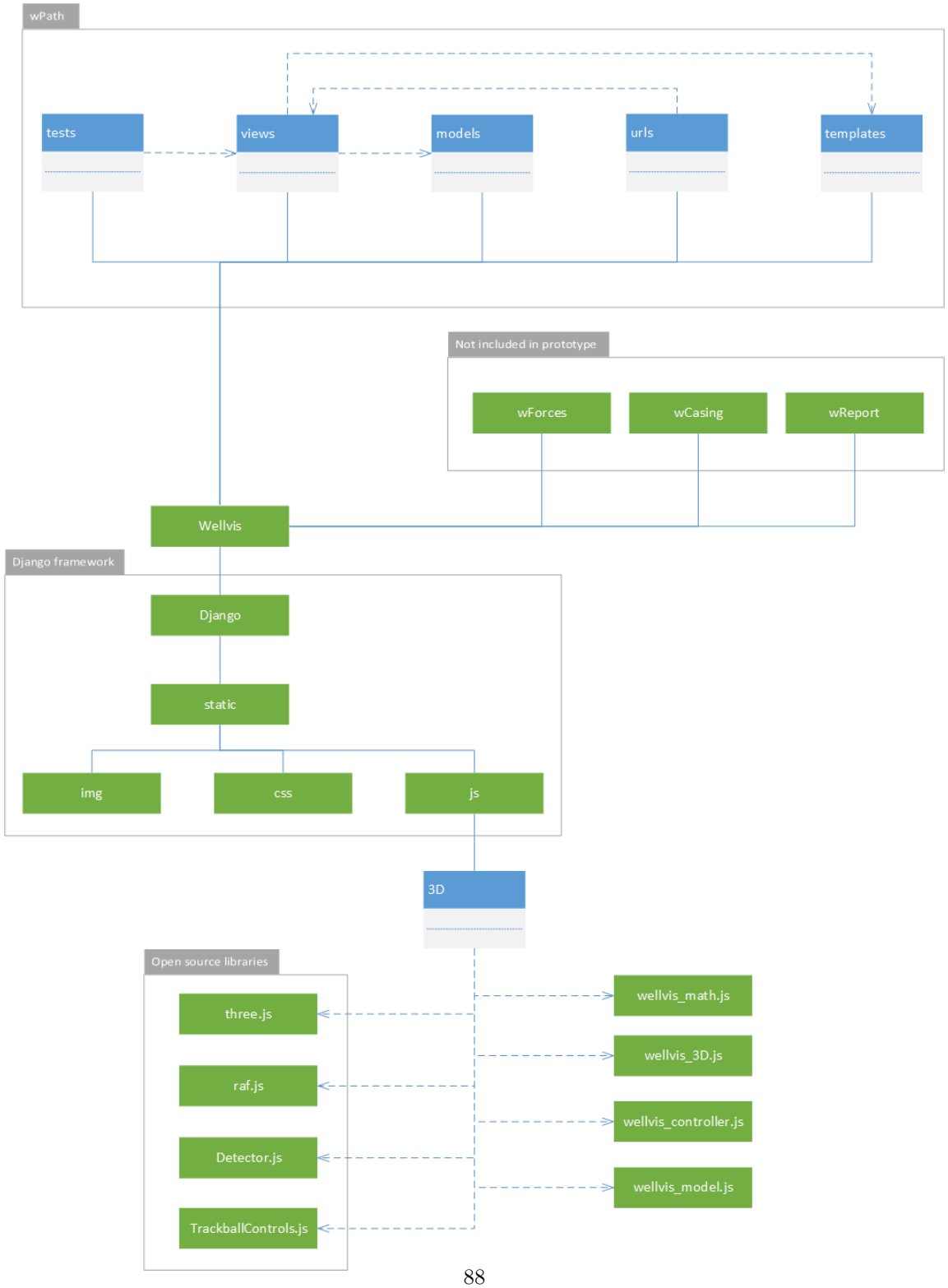


Figure 6.2: Class diagram

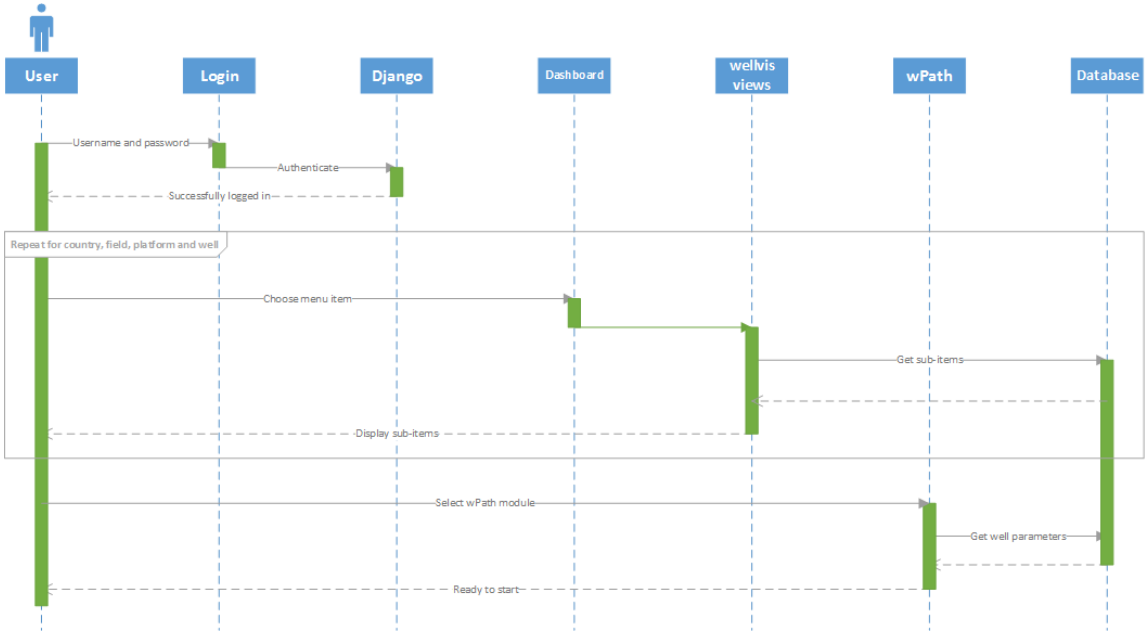


Figure 6.3: Sequence diagram

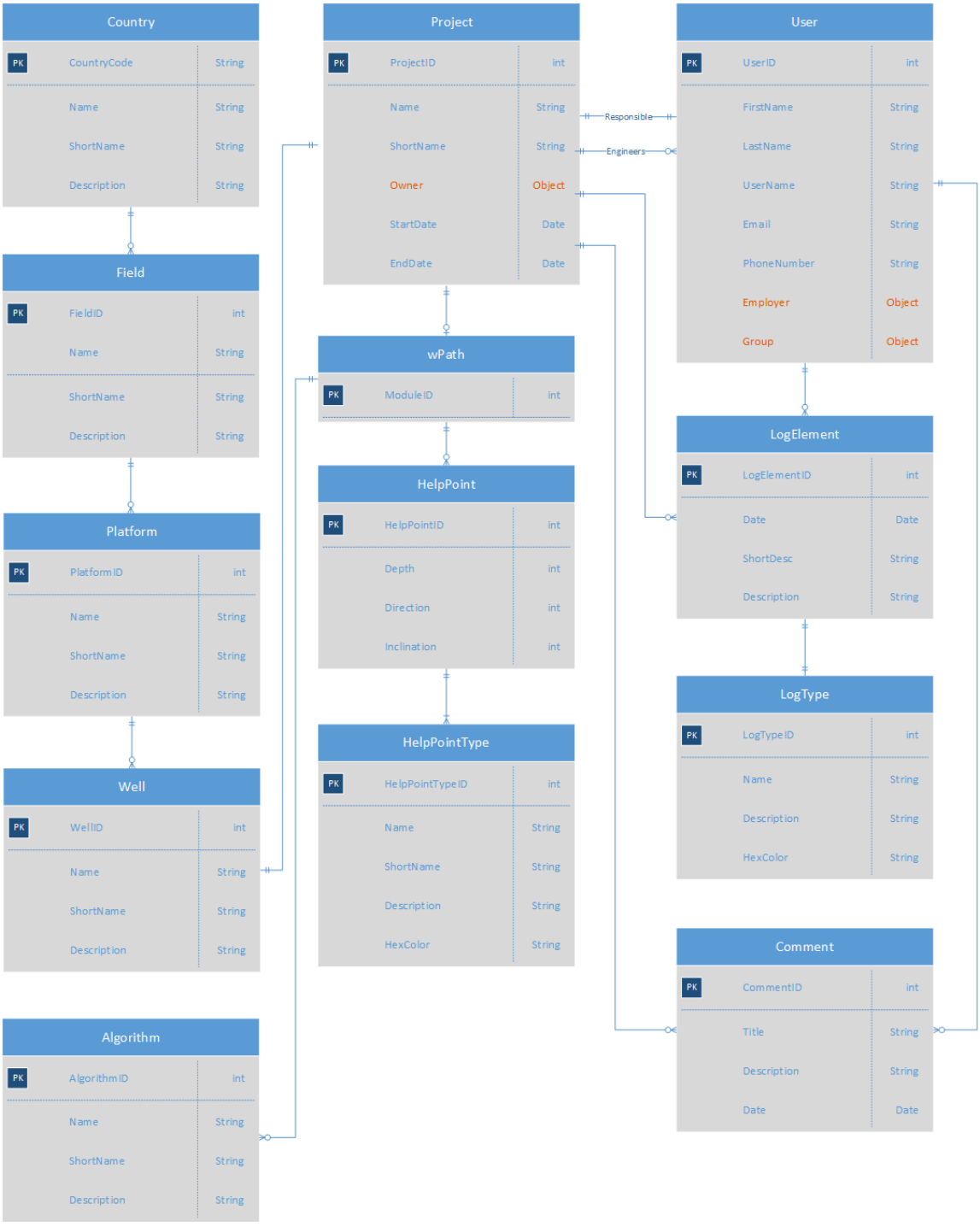


Figure 6.4: Entity relationship diagram

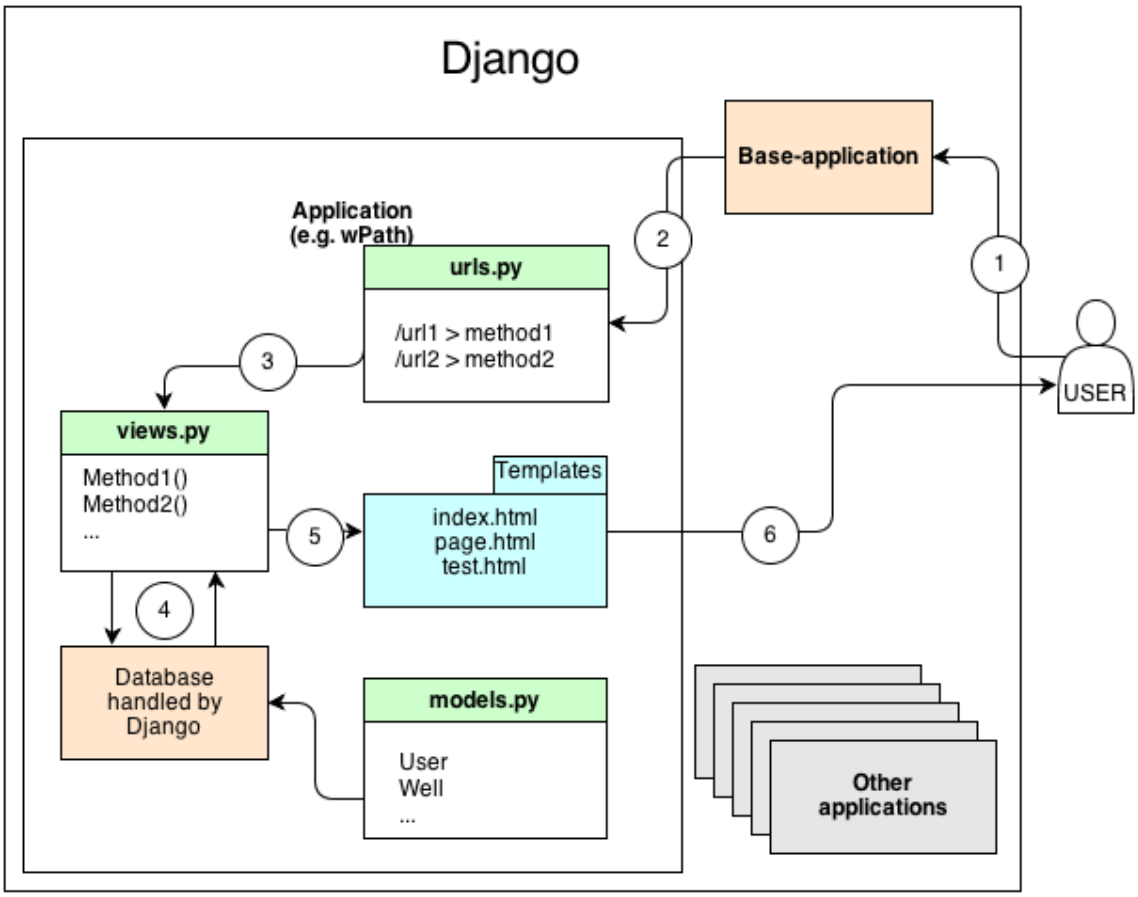


Figure 6.5: How Django processes a request. A user requests a specific URL (1), which django translates to a specific method (2 & 3). The method gathers the responding data to fullfil the request (4), and (usually) sends that data to the user through an HTML template (5 & 6)

Part II

Sprints

Chapter 7

Sprint 1

In this chapter we discuss sprint 1 and how it went. We planned the sprint together with our customer, and since this was the first sprint we had to test out a lot and see what worked and what did not work. We start of with how the sprint planning was in 7.1 with our goals and the duration of the sprint. Later in the chapter we have our work breakdown in section 7.2 before we continue with our system design for our requirements in 7.3. We also discuss what tests we have done and how they went in 7.4. In the end of this chapter we evaluate sprint 1, 7.5 and see what we can learn till next sprint. We also have a very short summary in the end, section 7.6, where we mention how many code lines we have added and what was good and bad in this sprint.

7.1 Sprint Planning

In this section we will discuss how this sprint was planned. We will discuss the duration of the sprint in 7.1.1 and our goals in 7.1.2. We discussed everything with our customer before deciding the sprints.

7.1.1 Duration

The first sprint has a duration from september 9th until september 27th, when the first prototype is to be revealed. Our goal is to finish a week before the deadline so that we can use the last week to fix bugs and test the system. We have weekly meetings with our advisor, but we will not have weekly meetings with our customers since our main contact is a lot away on job, and since we feel like we have the information we need, and know that our customers trust us.

7.1.2 Sprint Goal

The goal for this sprint is to set up server, development environment, virtual control and implement basic functionality. A user should be able to log into web interface and select different modules, and different wells, as specified in the backlog (Section 7.2.1. A simple path representation should be able to be created and viewed. In order to do this, we need to focus on evaluating and selecting different technologies in the preliminary studies (chapter 4).

7.2 Work breakdown

In this section we will discuss the work breakdown and show some important figures for knowing how the work has been done in this sprint. We start with the backlog in 7.2.1 and we continue with the work breakdown structure in section 7.2.2 and in the end we have the burndown, 7.2.3.

7.2.1 Backlog

All tasks were nicely estimated, and no major issues occured during the implementation phase of these.

ID	Description	Estimated	Actual
F1	Login	42	37
F2	Select module	5	8
F3	List existing wells	13	12
F4	Choose a well path	3	5
F5	Insert well path data	13	18
F6	Visualize well path data	42	45
Sum		118	125

Table 7.1: Backlog for sprint 1

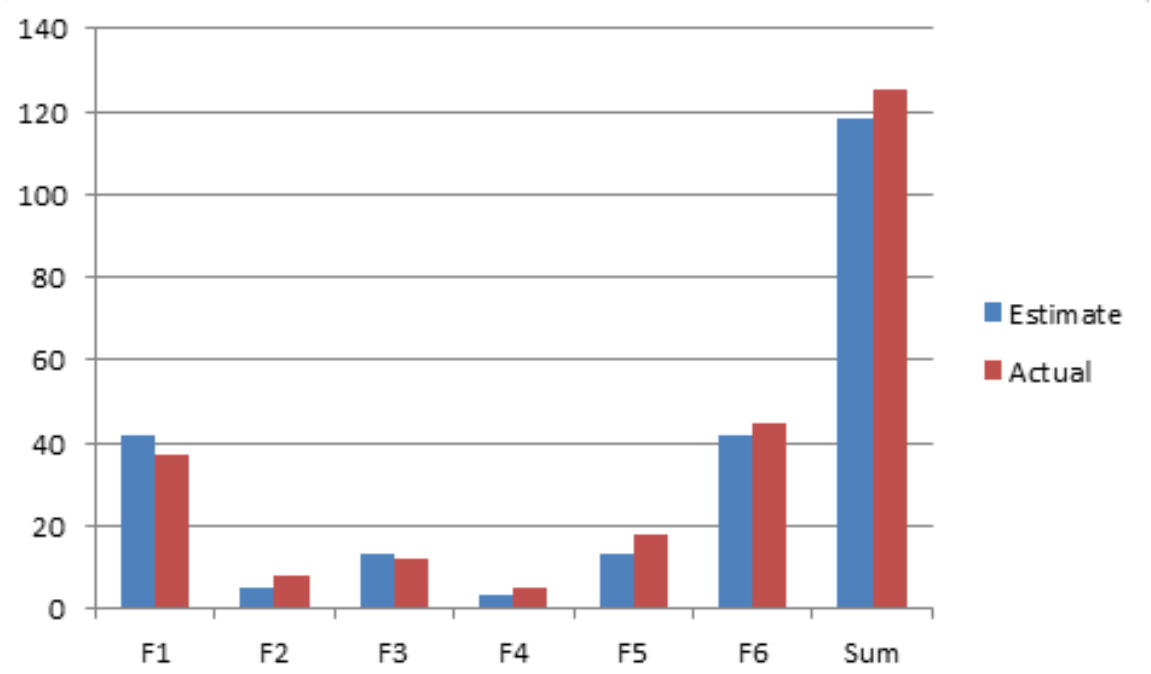


Figure 7.1: Sprint 1 estimations compared to actual time spent. Actual work ended up close to the estimated time.

7.2.2 Work breakdown structure

Work Breakdown structure is illustrated in figure 8.2 and described in further detail below.

Backend

In order to set up the solution, we need to set up the backend framework on a server, version control, database and dummy data.

1. Install web framework
2. Setup git repository
3. Setup virtual environment
4. Create and setup database models

5. Create users and dummy data

Frontend

In order to view and use the solution, we need to be able to log in, navigate different modules and select different wells.

1. Choose module: Selecting between different module, such as wPath.
   (a) Responsive CSS
   (b) Create theme
   (c) Make menu for login and module selection
2. List and choose wells: There should be a specific menu for navigation and selection of wells.
   (a) Create well menu
   (b) Populate menu from database
   (c) Create well-info page
3. Login functionality: A page is need for the user to log in, so he can use the system.
   (a) Authentication system
   (b) Create login page

3D Well

1. Insert well path: In order to create a well path, user must be able to insert data regarding the path.
   (a) Create panel for insertion
   (b) Store insertion to JSON
   (c) Read values from JSON
2. 3D view of well: In order to create a well path, user must be able to insert data regarding the path.

(a) Create panel for insertion

(b) Store insertion to JSON

(c) Read values from JSON

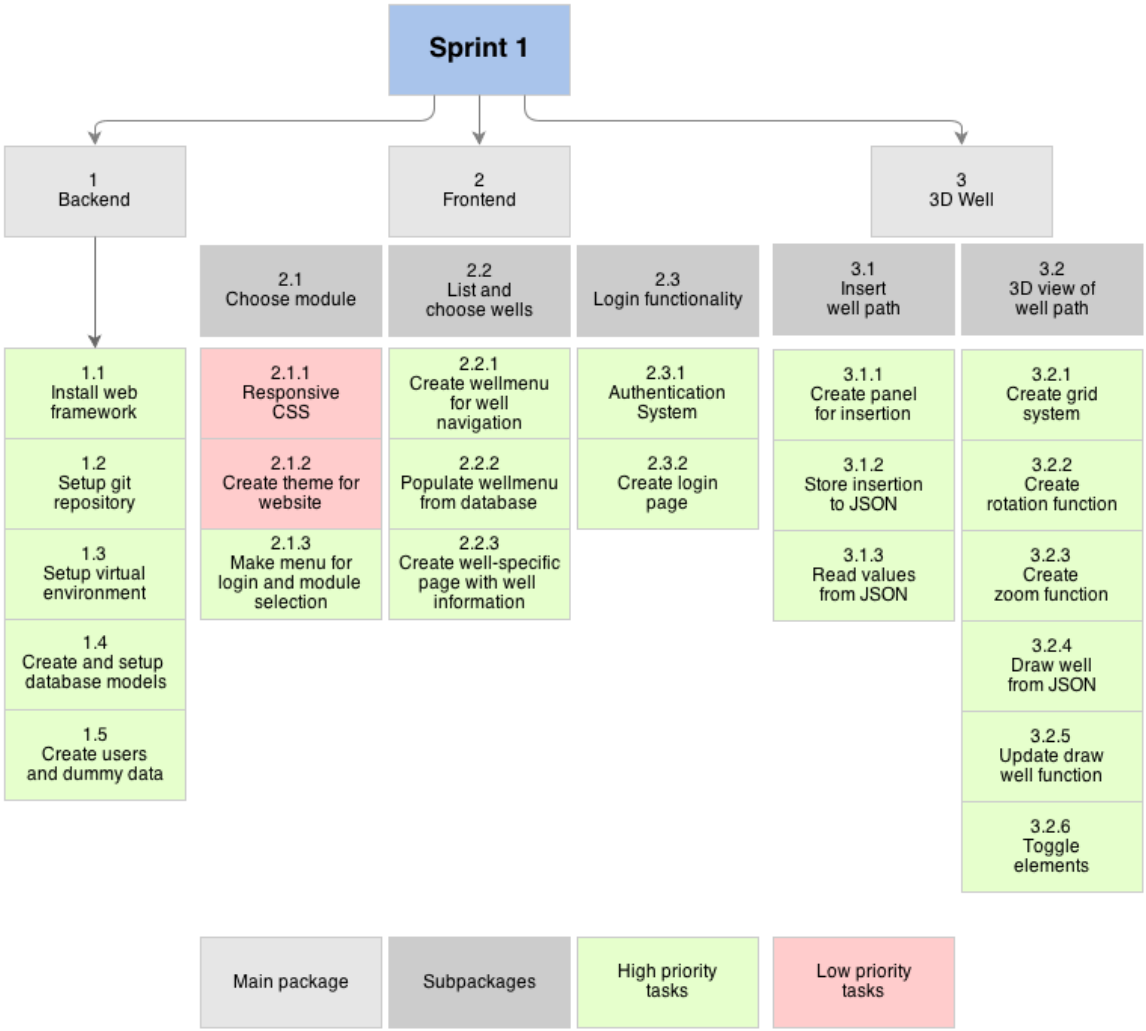


Figure 7.2: Sprint 1 Work Breakdown Structure. Task 1 includes backend functionality to set up the solution environment. Task 2 corresponds to requirement F2, F3+F4 and F1 respectively. Task 3 corresponds to requirement F5 and F6.

7.2.3 Burndown

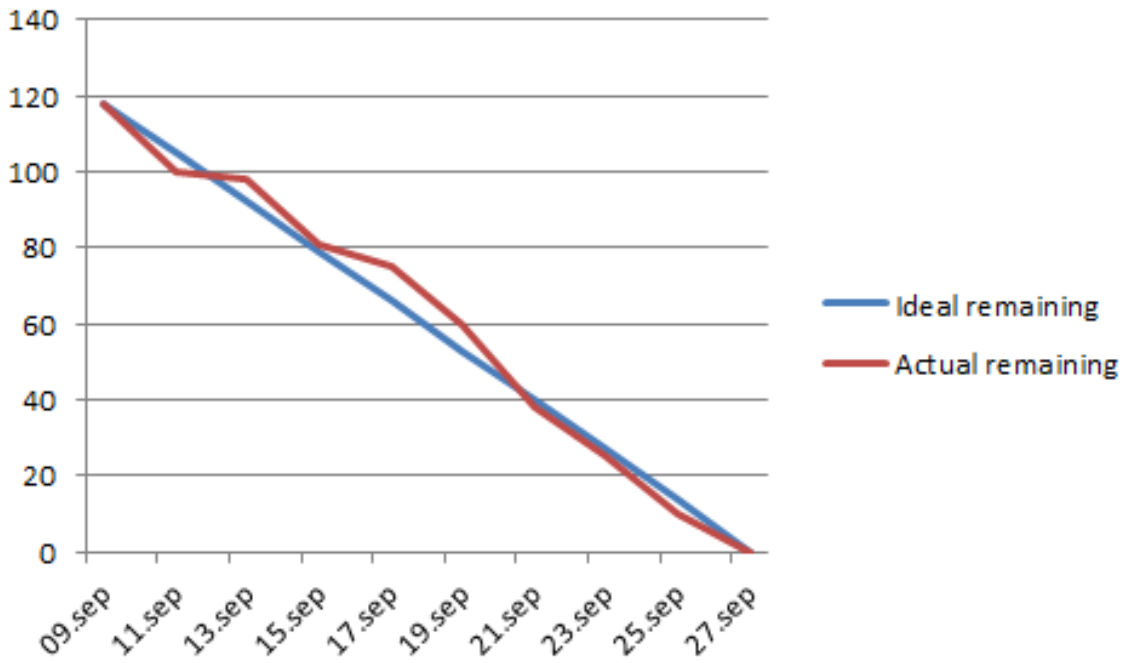


Figure 7.3: Sprint 1 Work Breakdown Structure.

7.3 System Design

We will in this section describe design for each funcional requirement by specifying the files and their role in fulfilling that requirement. In addition, a figure will show the flow of interactions for that case.

7.3.1 F1 - Login

Files and location

ID	Location	Purpose
F1-1	`/wellvis/urls.py`	Enables django admin core functionality and redirects link.
F1-2	`(Django core)`	Processes and authenticates, serves built-in templates.

Table 7.2: Files and location for F1

Interaction

The user loads url defined in F1-1 that is then handled by django core functionality to serve user a login page. The user then authenticates himself again through F1-1 and subsequentially the django core, which then serves him or her an admin page.

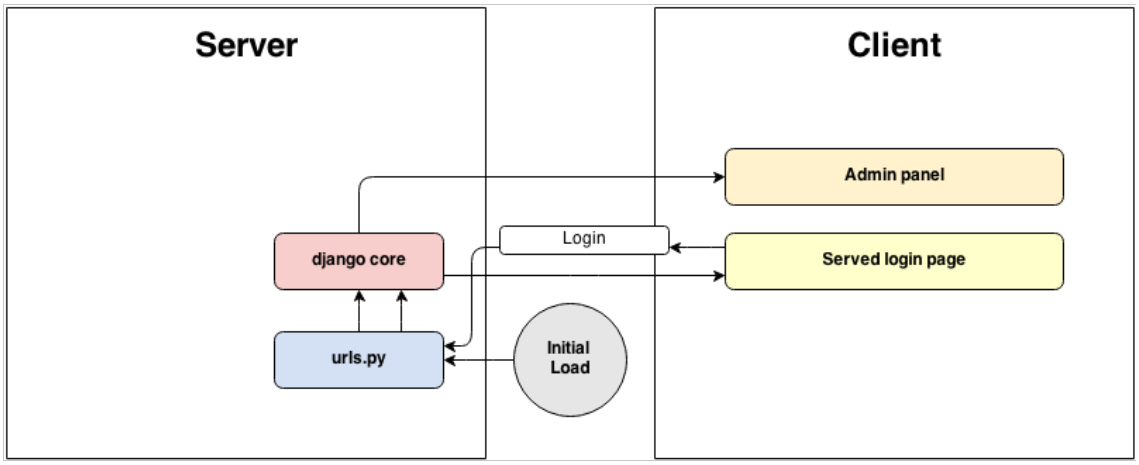


Figure 7.4: The design of interaction flow for F1.

7.3.2 F2 - Select module

Files and location

ID	Location	Purpose
F2-1	`/wellvis/urls.py`	Connects urls to responding methods.
F2-2	`/wellvis/views.py`	Contains responding methods, building a response.
F2-3	`/wellvis/templates/base/main.html`	Base template to serve.
F2-4	`/wellvis/templates/base/menu.html`	Template part containing menu.

Table 7.3: Files and location for F2

Interaction

User first requests URL from F2-1 serving an arbritrary page through F2-2 that extends F2-3. F2-3 includes F2-4 which contains the elements used to select module. When user is given this response, he should be able to select that element, and subsequentially load the corresponding module through the same procedure.

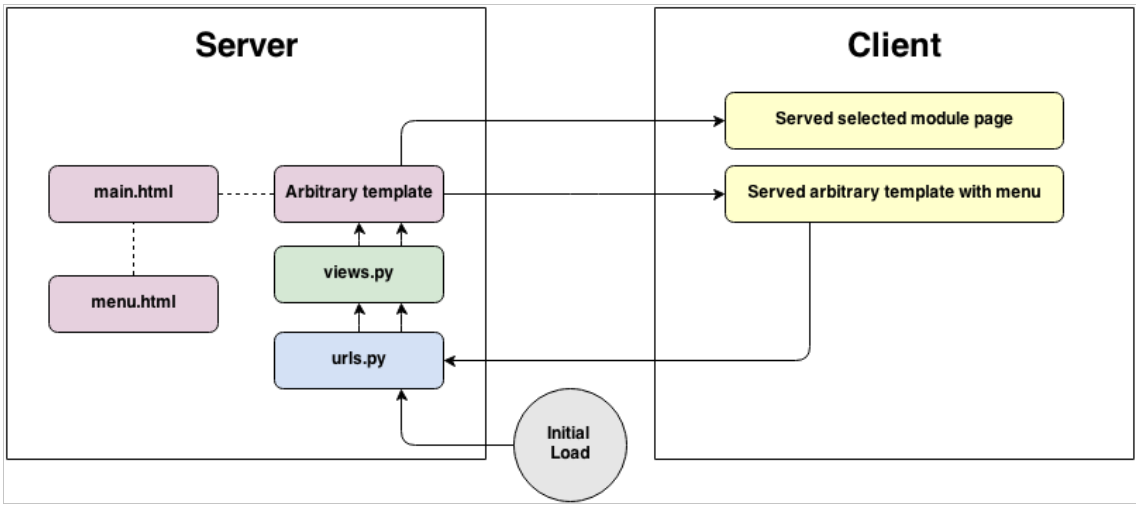


Figure 7.5: The design of interaction flow for F2.

7.3.3 F3 - List existing wells

Files and location

ID	Location	Purpose
F3-1	`/wellvis/urls.py`	Connects urls to responding methods.
F3-2	`/wellvis/views.py`	Contains responding methods, building a response.
F3-3	`/wellvis/templates/base/main.html`	Base template to serve.
F3-4	`/wellvis/templates/base/menu.html`	Template part containing menu.
F3-5	`/static/js/sidepanel.js`	Lets user browse well catalog.

Table 7.4: Files and location for F3

Interaction

User first requests URL from F3-1 serving an arbitrary page through F3-2 that extends F3-3. F3-3 includes F3-4 which contains the elements used to select module. When user is given this response, he should have a complete list of all wells, and parent platforms, fields and countries. He navigates through the hierarchy using F3-5.

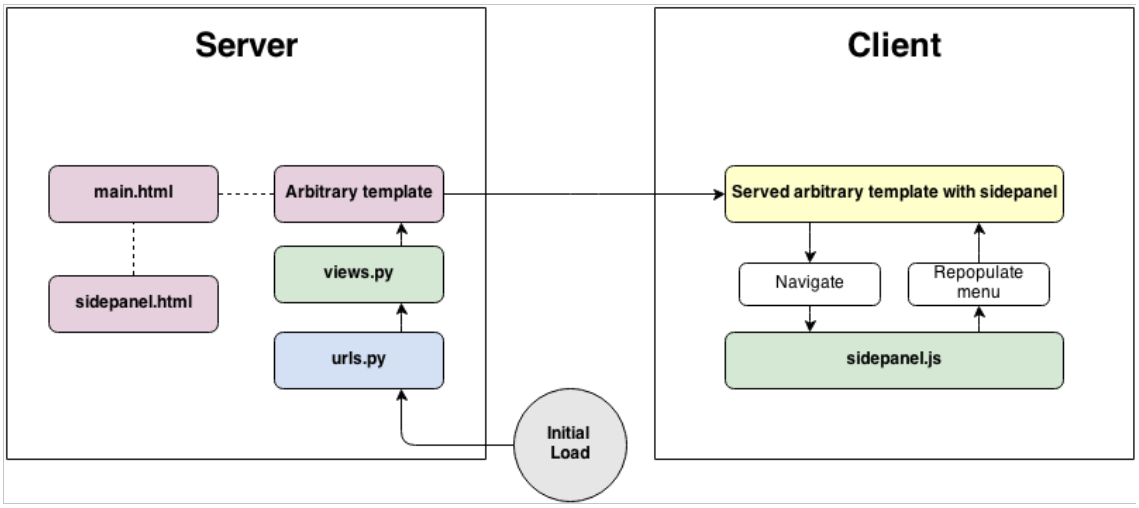


Figure 7.6: The design of interaction flow for F3.

7.3.4 F4 - Choose a well path

Files and location

ID	Location	Purpose
F4-1	`/wellvis/urls.py`	Connects urls to responding methods.
F4-2	`/wellvis/views.py`	Contains responding methods, building a response.
F4-3	`/wellvis/templates/wellvis/well.html`	Base template to serve.
F4-4	`/static/js/sidepanel.js`	Lets user browse well catalog.

Table 7.5: Files and location for F4

Interaction

Given that a user is already on a page containing the sidepanel, he should have a complete list of all wells, parent platforms, fields and countries. He navigates through the hierarchy using F4-4, and once he selects a specific well, a request for that well is being sent to F4-1. This will forward the request to F4-2, which builds the response from the database. Data will be forwarded F4-3, which sends user a finished response contiaing the requested well details.

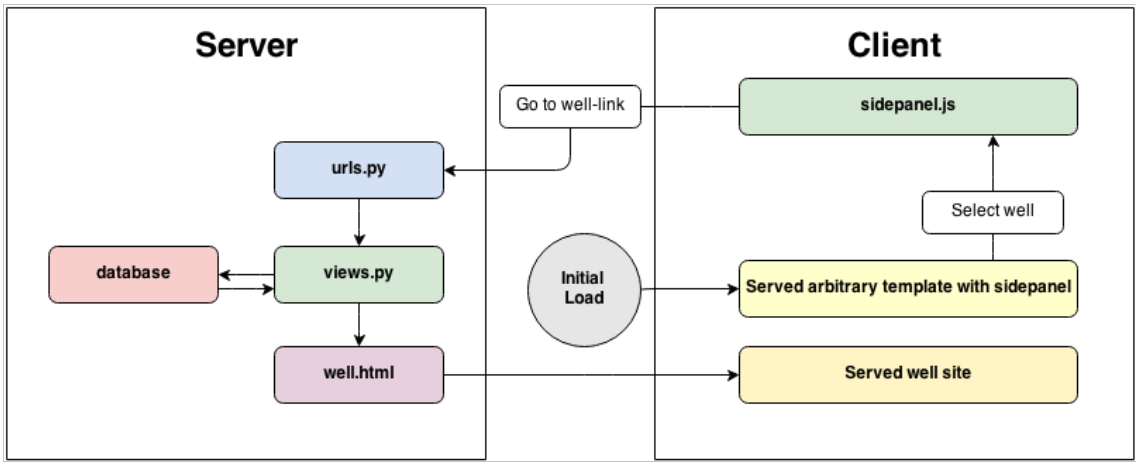


Figure 7.7: The design of interaction flow for F4.

7.3.5 F5 - Insert well path data

Files and location

ID	Location	Purpose
F5-1	`/wpath/templates/wpath/3D_View.html`	Serve 3D path elements
F5-2	`/static/js/3D/wellvis_controller.html`	Control user interaction with 3D path
F5-3	`/wellvis/ajax.py`	Logic behind save/load user 3D config file
F5-4	`/wellvis/ajaxurls.py`	Define urls for client-server comm.
F5-5	`/static/js/ajax-proxy.js`	Handle client-server comm. client-side
F5-6	`/static/js/jquery-growl.js`	Handles user notifications

Table 7.6: Files and location for F5

Interaction

Given that the user already has loaded page F5-1, an insertion of well point will trigger F5-2 which will store and update the path locally, as well as sending an update to the server through F5-5 and F5-4 before F5-3 stores the new path in database. F5-3 will notify F5-2 of the status of save, which might trigger F5-6 to notify the user in case of failure.

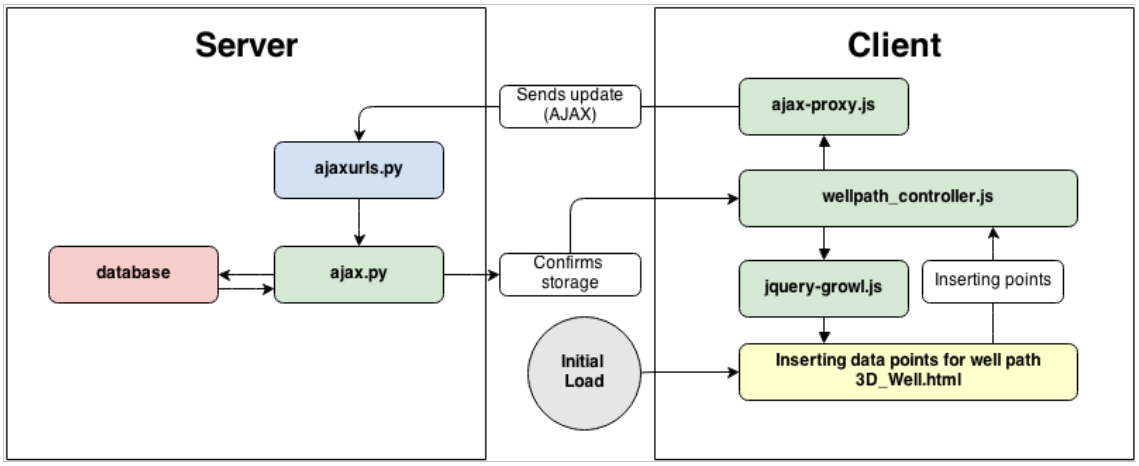


Figure 7.8: The design of interaction flow for F5.

7.3.6 F6 - Visualize well path data

Files and location

ID	Location	Purpose
F6-1	`/wpath/templates/wpath/3D_View.html`	Serve 3D path elements
F6-2	`/static/js/3D/wellvis_controller.js`	Control user interaction with 3D path
F6-3	`/static/js/3D/three.js`	Generates 3D elements

Table 7.7: Files and location for F6

Interaction

Given that a F6-1 is loaded, and user interacts with the 3D elements (eg. attempting to rotate or inserting new data points), F6-2 picks this up and generates new graphics through F6-3.

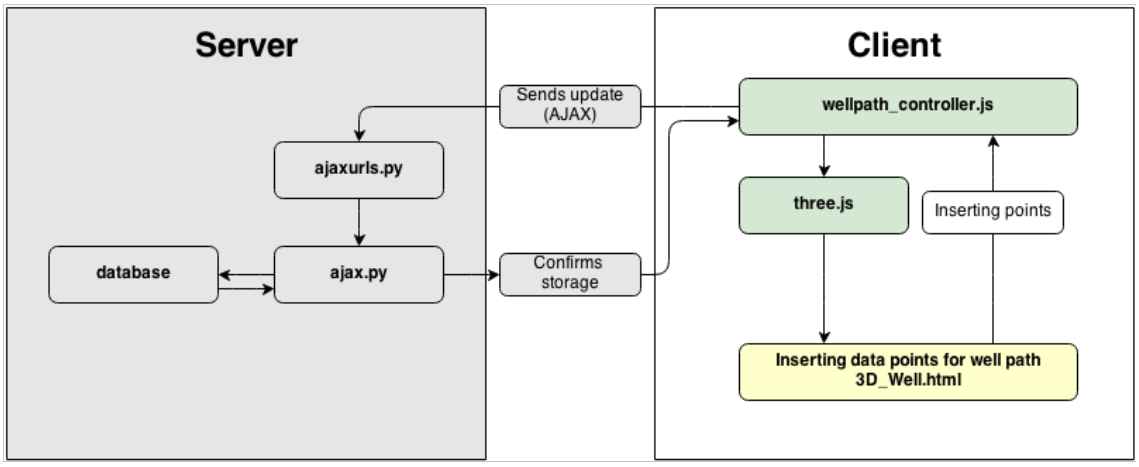


Figure 7.9: The design of interaction flow for F6.

7.4 Sprint Testing

This section is about how the testing for sprint 1 went. In 7.4.1 we show our test results and in 7.4.2 we evaluate how the testing went.

7.4.1 Test Results

During the sprint the team executed a total of six test cases. Here is an example of a test case that has been run in this sprint and the rest of the test cases can be found in Appendix F.

ID	ID01
Description	Log in
Precondition	Must have username and password
Feature	Test that it is possible to log in
Execution	1. Write in username and password 2. Press Login
Expected Result	User should now be logged in to the system

Table 7.8: Test case ID01

ID	ID01
Description	Log in
Tester	Tomas
Date	2013-09-24
Result	Success

Table 7.9: Test result for ID01

ID	ID02
Description	Add element
Tester	Tomas
Date	2013-09-26
Result	Success

Table 7.10: Test result for ID02

ID	ID03
Description	Navigate
Tester	Tomas
Date	2013-09-26
Result	Success

Table 7.11: Test result for ID03

ID	ID04
Description	Hide navigation
Tester	Tomas
Date	2013-09-26
Result	Success

Table 7.12: Test result for ID04

ID	ID05
Description	Insert well path points
Tester	Tomas
Date	2013-09-27
Result	Success

Table 7.13: Test result for ID05

ID	ID06
Description	Remove well path points
Tester	Tomas
Date	2013-09-27
Result	Success

Table 7.14: Test result for ID06

7.4.2 Test evaluation

All of our tests in this first sprint succeded. We were very clear from the beginning on how to test and we were good making sure that our code was testable. The only thing that took time with the tests was that the code needed to be finished before testing, which meant that some tests was not done until the last day.

7.5 Sprint Evaluation

7.5.1 Customer feedback

The customer seemed very happy with what we had managed to do in the first sprint. They especially liked the fact that you had live-update from input data onto the 3D-model, which ran smoothly.

- "Already better than the existing tool" – Bjørn about the 3D module, 2013-09-29
- "Incredible what you've been able to do in so short amount of time" – Bjørn, 2013-09-29
- "User interface is modern and snappy" – Øyvind, 2013-09-30
- "Useful to see each step and not only the finished path" – Joakim, 2013-09-30
- "Grid should be less visible. Would also be nice to change colors." – Øyvind, 2013-09-30
- "I think it would be nice with a denser well-path" – Sigbjørn, 2013-09-30

7.5.2 Positive Experiences

The frameworks used impressed us. They make it simple and easy to create solid functionality in a short amount of time. No larger technical problems were encountered, and alot were created in only three short weeks.

- Smooth progress
- Twitter Bootstrap framework
- Three.js framework
- Django framework

7.5.3 Negative Experiences

Poor preplanning and task specification made it hard to start working if you did not know what to do. *What to do now?* was a question that appeared too often. Once developers were set to program, they played around and implemented tasks with low inclusion of the rest of the group. This made it even harder to join developing for the other team members if they wanted. Neither was it easy to document on parts that involved implementation for non-developing members.

- Low inclusion
- Imprecise task specification
- Imprecise task delegation

7.5.4 Planned Actions

Better sprint planning

Better specification of sprint tasks in project task tool should be done before implementation, to easier see what can and will be done.

Documenting in Parallell while Implementing

Developers should document in parallell while implementing. This would ensure a steadier documentation progress, written by those that know how implementation was done.

Split Coding and Report Writing between team members

Go back and define more clearly who got responsiblity of what, and divide the tasks so everyone knows what they're suppose to have control over.

7.6 Summary

- Added 15 521 lines of Open Source code.
- Added 1 075 own lines of code.
- All planned tasks implemented
- Good estimations
- Bad task delegation and specification

Chapter 8

Sprint 2

In this chapter we discuss sprint 2 and how it went. We planned the sprint together with our customer after sprint 1, and since this was the second sprint we had learned a lot from the first and had seen some of what worked and what did not work. We start of with how the sprint planning was in 8.1 with our goals and the duration of the sprint. Later in the chapter we have our work breakdown in section 8.2 before we continue with our system design for our requirements in 8.3. We also discuss what tests we have done and how they went in 8.4. In the end of this chapter we evaluate sprint 1, 8.5 and see what we can learn till next sprint. We also have a very short summary in the end, section 8.6, where we mention how many code lines we have added and what was good and bad in this sprint.

8.1 Sprint Planning

8.1.1 Duration

The second sprint has a duration from September 30th until October 21th. What we will do is decided during our presentation for the customer after sprint 1 to see if some requirements might have changed. Our goal is to finish a week before the deadline so that we can use the last week to fix bugs and test the system. We have weekly meetings with our advisor, but we will not have weekly meetings with our customers since our main contact is a lot away on job, and since we feel like we have the information we need, and know that our customers trust us.

8.1.2 Sprint Goal

The goal for this sprint is to implement core functionality, such as modifiable graphics, adding of curvature methods and user friendly save and load methods between server and client through AJAX. Hopefully, we will be able to use the whole next sprint for adjusting and finishing then existing in collaboration with customer.

8.2 Work breakdown

In this section we will discuss the work breakdown and show some important figures for knowing how the work has been done in this sprint. We start with the backlog in 8.2.1 and we continue with the work breakdown structure in section 8.2.2 and in the end we have the burndown, 8.2.3.

8.2.1 Backlog

Due to the heavy underestimation of tasks F17-F19, other tasks was cancelled. Task F19 was continued in sprint 3.

ID	Description	Estimated	Actual
F7	Distinction between end- and help-targets	8	N/A
F12	Insert geological data and no-go areas	42	N/A
F15	Import data from existing wells	13	N/A
F16	Warning on intersecting paths	8	N/A
F17	Configurable 3D view	8	32
F18	Automatic load/save of graph	21	45
F19	Create curvature for paths	42	80+
Sum		142	157

Table 8.1: Quality requirements

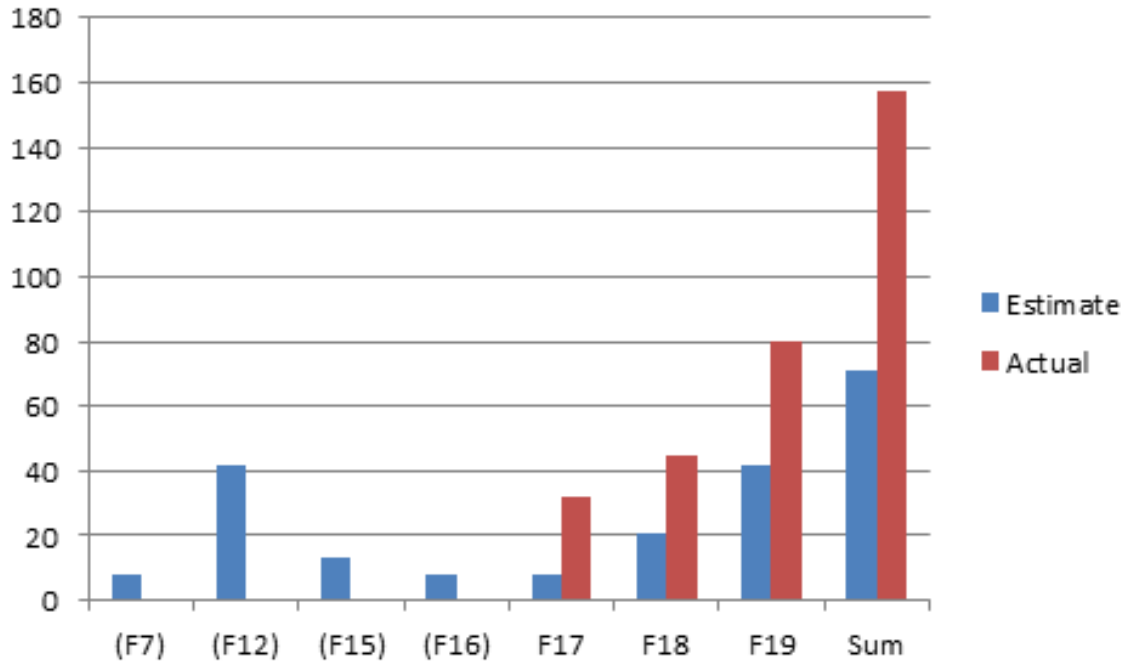


Figure 8.1: Sprint 2 estimations compared to actual time spent. F7, F12, F15, F16 was cancelled after seeing the large misestimations for the more important tasks. F19 was not finished and continued in sprint 3.

8.2.2 Work breakdown structure

Work Breakdown structure is illustrated in figure 8.2 and described in further detail below.

3D config

A user should be able to change his personal settings regarding the different colors, thickness and opacity of graphic elements, so he can match the interface with his personal preferences.

1. Defining variables
2. Reading settings from JSON
3. Implement settings model
4. JSON settings needs to be sent queried out of the DB, and sent to the template

5. Module to support settings needs to be installed in Django
6. Configure settingspage
7. JS Color picker
8. 3D graph + AJAX storage of settings

Save/load 3D graph

A graph needs to be retrieved and saved to the database, so a user does not have to change settings for each time he refreshes the page. This should be saved with AJAX, so that the user does not have to reload page.

1. Define JSON structure
2. Read from JSON structure
3. Send graph JSON to template
4. AJAX recieve of graph
5. AJAX send of graph

Target types (low priority)

Additional points in the graph of type no-go should be implemented, so that the user can see where he should not drill.

1. Update JSON structure
2. Update read from structure
3. Update DB model
4. Update template generation
5. Implement display of no-go points

XYZ to MD+

The user should be able to enter well point data using MD, azimuth and inclination, to make it easier to create a good well path.

1. Update DB model
2. Calculate x, y, z positions from new input
3. Update JSON
4. Update template generation
5. Update JSON after 3D curvature impl.

3D curvature

The well should be able to project realistic curvatures between points, to minimize offset from realistic path and chances of creating a bad well path.

1. Calculate position of intermediate points
2. Update JSON structure to facilitate intermediate points
3. Drawing of intermediate points
4. Update DB model to facilitate intermediate points
5. Update template generation for intermediate points
6. AJAX: Recieve intermediate points actionss
7. AJAX: Send intermediate points actionss

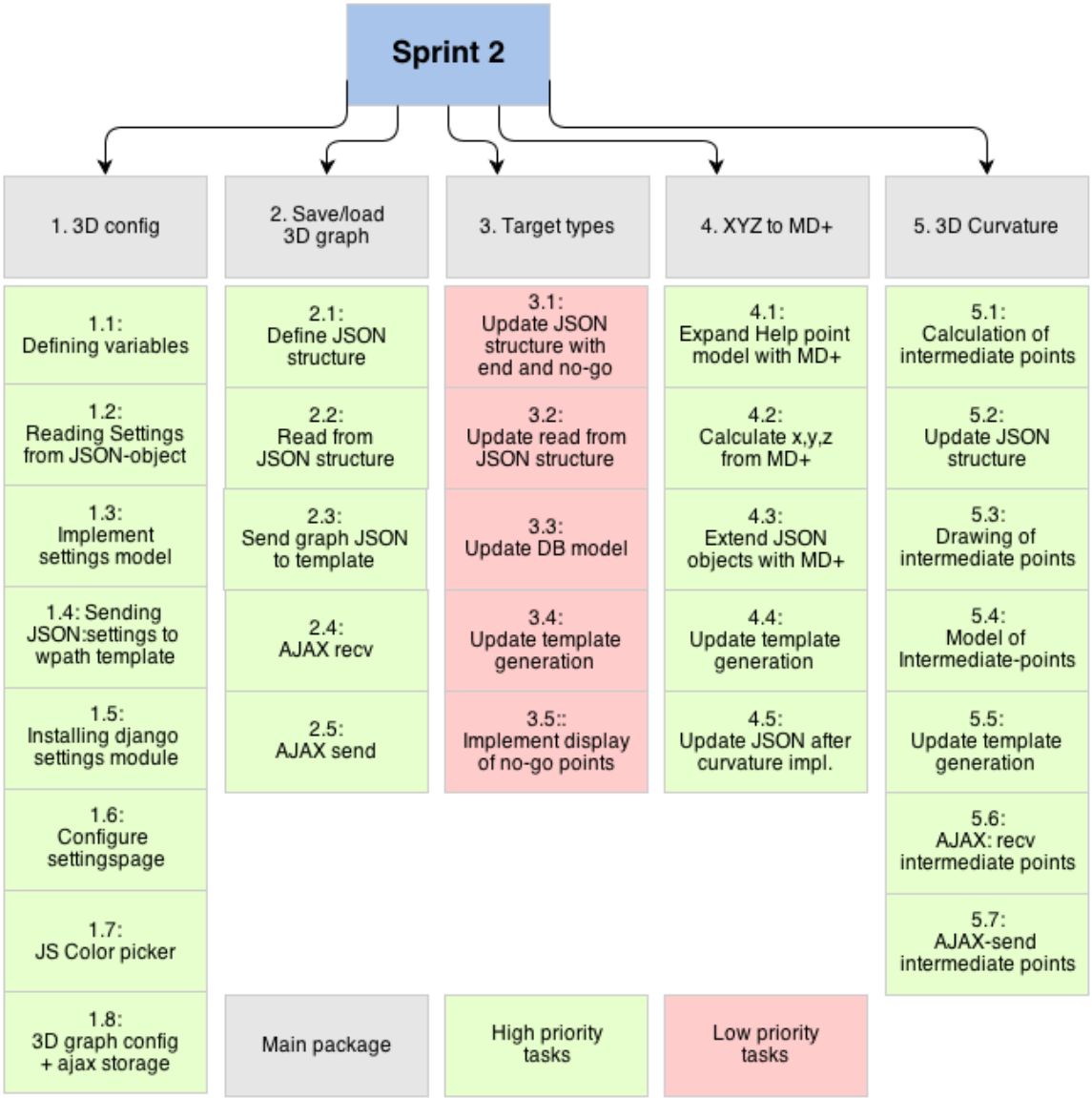


Figure 8.2: Sprint 2 Work Breakdown Structure.

8.2.3 Burndown

Sprint 2 burndown has a large offset between actual remaining time and ideal remaining time. This is due to the realization of huge underestimates of the main requirements for the sprint (see figure 8.1). Around 20. october, we decided to cancel several of the tasks, cutting down the remaining

time from 110 to 42 before the sprint ended. (see figure 8.3).

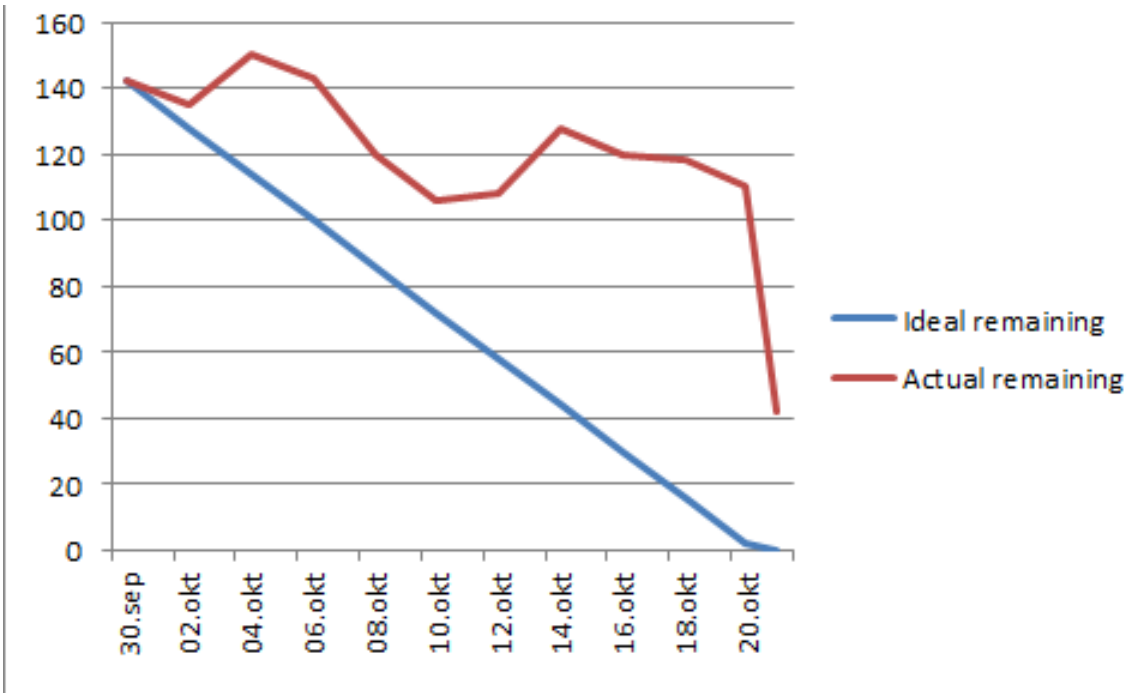


Figure 8.3: Sprint 2 Burndown chart.

8.3 System Design

We will in this section describe design for each funcional requirement by specifying the files and their role in fulfilling that requirement. In addition, a figure will show the flow of interactions for that case.

8.3.1 F17 - Configurable 3D view

Files and location

ID	Location	Purpose
F17-1	`/main/models.py`	To connect a JSON field to a user
F17-2	`/wellvis/ajax.py`	Logic behind save/load user 3D config file
F17-3	`/wellvis/ajaxurls.py`	Define urls for client-server comm.
F17-4	`/static/js/ajax-proxy.js`	Handle client-server comm. client-side
F17-5	`/static/js/3D/wellvis_controller.js`	Handle client interaction
F17-6	`/wpath/templates/wpath/3D_View.html`	Show client gfx elements

Table 8.2: Files and location for F17

Interaction

User interaction with DOM in F17-6 triggers F17-5 which will update gfx elements and request ajax calls to server through F17-4. This will subsequentially trigger F17-3, F17-2 to ensure updates serverside.

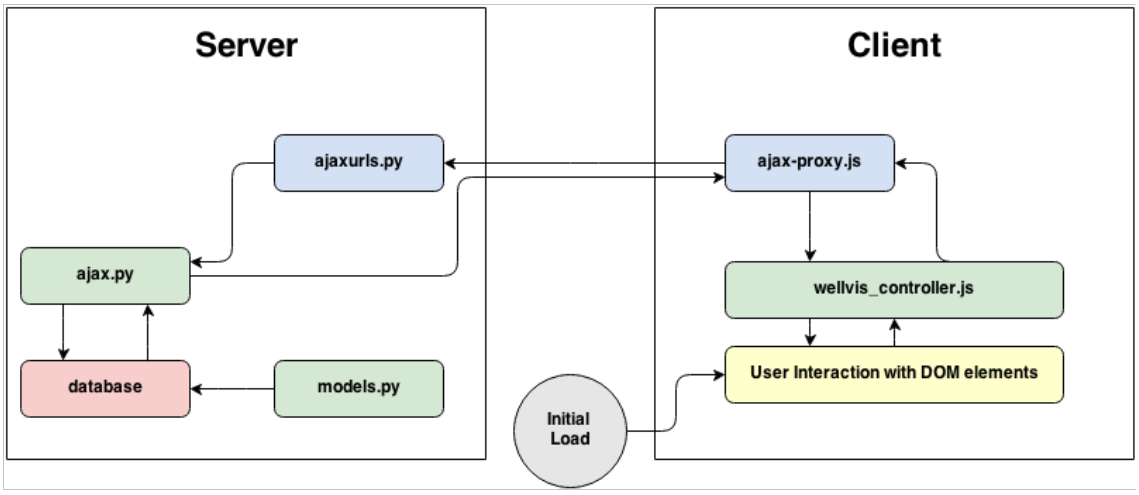


Figure 8.4: The design of interaction flow for F17.

8.3.2 F18 - Automatic load/save of graph

Files and location

ID	Location	Purpose
F18-2	`/wellvis/ajax.py`	Logic behind save/load path
F18-3	`/wellvis/ajaxurls.py`	Define urls for client-server comm.
F18-4	`/static/js/ajax-proxy.js`	Handle client-server comm. client-side
F18-5	`/static/js/3D/wellvis_controller.js`	Handle client interaction
F18-6	`/wpath/templates/wpath/3D_View.html`	Show client gfx elements

Table 8.3: Files and location for F18

Interaction

Interaction goes essentially the same way as saving/loading 3d config. The only difference is the use of different methods, saving to different objects in database etc. User interaction with DOM in F18-6 triggers F18-5 which will update gfx elements and request ajax calls to server through F18-4. This will subsequentially trigger F18-3, F18-2 to ensure updates serverside.

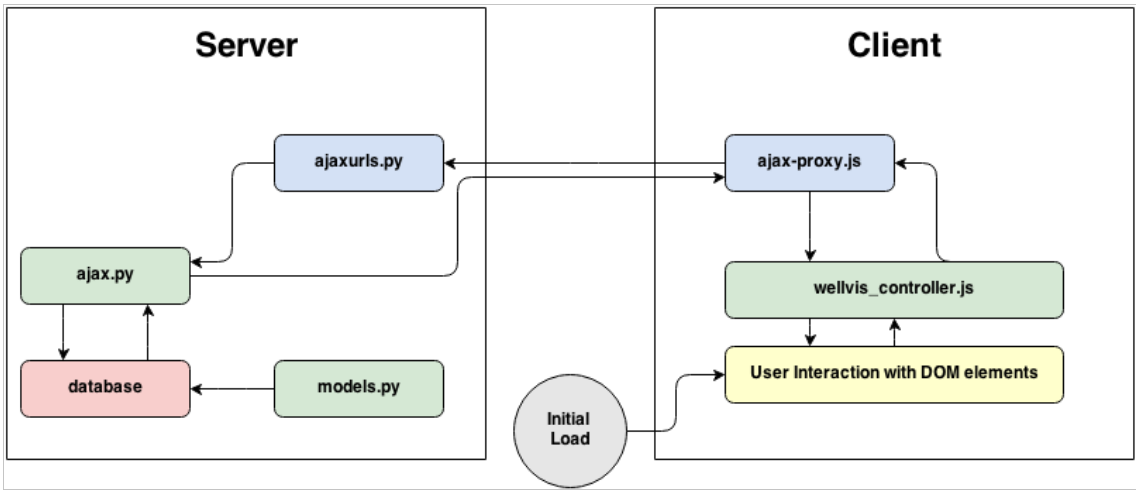


Figure 8.5: The design of interaction flow for F18.

8.3.3 F19 - Create curvature for paths

Files and location

ID	Location	Purpose
F19-1	`/static/js/3D/wellvis_controller.js`	Handle client interaction
F19-2	`/static/js/3D/wellvis_math.js`	Calculations intermediate points
F19-3	`/wpath/templates/wpath/3D_View.html`	Show client gfx elements

Table 8.4: Files and location for F19

Interaction

The new elements in F19-3 will trigger F19-1 when used, which will then call F19-2 to calculate intermediate points which then is returned and inserted into the wellpath data. Server calls are executed to ensure consistency with server, but is not considered a part of the requirement.

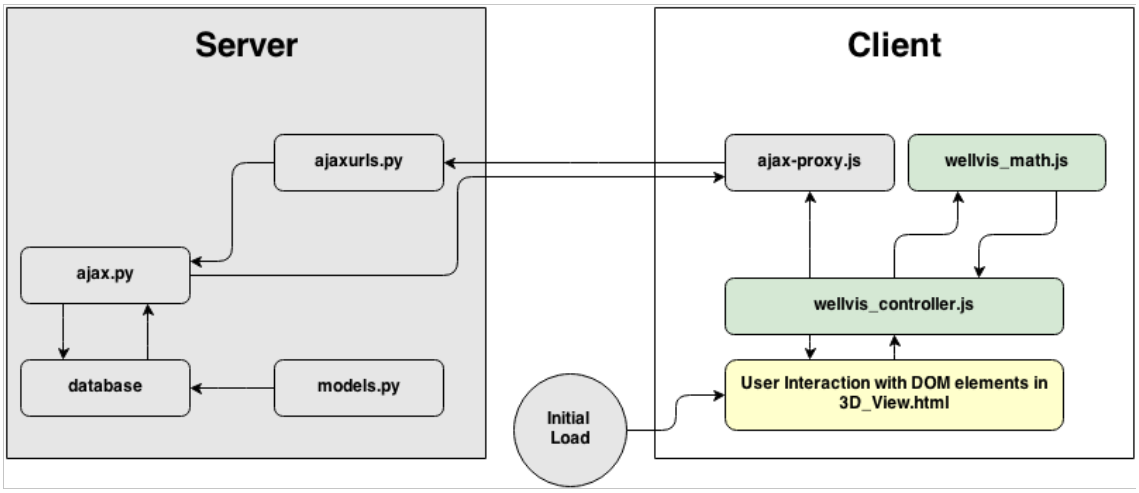


Figure 8.6: The design of interaction flow for F19.

8.4 Sprint Testing

This section is about how the testing for sprint 2 went. In 8.4.1 we show our test results and in 8.4.2 we evaluate how the testing went.

8.4.1 Test Results

During this sprint the team executed a total of six test cases. Here is an example of a test case that has been run in this sprint and the rest of the test cases can be found in Appendix F.

ID	ID07
Description	Choose opacity on grid
Precondition	Must be logged in and have a well path
Feature	Configurable 3D view
Execution	1. Choose configuration 2. Choose and change opacity on grid
Expected Result	The grids opacity should be less visual

Table 8.5: Test case ID07

ID	ID07
Description	Choose opacity on grid
Tester	Tomas
Date	2013-10-11
Result	Success

Table 8.6: Test result for ID07

ID	ID08
Description	Choose color on x-axis
Tester	Tomas
Date	2013-10-11
Result	Success

Table 8.7: Test result for ID08

ID	ID09
Description	Choose size on labels
Tester	Tomas
Date	2013-10-11
Result	Success

Table 8.8: Test result for ID09

ID	ID10
Description	Move rotation point
Tester	Tomas
Date	2013-10-14
Result	Success

Table 8.9: Test result for ID10

ID	ID11
Description	Zoom grid
Tester	Tomas
Date	2013-10-14
Result	Success

Table 8.10: Test result for ID11

ID	ID12
Description	Move grid
Tester	Tomas
Date	2013-10-14
Result	Success

Table 8.11: Test result for ID12

8.4.2 Test evaluation

All tests in this sprint also succeded. We were early finished with what was supposed to be tested in this sprint which made sure we got to do the testing earlier than the last sprint. This was good because if the tests had not succeded we would have had good time to fix it.

8.5 Sprint Evaluation

8.5.1 Customer feedback

The demonstration was presented to the customer representatives where the well-path was built using Tangent method. The other three methods which are used to draw the curve needed a series of calculations; the team was finding it difficult to write algorithms for them. So for these three methods customer representatives helped the team to understand the calculations and the steps required. This cleared the team's confusion and thus could write a proper algorithms. The feedback

on the presentation was positive. They were impressed by the amounts of progress we had made. They suggested some more features for the software but told us that these could be added to further development until we were done with our current load.

The suggestions were:

- Detachable well panel
- Description of estimation method used as a column
- Printout the survey/data

8.5.2 Positive Experiences

- AJAX through jQuery is easy and cool
- Django provides possiblity to store custom fields in DB
- Better task specification
- Better connection between commits and tasks

8.5.3 Negative Experiences

Half of the group was accosiated with UKA, and could not prioritize this project during sprint 2. In addition, one person was much sick during this period. Developers also struggled hard with high priority tasks that proved much more difficult than first anticipated. Two members also had some disagreements.

- Overestimation
- Illness
- Low participation in meetings
- Bad group dynamics

8.5.4 Planned Actions

For illness and low participation, we think this will solve itself as the reasons for this is over.

Task delegation

In order to not let disagreements and differences in personalities interfere with the project, we think clearer boundaries on tasks and delegating them appropriatly would decrease friction between group members.

Lowering implementation ambitions

We had great ambitions of what to implement during sprint 2, and had instead very slow progress. Instead of trying to do as much as possible, we should focus on doing a few things properly.

8.6 Summary

- Added 58 208 lines of Open Source code.
- Added 5 178 own lines of code.
- An underestimation of 128 hours was made, heavily underestimating the most important tasks.
- 4 out of 7 requirements was cancelled due to time restrictions.
- 2 out of 7 requirements was implemented
- An estimated 42 hours remained of one of the requirements
- Friction and slow progres caused low motivation and participation

Chapter 9

Sprint 3

In this chapter we discuss sprint 3 and how it went. We planned the sprint together with our customer after sprint 1 and 2, and since this was the last sprint, we felt like we knew what worked and what did not work. We start this chapter with how the sprint planning was in 9.1 with our goals and the duration of the sprint. Later in the chapter we have our work breakdown in section 9.2 before we continue with our system design for our requirements in 9.3. We also discuss what tests we have done and how they went in 9.4. In the end of this chapter we evaluate sprint 1, 9.5, and see what we can learn till next sprint. We also have a very short summary in the end, section 9.6, where we mention how many code lines we have added and what was good and bad in this sprint.

9.1 Sprint Planning

9.1.1 Duration

Our last sprint has a duration from October 24th until November 8th. This gives us time to fix if anything is missing or not done untill delivery day. What we will do is decided during our presentation for the customer after sprint 1 and 2 to see if some requirements might have changed. Our goal is to finish a week before the deadline so that we can use the last week to fix bugs and test the system. We have weekly meetings with our advisor, but we will not have weekly meetings with our customers since our main contact is a lot away on job, and since we feel like we have the information we need, and know that our customers trust us.

9.1.2 Sprint Goal

The main goal of this sprint is to finish previously started functionality and improve upon the ones we have. Instead of focusing on adding new features (which we did too much of in sprint 2), we want this to look sharp when it is handed over, and it being easy to continue developing. Documentation of the solution will therefore also be prioritized.

9.2 Work breakdown

In this section we will discuss the work breakdown and show some important figures for knowing how the work has been done in this sprint. We start with the backlog in 9.2.1 and we continue with the work breakdown structure in section 9.2.2 and in the end we have the burndown, 9.2.3.

9.2.1 Backlog

ID	Description	Estimated	Actual
F19	Create curvature for path (continues)	21	46
F20	Version Control of input	13	10
F22	Fullscreen mode	8	9
Sum		42	65

Table 9.1: Backlog for sprint 3

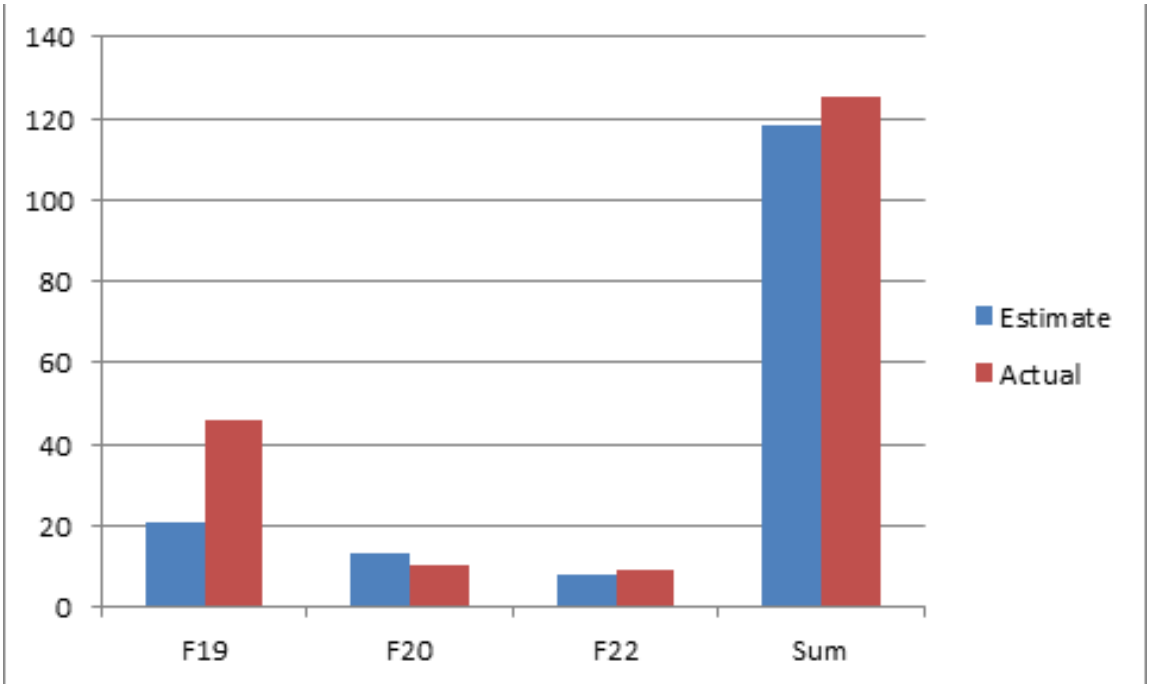


Figure 9.1: Sprint 3 estimations compared to actual time spent.

9.2.2 Work breakdown structure

Work packages are illustrated in figure 9.2.

Implementation of curvature

The remaining parts of the functional requirement F19 needs to be completed. This is different methods for caluculating curvature.

1. M1-inc
2. M1-azi
3. M1-TVD

Wellpath version control

A well path should be automatically version controlled (VC), so that a user can roll back to a previous version of the path.

1. VC webpage
2. VC populate from DB
3. VC restore a previous version to current

Improve Growl

Growl needs better styling, in order to make the user feel comfortable with the notifications.

1. Padding
2. Resizing images to a normal size
3. Modifiable timeout time

Fullscreen mode

User should have fullscreen mode, in order to get a cleaner working surface and better overview.

1. Create button.
2. Remove "frame" functionality
3. Call recalculation of window after removing frame.

Proper login page (low priority)

The login page should redirect user to a relevant user page instead of the admin panel, for better user friendliness.

1. Login page design
2. Backend redirection functionality

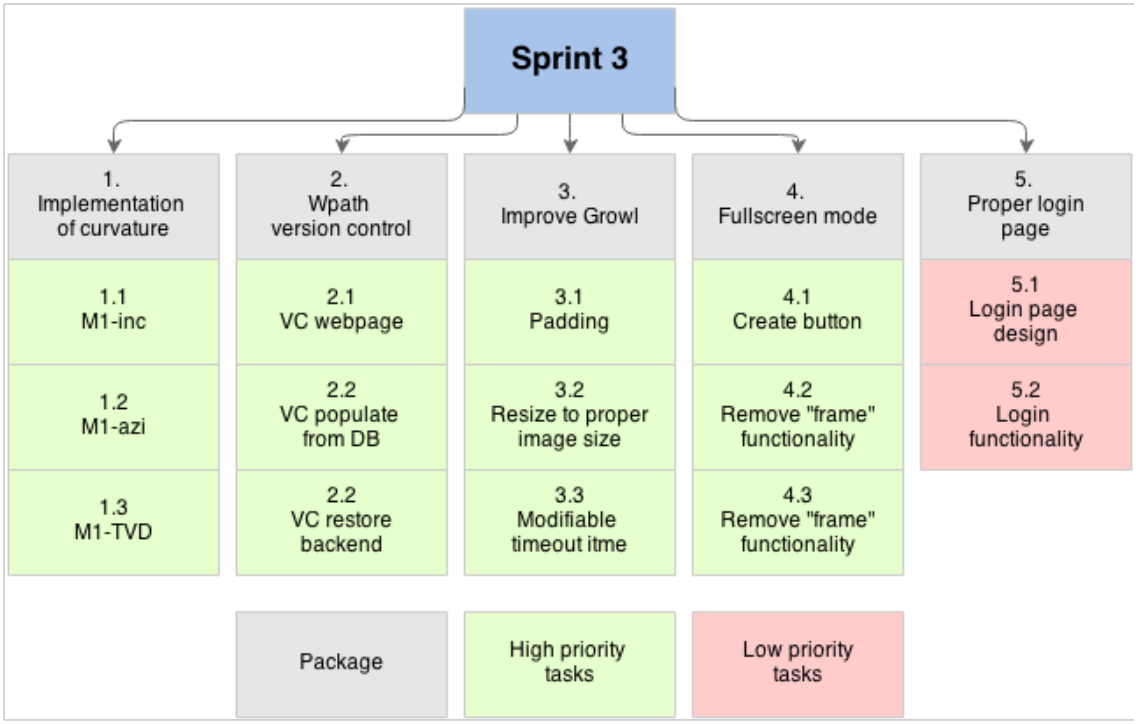


Figure 9.2: Sprint 3 Work Breakdown Structure.

9.2.3 Burndown

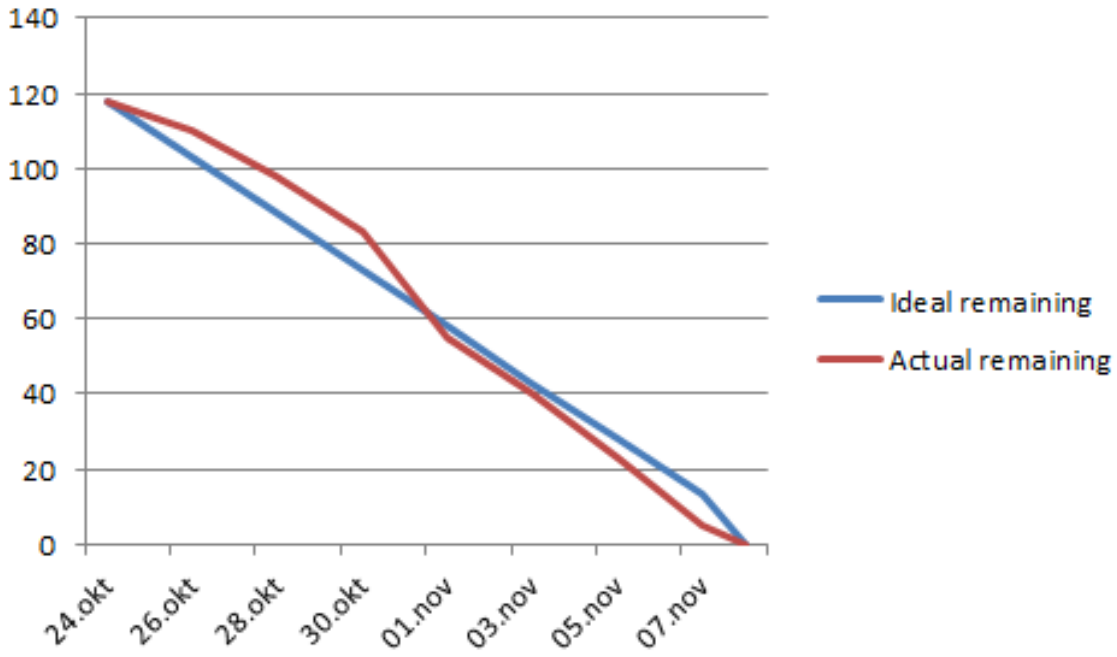


Figure 9.3: Sprint 3 burndown chart.

9.3 System Design

We will in this section describe design for each funcional requirement by specifying the files and their role in fulfilling that requirement. In addition, a figure will show the flow of interactions for that case.

9.3.1 F19 - Create curvature for well path

Files and location

The design for this task was illustrated in sprint 2. See section 8.3.3

9.3.2 F20 - Version control of input

Files and location

ID	Location	Purpose
F20-1	`/wellvis/templates/wellvis/well.html`	Show client gfx elements
F20-2	`/static/js/ajax-proxy.js`	Handle client-server comm. client-side
F20-3	`/static/js/well.js`	Handle client interaction
F20-4	`/wellvis/ajax.py`	Logic behind restoring previous path
F20-5	`/wellvis/ajaxurls.py`	Define urls for client-server comm.

Table 9.2: Files and location for F20

Interaction

User interaction with DOM in F20-1 triggers F20-3 which will update gfx elements and request ajax calls to server through F20-2. This will subsequentially trigger F20-5, F20-4 to ensure updates serverside.

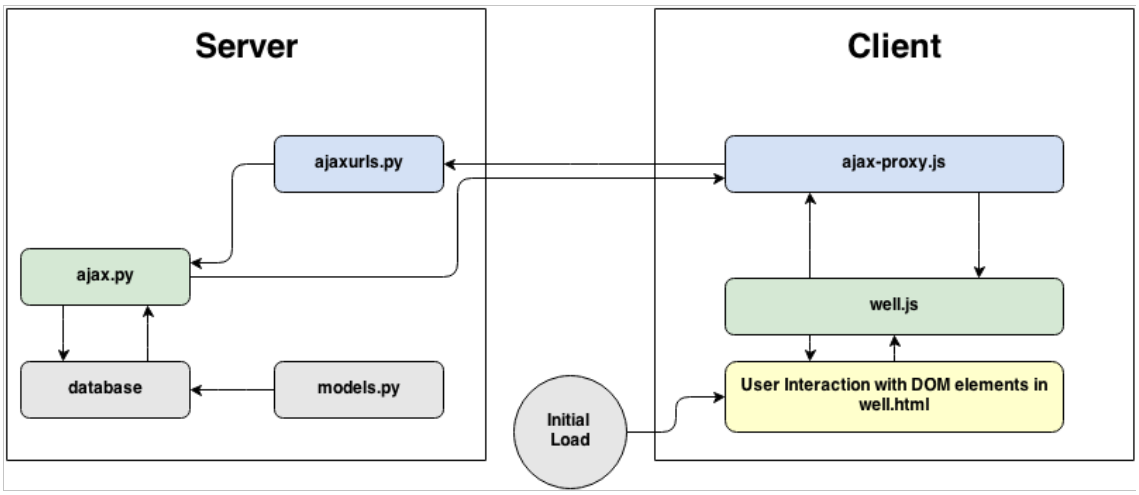


Figure 9.4: The design of interaction flow for F20.

9.3.3 F22 - Fullscreen mode

Files and location

ID	Location	Purpose
F22-1	`/static/js/3D/wellvis_controller.js`	Handle client interaction
F22-2	`/wpath/templates/wpath/3D_View.html`	Show client gfx elements

Table 9.3: Files and location for F22

Interaction

Fullscreen mode is essentially just a client-side grafics modification, which is handled through javascript in F22-2 after being triggered in F22-1.

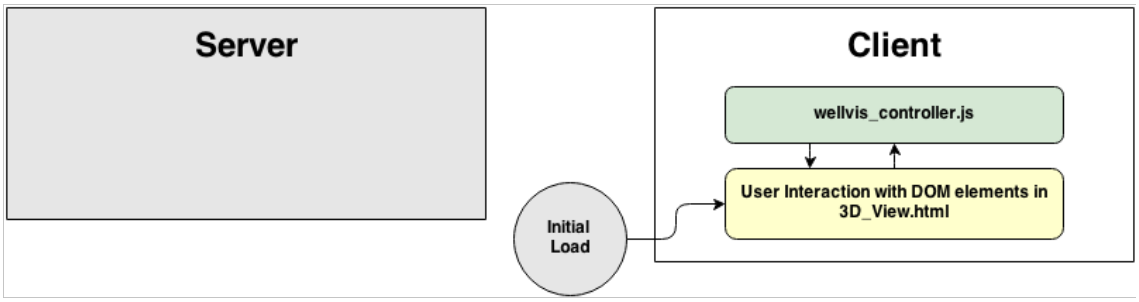


Figure 9.5: The design of interaction flow for F22.

9.4 Sprint Testing

This section is about how the testing for sprint 3 went. In 9.4.1 we show our test results and in 9.4.2 we evaluate how the testing went.

9.4.1 Test Results

During this sprint the team executed a total of two test cases. Here is an example of a test case that has been run in this sprint and the rest of the test cases can be found in Appendix F.

ID	ID12
Description	Choose fullscreen
Precondition	Must be logged in and have a well path
Feature	Test that fullscreen is available
Execution	1. Visualize well path 2. Choose Fullscreen
Expected Result	The well path should be in fullscreen mode

Table 9.4: Test case ID12

ID	ID12
Description	Choose fullscreen
Tester	Tomas
Date	2013-11-06
Result	Success

Table 9.5: Test result for ID12

ID	ID13
Description	Choose previous well path
Tester	Tomas
Date	2013-11-07
Result	Success

Table 9.6: Test result for ID13

9.4.2 Test evaluation

All tests in this sprint also succeded. In this sprint we chose not to have too many requirements to make sure we had more time for documentation. This is why we had so few tests in this sprint. We were using a lot of time on the curvature so the smaller things that we were too test was not prioritized and that is why it was tested in the end of the sprint.

9.5 Sprint Evaluation

9.5.1 Customer feedback

Customer, represented by Bjørn Brechan, visited us on campus 18th of November and came with lots of constructive feedback. The following points are not direct quotes, but summary points of the feedback.

- This is basically a finished part of the end product
- Focus on reflection and the report
- Further development thoughts are important

9.5.2 Positive Experiences

Planned implementation was successfully done in this sprint, which rised the participation and improoved the atmosphere. One developer was moved to documentation which improved communication and dynamics in the group.

- Planned implementations went successfully again
- Group dynamics improved
- More collaboration
- Better communication

9.5.3 Negative Experiences

Stress level increasing lead to not doing routine tasks, such as logging and usage of project task tool. There was still some of the negative experiences left from the previous sprint, though they did improve.

- Stress
- Skipping seemingly trivial routines
- Still suboptimal group dynamics

9.6 Summary

- Added 0 lines of Open Source code.
- Added 5 390 own lines of code.
- All planned implementation successfully done
- Stress caused missing routine task
- Group dynamics improving, but still suboptimal

Part III

Conclusion and Evalution

Chapter 10

Further Development

In this section, we will talk about our thoughts for further development of the Wellvis project.

This includes:

- Choice of technologies
- Choice of tools
- Choice of external services
- Improvements on our implementation
- Risk factors and difficulties
- Feature ideas
- User friendliness ideas

10.1 Supporting different platforms

Customer had a wish of being independant of user platforms. We think this is a good idea, as it is percieved that people will continue to use more and more different platforms in work situations [131]. How we initially thought to solve this was serving the application through a browser window. We still think a good idea. Using representation frameworks like Twitter Bootstrap, Foundation or other have the capabilities to provide a nice interface for different type of screens with minimal amount of effort. If there at one point are resources to create native applications, that should be

considered since they generally run smoother than web pages. We do not recommend you starting this before first version of the software is released, and customers have started using the application.

10.2 Big data and retrieving pattern information

Big data is about using large quantities of data to say something about patterns. The first pattern that we see as potentially interesting is user patterns (what platform do they use, *long* vs *short* clicks [63] etc). This can be registered using Google Analytics [56], a free and powerful tool for you to say something about your users. The second pattern comes into play if you register such events as faults and errors. You could then data mine [44] the patterns between well paths and errors to indicate when a plan is in a state that has a tendency to produce errors. The large amount of data from drilling operations could serve as a potential gold mine for this task.

10.3 Saving large quantities of data from operation

While we have not discussed this much, customer has mentioned that several hundred GB of data will be stored from instruments during every single drilling operation, and that it needs to be accessible to outside companies. Serving this data publically has challenges related to cost. The amount of traffic and data usage, makes this especially expensive to serve from pay-to-use database services such as Amazon RDS [94]. Serving it without pay-to-use databases has operational challenges, and a necessity for great backbone structure. We encourage the customer to reconsider this service, or implement it after Wellvis has achieved decent revenue. Due to the different nature of this service and the challenges it impose on the server, it should anyway be a separated part from the rest of the Wellvis Software.

10.4 Integration between different parts of the software

The solution in use today is very separated, and requires the user to spend much time manually entering data. This is due to the parts originally being separate applications, and will not be any issue if Wellvis is developed as a single solution. The Django framework allows easy communication between subapplications. If a software-part emerges that should have been a part of the Django part, multiple databases can be accessed [34]. We strongly agree with the customer on ensuring this requirement. Neither do we think it is going to be of any issue in the development of the Wellvis software.

10.5 New features

In this section, we will briefly discuss features and ideas for the software. One this are detachable well panel that you can read about in 10.5.1, and another is warnings and notifications in 10.5.2

10.5.1 Detachable Well panel

A wish that emerged from the customer after sprint 2 was multiple windows, specifically a detachable well panel so data points could be entered in one window, and the graphics being represented in another. We did decide not to implement this, due to the time this would take. The basic functionality is one browser window sending an update message to another. Some base restrictions makes this a large task:

- Different browser windows can not interact directly without a native application installed
- Server can not send messages to client browser without a persistent HTTP connection [133]
- The viewing window could poll server every x second.

While persistent HTTP connections might be a good idea when creating Well paths, it also complicates the server side of things, as it cannot be stateless [135]. That being said, we see the functionality as a great idea, and it would simplify design as the viewing and creation of well could be separated in two different views. The simple solution to this is having an Update-button in the viewing window, which requires the user to manually click it so see his updates. This would not require persistent connections, unnecessary polling or native applications at the cost of user not instantly seeing his insertions.

10.5.2 Warnings and notifications

Sending the user subtile notifications and feedback that does not impose restrictions on what they're doing at the moment is why we implemented the Growl module. While this does not provide much useful information at the time being, we suggest using and continue to develop it for future functionality. At the top of our head, we see the following events as notify-worthy:

- Loss or restore of server communication
- Warnings of close proximity to other path
- Above guidance tresholds in either input

10.6 Language and developing

In this section we talk about the languages that can be used, 10.6.1. We also discuss what developing methods that can be discussed in 10.6.2 for further development.

10.6.1 Programming language

As discussed in section 4.6, we ended up choosing Python for developing the main parts of this project. Python and Ruby both have strong web frameworks that are ideal for rapid web development online, and are quicker to develop in (see figure 4.8). However, CPU time is cheaper and easier to buy than developer time. The languages is also considered weaker in really large projects due to the smaller possiblity to check and test with established tools, but Django's framework for dividing the application into subapplication and testing makes up for this a long way. For datamining and handling the large amount of operational data, other languages has more established tools and are required for speed. We suggest Java and C languages for further considered here.

10.6.2 Developing methods

While we have not used it ourselves due to the lack of experience and time, Continuous Integration [43] is a agile and great and clean way to develop in teams. It also avoids potentially messy merging of versions, by small integration cycles. Even with only one developer, it creates a very nice and clear version history. We have fallen in love with git and Github [54], and would recommend it.

10.7 Risks

In this section we will mention some of the risks that can emerge in the further development stage. We mention the possibility of competitors in 10.7.1 and unmanagable code base in 10.7.2

10.7.1 Competitors

The main selling point for the Wellvis software is that all other competing software is outdated and old. This is a dangerous and potentially fragile point. The customer should therefore make an effort to search out and evaluate potential competing software, both for getting ideas from similar software and ensuring the market does not already have a software that Wellvis is going to make. Especially, we found what seemed to be a well established, very similar software: TechDrill [102].

10.7.2 Unmanagable code base

Starting up with one developer has the advantage that routines and development procedures can be easily set. It is therefore very important, both for easier later hiring and ensuring a good structure on code base that good habits are formed early. For this, we suggest

- Using Django or finding another well structured framework
- Start out with good version control habits
- Documention along the way (ie. docstrings in Python)
- Writing tests, preferably pre-implementation (TDD [125])

10.8 Stability

Making sure the database is online, the internet connection stable, and the serving of webpage works is what we define here as stability. These are cumbersome tasks for a startup company, and have scaling issues. Since these tasks neither does exploit the customers main strength (good domain knowledge in well drilling) it should be considered to be outsourced. Web services such as WebFaction [126] can help with this, and when necessary a migration to dedicated database services such as Amazon Data Center [94].

10.9 Offline/Operation functionality

We talked a lot in the beginning about an offline/operation functionality. This was not something we chose to focus on, but some tips for further development about offline mode is in section 10.9.1 and some on operational mode in 10.9.2

10.9.1 Offline mode

The customer has explained that parts of the software should be accessable offline for certain scenarioes. In a traditional, simple server-client web page, this is not the case. We have thought of two options to make this functionality possible. The first option is to create a native application for the relevant platform that updates from and to the server when network connection is established. It should have a *Work offline* or *Disconnect* functionality to download the relevant information before going offline in a controlled manner. While the customer wants platform independancy, the offline mode is needed only in special scenarios at during the drilling operations. Imposing a restriction on platform for this phase might not be a major drawback. It could also be exploited, so that a

customer needs a visit and unit from the Wellvis company, ensuring more work and income. The second option is a heavier client-side of the application through JavaScript that after disconnecting, sends data through a caching JavaScript library that stores temporarily using HTML5 client-side storage [90] could probably do. Note that using this has a file quota of usually 5 MB, and could be to small. Creating a browser application for e.g. Chrome [3] could expand this, but then again poses some limitations on the user.

10.9.2 Operational mode

During drilling operations, data should be inserted into the application to model where the drilling head is located, offset from plan and corrections to be made. Customer explained that this is a functionality that does not exist in the software used on Norwegian oil rigs today, and that calculations are done manually. We believe that implementing this operational mode would be of great value to the product, and recommend it strongly.

10.10 Further WebGL

WebGL is in rapid development, and many different frameworks are emerging. Investing heavily in developing using frameworks can therefore be risky in that it requires alot of adapting to be compatible with newer versions. We do not consider writing the application using plain HTML5 elements to be a viable option, due to the massive time necessary to both learn and write this. Using a relatively stable version of an existing library is possible, but then you will either be stuck with that version or continuously upgrade it. We have in this project shown that most of the developing of the well module is possible in a couple of months. And that was done by a developer that had never previously used WebGL. We therefore recommend to wait with the graphics until towards the end of the planning and eventual implemention. This is mainly to give the community time to settle and mature to a better and more stable situation.

Chapter 11

Conclusion

In this chapter we will conclude with how we felt things went in this project. We start of this chapter with an overview of our system in 11.1. We continue with talking about our testing in 11.2 where we talk aout our testing methods and conclude on how the testing went. In the end we have a summary, 11.3, where we conlude with what we have done and how we feel it has gone, and what we believe our customer feels about our project.

11.1 System Overview

Figure 11.1 shows the overall structure of the product. The outermost layer is the wellvis applications, which is the part that the user can directly interact with. It specifies valid urls (urls), performs logic (views), and serves the client files to the user (JavaScript and HTML). Along with the client files it would establish a connection to the database through the Django core. The WellPath module is the only part that we developed for this project so there would be multiple similar parts that can be served to the user. User will interact with a copy of the client files, which in turn will repeat an indentical procedure through the system. More details on the application structure can be found in the appropriate section 6.4.

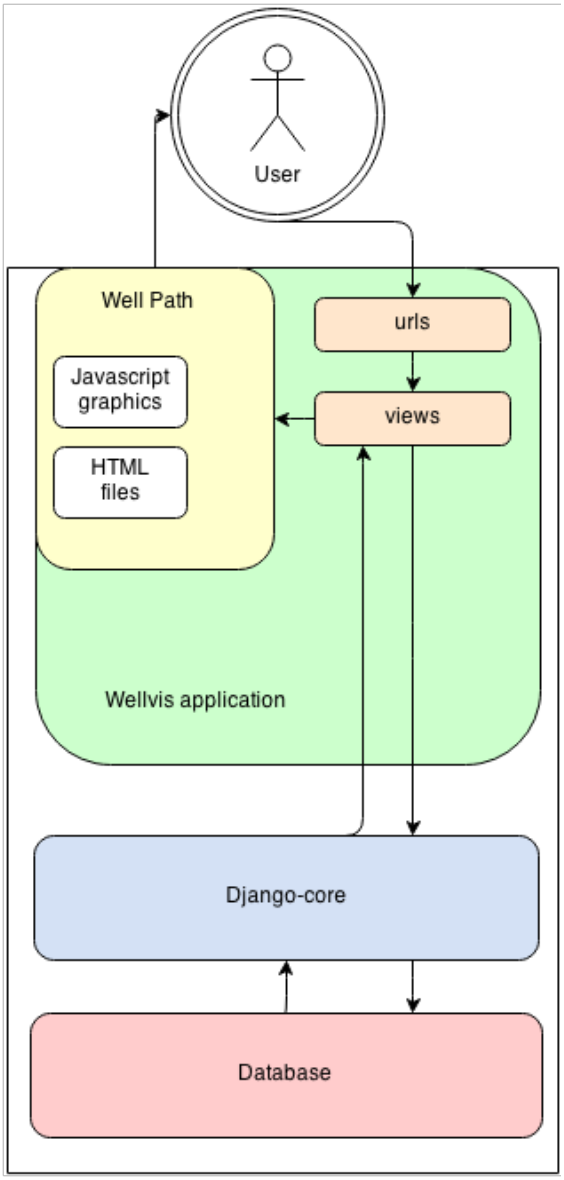


Figure 11.1: The system overview and communication between different parts

11.2 Testing

At the beginning of the project we all agreed that we were going to focus on doing extensive and proper testing of our system. In this section we will discuss how we decided to do this testing, what methods were used can be read in 11.2.1. We also have a conclsion of our tests in 11.2.2 where we discuss if we have reached our goals.

11.2.1 Testing Method

We mostly used black box testing during our testing phase. This did not necessarily show us where we had coded wrong, but it showed us what was wrong in the interface. Since our software is only a part of a bigger, we have to consider that a lot of code, if software is to be used later, will be changed to fit the other modules. So this is why we decided to prioritize black box testing and showing what the system can do. We made test cases for our requirements and used these to figure out what to test and how to do it.

11.2.2 Testing Conclusion

We can conclude with that our focus on testing is not that necessary as it would have been for a finished product, but still we have worked hard on it because it is such an important part of a software project. Our testing methods working well and we feel like we have done sufficient testing for this project.

11.3 Summary

Our team was given a project that had a big scope and that had a lot of oppurtunities. It was a lot to wrap our heads around which made it a bit difficult to get started, but when we had enough information and understanding we decided to go for the module that we felt that was most needed, and that was interesting to work with.

Our job was to create a module for the software that was easier to use but still with the same and better functionalities than the old version. We focused on the graph because we felt that this was something wecould greatly improve and, we also managed to this very well and the customer agreed that it was already better than the original already after the first sprint.

The customer thinks the graph part of the module is done and we believe that the customer will be able to use our product in their continuation with their project. We therefore feel that we have delivered a part of the solution to the big task that the customer presented at the beginning of the project and that we have fulfilled the project in a way that is pleasing to them.

Chapter 12

Evaluation

In this chapter we will evaluate our work during this project. In section 12.1 we will discuss how we have worked as a group, while we in 12.2 will explain how we have handled our risks. The scrum process is new to us, and how we handled that is explained in 12.3 and our time estimation is discussed in section 12.4. Discussion of our quality assurance follows in 12.5 and we explain how our customer relations have worked in section 12.6. We also think it was appropriate to discuss our lessons learned in 12.7 and in the end we have a summary, 12.8.

12.1 Group Dynamics

In this section we talk about our group dynamics through these months. Group dynamics is important for managing to do a project together. We start of with our goals and team building in section 12.1.1 and we also mention the team evolution for these months in 12.1.2.

12.1.1 Goals and Team Building

In this project we were all assigned groups randomly, this was done so that we would work in a realistic work setting, where we usually are unable to choose who we work with. Our group originally consisted of four norwegians, but one did not show up since he was taking another course isntead. We sent a mail to the coordinator and told them that we were only three persons, and later in the day we got assigned a new member, an exchange student from Nepal.

When it comes to relations between team members, none of us had worked together earlier. Some of the norwegian members had been introduced to eachother on a class trip for computer science students.

Because not everyone spoke Norwegian, we had to communicate in English. Our supervisor spoke English as well, and since the report were to be written in English it all worked out well. No one had any any problems with speaking or writing English, but sometimes some discussions were done in Norwegian, but this happened only when everyone at the meeting could understand it.

The course had arranged a team building course, during which we decided on some shared goals for this project.

- *Grade:* Some of the members wanted an A in the course, so everyone decided that they would to their best to contribute to this goal. Although the goal is big, it is important to have something to reach for.

- *Decisions:* Always consider options and strengths and weaknesses, then vote. This should be done quick.

- *Conflicts:* Must take the time to discuss this. Use on of the meetings in the week to discuss what we like and dislike.

- *"Freeriders":* Must confront them and make sure to give them something to do to make up lost time.

- *Help:* Ask for help when you are stuck.

- *Cultural differences:* Should speak English so that no one feels left out, and if something is more easily discussed between some members in Norwegian, we should translate later so everyone is up to date.

- *Knowledge:* Utilize everyones knowledge so no one feels left out and to work the best way possible

- *Roles:* Need to establish roles so everyone has one main responsibility.

These goals has helped us a lot during the project and it has contributed to that making sure everyone is working at maximum capacity.

12.1.2 Team Evolution

During the first weeks we were gathering information about our customers and each other and we discussed how we were going to procede with this project. We did not have any real leader type, which made things a bit hard because when people were unsure of what to do, no one took command. We also used long times on our meetings.

Luckily we figured this out early in the project and we decided in the start that two people worked on the documentation, while the other two worked on the implementation for the first sprint. We

also started having scrum meetings where we quickly found out what people had done since last time, problems and questions, and what they were to do till next time. This worked a lot better because then the people who were not sure what to do got their own tasks. The taskmanaging with Pivotaltracker was not used so much during sprint 1, so when people forgot the scrum meetings, and everyone did not attend the meetings, someone did not have anything to do, even though they asked for something to do. There was also a problem that when the person who wrote the summaries for the meetings was missing, no one else wrote the summaries.

In sprint 2 we managed the tasks in Pivotaltracker and this made it a lot easier to find something to work with when you had not been able to attend the meetings. To find time for the meetings was not so easy this month since half the group was involved with UKA, and one of these had to work with UKA approximately every second day. Luckily someone else tried to make sure the scrum meetings were filled out, and the day before the status report were to be sent in we also added what had happened that week so it would be easier to write the status report even if not all scrums were filled out. We also found out that we had to many requirements and did not have the time for all of them in this sprint.

In our last sprint we decided to use more time on documenting and less time on implementing. This was to make sure that we were not to have to much to do with the report when the last sprint were to be done. We also did this because we saw that we had to much in sprint 2 and it was better to make sure we had a working product in the end. We also had the problem with people being involved with UKA, and sickness, so it was important to make sure that we reached our goals in the last sprint.

Some of the team members had a big disagreement about how to work and help each other. One person felt that when another person rewrote everything they had done, that what they had done was useless and did not feel like they had produced anything, while the other one was used to doing so because he was used to collaborating on wikipedia-articles and such. It was decided that if you disagreed you were to comment so that the person could fix it themselves if they agreed, unles there were some grammar errors and such. This was to make sure that no one felt that their work meant less and because there could be several possibilities that are right, it is not necessarily just one person that can write that chapter or section the right way.

We had a few difficulties here and there, but it did not necessarily negatively influence the final product or report. We have learned a lot and when we have seen that something have been missing, or not done well enough, we have tried to improve this for the next sprint and even the next week.

12.2 Risk Handling

Some of the risks dicussed have occured during this project to a certain degree. This section will discuss how these risks were handled.

R1 Sickness/No-show

Over these months we have had some sickness. We have also had the problem that members have several projects and have had to prioritize these sometimes. The biggest problem might have been that one of the members have had a lot to do with UKA. We have tried fixing this by working from home, but sometimes there have been som communication issues that has contributed to that the persons at home do not understand what to do, or have not understood what they are supposed to do. This has mostly caused a delay in the documentation, but we have tried really hard to learn from mistakes and be better at communicating and letting the group know when there are some issues with the tasks.

R3 Arguments

This has not been a big issue, but because people work in different ways, it has come up. Some of the team-members did not agree on how to help each other when documenting. This is because they are used to working in different ways. We have talked about this so that everyone have agreed on what is okay to edit, and what should be commented so that people can fix it if they think it is necessary as well. Luckily this happend early in the project so there was a lot of time to fix it.

R4 Low Motivation

Working in a team where not everyone is able to work together all the time because of other obligations and projects makes it hard to motivate to work when others arenot working. Also not being able to fully understand what to do has lead to low motivation for some. Because we really want to please our customer and do a good job on this project this has not been a big issue and everyone has tried to do their best in prioritizing.

R5 Underestimating Size

Underestimating the project had a high likelihood for happening and also happened in our case. When planning it is not always easy knowing how long it takes to implement all requirements. We knew that we might not be able to implement all requirements, and that our requirements would change during the whole process. We decided on what we should implement, and what we could implement if we had the time. We should also have divided the tasks into smaller parts. Even though we had some problems we still learned a lot in planning a project and we still appreciated what we managed to complete.

12.3 Scrum Process

It was decided that we were to use scrum in the preplanning phase. This was recommended by the course, and it is also recommended by several companys, and is used a lot in the real world so it was a clear choice for us. We have not had any courses or such so no one knew a lot about how to do scrum, we only knew some parts and that it is a good technique.

Because our project was so big we needed a lot of information from our customer before starting to plan the sprints. When we finally felt like we had the requirements we needed, we planned the sprint together with the customer. We decided to match the sprint with some of the deadlines given to us by our supervisor.

After the first sprint we had a customer meeting. In this meeting we found out that some of the earlier requirements was no longer needed, and we also got some new requirements. This made us change the sprint plan, but it helped us a lot with following what the customer wanted. This was also done after sprint two. Since we were not going to do any more implementation after sprint three, we decided not to have a customer meeting, since they will see everything we have done in the last presentation.

Our sprints all lasted for three weeks. This might have been a bit too long we see know, and we have also heard from a lot of companies that a sprint should be short and concise, approximately a week. Since we did not have any experience we did not think this through. Luckily we did not use much time on requirements that appeared to be unnecessary during the customer meetings so it did not affect the final product. We had a lot of contact with the customer during the whole project via mail.

We think that scrum worked very good for this project, and we will probably use it more. It made it a lot easier getting input from the customer during the project, and not only in the end, because that made sure that we followed the customer's wishes, and when they changed their requirements, we were able to do it as well. This made the project more suited for the customer.

12.4 Time Estimation

For this project we estimated the total effort to be 1400 hours. We ended up using 1337 hours. Most of these were from the end of the project and in the end of each sprint because we needed to reach our goals and deadlines.

We used less hours than we were supposed to, but we still managed to do everything we were to do. Some of the reasons for this was because we in the start did not have enough information to go on with the project, and some others was that there was a lot of other projects and such in periods. This means that there were some weeks we worked a lot more than 25 hours, and some we worked a bit less.

We estimated approximately time, but we did not hold it. The implementation took a lot more time than we had estimated. Some of the reasons for this was because there was really heavy math equations behind the curvature and it did take up a lot of time during the last two sprints. We also got a bit cocky after sprint 1 and did not think that the requirements from sprint one was a lot less work than the ones in sprint 2.

We started of really good with documenting and if we had kept that pace the documentation would have been no problem. Sadly we kind of stopped a bit on the documentation during the sprints, and also some of the documentation needed to be done by someone who was programming, and because of this we ended up with using a lot more time on documenting in the end than we were supposed to.

We tried to keep the activity up to 25 hours a week from week 2-3 but this was not easy when we did not always have enough to work with. When we starter the sprints there was no problem in getting enough hours, but we had the problem that people had other obligations and group work, and that made it a bit hard to be able to work the hours we promised to put in. This however changed in the end of each sprint and in the end of the project when everyone worked to finish the report and prototype.

12.5 Quality Assurance

We made several plans for ensuring quality when starting this project.

We had planned to always have another team member to double check when finishing a task to ensure the best quality. We were unable to follow up on this because of poor planning. Everyone also had more than enough to do on their own, and we did not have the time to go through what the others had done together ith them. Since we all had other projects and obligations we decided that as long as it worked, it was okay. When it comes to the documenting this was something we tried to go through in the end, but mostly to check for spelling mistakes and errors.

Other plans involved having a scrum meeting so we always were to know what everyone was doing. This mostly worked as long as people attended the meetings. If they did not attend they did not fill in the scrum. We also had the problem that when the meeting leader, or two of the members did not attend, the scrum was not filled out at all. Another thing was to follow up on the responsibility given to each team member. Sometimes peolpe got a bit carried away with other things in the project that made them forget stuff like make tasks, but when figuring this out everyone managed to do what they were supposed to.

After each sprint we had a meeting with our customer to show our preliminary product and to talk about the next sprint. This worked really good and made sure we were always on the right track and that our customers always had a say in what was needed. This helped us with thorwing out no longer needed requirements, and discovering new ones. This was great for our project.

Our plan was to fill in a form after every work weher we filled in our hours. This worked great

sometimes, but not always. People forgot to fill in their hours, and we did not set aside time during our meetings because we often went on to work other places. This has been a problem when presenting our hours to our advisor. We should have learned to always end or start our meetings with filling out the hours.

12.6 Customer Relations

We had a really good relationship with our customer during the entire project. Since they are only three people in the company we had contact with all of them, but since one of them were writing their doctorate on this project, we mostly had contact with him. Everyone was invited to all customer meetings and got all status reports so that everyone knew how everything was going the whole time.

There were only one of the persons that had a technical background, the other two had background from the oil industry, but this did not slow or project in any way. They really wanted us to do what we wanted and that we were to use tools and such that were chosen by us. They were really open to all our ideas and they were really good at explaining everything we wondered about. Our main customer contact was in and out for four weeks at a time, but this did not stop him from keeping contact and answering our questions when needed.

Already in our first meeting they had a lot of paperwork we could read about what they were thinking and planning and they shortly after shared a dropbox file with us and it was filled up with new information often, both information that we requested, and shown too us in our meetings.

We were invited to come and see the existing sofware, at Statoil in Trondheim, so that we could make our minds to what we would like o do with it and what we thought we might be able to do. This helped us a lot in deciding which parts we wanted to do. Here they also introduced a fifth grader from petroleum who we also could ask if there was anything we were wondering about. They also made sure that we got permission to visit and test the software at petroleum section in NTNU.

It was not possible for us to get access to their database, but since we only are making a little part of the software, we did not really see this as a issue. Everything else we needed we got and they were really great.

We were really happy with our customer and we really felt we got a good connection and that we got everything we needed. We are really happy that they let us choose so much for our selves, this made this project so much easier and fun to work with. It was really interesting being involved in making the requirements and we are really happy with the fact that they appreciated all of the suggestions we had. They were always quick to answer when we sent mail with questions, and they were always ready to help when we needed them to. They really helped us understand the software and what they needed in a new one. All in all, we could not have had a better customer.

12.7 Lessons Learned

Through this experience we have learned a lot when it comes to working as a group, and at the same time learning new techniques.

Working in a group that is randomly assigned to you is a lot like working in real life. We have learned that it is important to understand how the other people work, and what they like and dislike so that no one feels left out. As a team we have agreed on what our goals are and learned how to work together as a team to reach these goals. There has been discussions and arguments that we have learned how to handle together, and we have helped each other and learned from each other.

Time has been a big issue during this process. Since we are students and this is not our only course, it has been hard finding a time for the meeting since we do not take the same courses. Also other commitments, like work and voluntarily work, have come in the way. Still we have learned that if we are good at giving assignments and still filling out the scrum, even though we do not have the time to meet, everyone still works, if it is from home or at school.

The scrum process was new for everyone, although we all had heard about it. Working with this technique was new and it took some getting used to. We learned that we should have started with the scrum meetings from the beginning because it really made sure that everyone had something to do. We also larned that a sprint that goes over three weeks are a bit long, this is because requirements change and when the customer sees what has been done, their requirements change.

When it comes to time estimation this has been really complicated. We have tried to fill in how much we have done everyday, but this has been forgotten many times, which have made things a bit more difficult. Some people have also worked more in some periods and less in other periods. All in all we have learned that planning is the most mportant thing when having a time estimation.

Our customer relations have learned us a lot. Having a real customer and how to communicate with them has been a completely new experience for us. We have learned how to behave and communicate with them. We have not had any problems with them, so we have not learned how to handle complications, with them, but we know how to handle them and it has been an amazing experience and we have been really lucky. We know that we might not be so lucky later, but you still learn a lot from a good experience.

12.8 Summary

We are very pleased with what we have managed to achieve in this project. We were four different people with different personalities and skills that had to worked together as a group. Our scope might have been one of the most complicated, and it had to do with an industry we did not know anything about. Luckily we got a lot of help from our customer that helped us understand and let us do what we wanted in our own way. We have worked very hard and learned a lot with help from

our customer and advisor and we are really proud of what we have managed. It has not been a bumpy ride with some ups and downs both when it comes to scrum and meeting techniques. Over the weeks we learned to communicate and collaborate better which increased our work effort. We have learned a lot through this challenging process and have gotten invaluable work experience.

Bibliography

[1] Apache. Subversion. http://subversion.apache.org/, November 2013.

[2] Apple. Thoughts on flash. http://www.apple.com/hotnews/thoughts-on-flash/, April 2010.

[3] Chrome apps. Create your first app. http://developer.chrome.com/apps/first_app.html, November 2013.

[4] Atlassian. Git vs mercurial: Why git? http://blogs.atlassian.com/2012/03/git-vs-mercurial-why-git/, February 2012.

[5] Atlassian. Getting started with distributed version control system. https://www.atlassian.com/dvcs/overview/dvcs-options-git-or-mercurial, November 2013.

[6] Babylon. Webgl framework. http://babylonjs.com, November 2013.

[7] Balsamiq. drawing software. http://balsamiq.com/, November 2013.

[8] Twitter Bootstrap. Mit license. https://github.com/twbs/bootstrap/blob/master/LICENSE-MIT, November 2013.

[9] Bjørn Brechan. *System Development Report.* PhD thesis, Department of Petroleum Engineering and Applied Geophysics, NTNU, 2013.

[10] Encyclopedia Britannica. C++. http://global.britannica.com/EBchecked/topic/688345/C, November 2013.

[11] BuiltWith. Foundation usage. http://trends.builtwith.com/docinfo/Foundation, November 2013.

[12] BuiltWith. jquery usage statistics. http://trends.builtwith.com/javascript/jQuery, November 2013.

[13] BuiltWith. Twitter bootstrap trends. http://trends.builtwith.com/docinfo/Twitter-

Bootstrap, November 2013.

[14] Mobile Development Center. Will html 5 replace java. http://mobiledevelopmentcenter.info/2013/01/will-html-5-replace-java.html, Januar 2013.

[15] Debian. The computer language benchmark game. http://shootout.alioth.debian.org/, May 2009.

[16] Debian. build-essential package. http://packages.debian.org/sid/build-essential, November 2013.

[17] denyhosts. Gpl license. http://en.opensuse.org/DenyHosts, November 2013.

[18] Denyhosts. Security software to thwart ssh attacks. http://denyhosts.sourceforge.net/, November 2013.

[19] Django. Bsd licence. https://github.com/django/django/blob/master/LICENSE, November 2013.

[20] Django. The django admin documentation generator. https://docs.djangoproject.com/en/dev/ref/contrib/admin/admindocs/, November 2013.

[21] Django. The django admin site. https://docs.djangoproject.com/en/dev/ref/contrib/admin/, November 2013.

[22] Django. django-admin.py and manage.py. https://docs.djangoproject.com/en/1.5/ref/django-admin/, November 2013.

[23] Django. Django and doctests. https://docs.djangoproject.com/en/1.5/topics/testing/doctests/, November 2013.

[24] Django. Django settings. https://docs.djangoproject.com/en/1.5/topics/settings/, November 2013.

[25] Django. The django template language. https://docs.djangoproject.com/en/1.5/topics/templates/, November 2013.

[26] Django. How to install django. https://docs.djangoproject.com/en/dev/topics/install/, November 2013.

[27] Django. Models. https://docs.djangoproject.com/en/1.5/topics/db/models/, September 2013.

[28] Django. Queryset api reference. https://docs.djangoproject.com/en/1.5/ref/models/expertsets/, November 2013.

[29] Django. Request and response objects. https://docs.djangoproject.com/en/1.5/ref/request-response/, October 2013.

[30] Django. Testing in django. https://docs.djangoproject.com/en/1.5/topics/testing/, September 2013.

[31] Django. Url dispatcher - http reveesing url namespaces. https://docs.djangoproject.com/en/dev/topics/http/urls/#topics-http-reversing-url-namespaces, November 2013.

[32] Django. The web framework for perfectionists with deadlines. https://www.djangoproject.com/, November 2013.

[33] Djangobook. Chapter 6: The django admin site. http://www.djangobook.com/en/2.0/chapter06.html, October 2013.

[34] Django documentation. Multiple databases. https://docs.djangoproject.com/en/dev/topics/db/multi-db/, November 2013.

[35] Doxygen. Generate documentation from source code. http://www.stack.nl/ dimitri/doxy-gen/, November 2013.

[36] draw.io. Copyright. http://draw.io, November 2013.

[37] draw.io. online drawing tool. http://draw.io, November 2013.

[38] Chris Duckett. Mozilla edges closer to replacing flash with javascript. http://www.zdnet.com/mozilla-edges-closer-to-replacing-flash-with-javascript-7000021499/, October 2013.

[39] easyBacklog. Backlog management tool. http://easybacklog.com, November 2013.

[40] Eclipse. Homepage. http://www.eclipse.org/, November 2013.

[41] Flask. A python microframework. http://flask.pocoo.org, November 2013.

[42] Foundation. Frontend css framework. http://foundation.zurb.com/docs/, November 2013.

[43] Martin Fowler. Continous integration. http://www.martinfowler.com/articles/continuousIntegration.html, May 2006.

[44] Jason Frand. Data mining: What is data mining? http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm, November 2013.

[45] FreeFormatter. Mime types list. http://www.freeformatter.com/mime-types-list.html, Octo-

ber 2013.

[46] Frothkit. Objective-c web application framework. https://code.google.com/p/frothkit/, November 2013.

[47] Git. About version control. http://git-scm.com/book/en/Getting-Started-About-Version-Control, November 2013.

[48] Git. Gplv2 licence. http://git-scm.com/about/free-and-open-source, November 2013.

[49] Git. Post receive hooks. https://help.github.com/articles/post-receive-hooks, November 2013.

[50] Git. A short history of git. http://git-scm.com/book/en/Getting-Started-A-Short-History-of-Git, November 2013.

[51] Git. A short history of git. http://git-scm.com/book/en/Getting-Started-A-Short-History-of-Git, November 2013.

[52] Git. Version control software. http://git-scm.com/, November 2013.

[53] Github. Set up git. https://help.github.com/articles/set-up-git#platform-linux, November 2013.

[54] Github. Version control software. http://github.com, November 2013.

[55] GNU. Screen. https://www.gnu.org/software/screen/, November 2013.

[56] Google. Google analytics. http://www.google.com/analytics/, November 2013.

[57] JetBrains. Pycharm. http://www.jetbrains.com/pycharm/, November 2013.

[58] jQuery. Javascript framework. http://jquery.com, November 2013.

[59] jQuery. Mit licence. https://jquery.org/license/, November 2013.

[60] jQuery growl. License. http://projects.zoulcreations.com/jexpert/growl/jquery.growl.js, November 2013.

[61] Canvas 3D JS. Webgl framework. http://www.c3dl.org/, November 2013.

[62] HTML Kickstart. Frontend css framework. http://www.99lime.com/, November 2013.

[63] Aj Kohn. Short clicks vs long clicks. http://www.blindfiveyearold.com/short-clicks-versus-long-clicks, September 2009.

[64] Philippe Kruchten. Architectural blueprints—the “4+1” view model of software architecture.

http://www.cs.ubc.ca/ gregor/teaching/papers/4+1view-architecture.pdf, November 1995.

[65] LangPop.Com. Programming language popularity. http://www.langpop.com/, November 2013.

[66] Paul clements Len Bass and Rick Kazman. *Software Architecture in Practice*. Pearson Education, Inc, 2013.

[67] Lucidchart. "flow chart maker & online diagram software". https://www.lucidchart.com/, November 2013.

[68] Guillaume Marceau. The speed, size and dependability of programming languages. http://blog.gmarceau.qc.ca/2009/05/speed-size-and-dependability-of.html, May 2009.

[69] Mozilla. Javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript, November 2013.

[70] Netatalk. Gplv2 license. http://netatalk.sourceforge.net/2.2/ReleaseNotes2.2.1.html, November 2013.

[71] Netatalk. Open source afp file server. http://netatalk.sourceforge.net/, November 2013.

[72] Netscape. Netscape and sun announce javascript, the open, cross-platform object scripting language for enterprise networks and the internet. http://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html, December 1995.

[73] Jolie O'dell. 8 experts break down the pros and cons of coding with php. http://mashable.com/2010/11/19/pros-cons-php/, November 2010.

[74] Norwegian Ministry of Petroleum and Energy. Norway's oil history in 5 minutes. http://www.regjeringen.no/en/dep/oed/Subject/oil-and-gas/norways-oil-history-in-5-minutes.html?id=440538, October 2013.

[75] Ruby on Rails. Web framework. http://rubyonrails.org/, November 2013.

[76] oomph. Will html 5 replace flash in the next 5 years? http://www.oomphinc.com/thinking/2010-02/html-5-flash-future/, February 2010.

[77] OpenSSH. http://www.openssh.com/, November 2013.

[78] openssh server. Bsd license. http://www.openssh.com/, November 2013.

[79] Python Pip. Mit license. https://pypi.python.org/pypi/pip, November 2013.

[80] Pylons. Pyramid web framework. http://www.pylonsproject.org/, November 2013.

[81] pytest. "helps you write better programs". http://pytest.org/latest/, November 2013.

[82] Python. Data structures - dictionary. http://docs.python.org/2/tutorial/datastructures.html#dictionaries, November 2013.

[83] Python. doctest. http://docs.python.org/2/library/doctest.html, November 2013.

[84] Python. Pip. https://pypi.python.org/pypi/pip, November 2013.

[85] Python. Psf licence. http://docs.python.org/2/license.html, November 2013.

[86] Python. pydoc - documentation generator and online help system. http://docs.python.org/2/library/pydoc.html, November 2013.

[87] Python. unittest. http://docs.python.org/2/library/unittest.html, November 2013.

[88] Python. Web framworks. https://wiki.python.org/moin/WebFrameworks, November 2013.

[89] Top Ten Reviews. "microsoft visio review - 9.28/10". http://flowchart-software-review.toptenreviews.com/visio-review.html, November 2013.

[90] HTML5 rocks. Client-side storage. http://www.html5rocks.com/en/tutorials/offline/storage/, October 2010.

[91] GNU screen. Gpl license. http://en.wikipedia.org/wiki/GNU_Screen, November 2013.

[92] Selenic. Mercurial. http://mercurial.selenic.com/, November 2013.

[93] Selenium. Web browser automation. http://docs.seleniumhq.org/, November 2013.

[94] Amazon Web Services. Amazon relational database service (amazon rds). http://aws.amazon.com/rds/, November 2013.

[95] Rally Software. Rallydev community edition.

[96] TIOBE Software. Tiobe programming community index for november 2013. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, November 2013.

[97] Ian Sommerville. *Software Engineering, Ninth Edition.* Pearson Education, Inc, 2011.

[98] South. Apache licence. https://pypi.python.org/pypi/South/, November 2013.

[99] Spectrum. License. https://github.com/bgrins/spectrum/blob/master/LICENSE, November 2013.

[100] Sphinx. Python document generator. http://sphinx-doc.org/, November 2013.

[101] Spiceworks. "microsoft visio reviews - 4.5/5". http://community.spiceworks.com/product/14818-microsoft-visio, November 2013.

[102] Techdrill. "the all-in-one integrated drilling software". http://techdrill.com/, November 2013.

[103] The Top Tens. "best visio alternatives". http://www.thetoptens.com/best-visio-alternatives/, November 2013.

[104] Sublime Text. "the text editor you'll fall in love with". http://www.sublimetext.com/, November 2013.

[105] Patrick Thomson. Git vs mercurial: Please relax.

[106] Three. Mit license. https://github.com/mrdoob/three.js/blob/master/LICENSE, November 2013.

[107] THREE. Webgl framework. http://threejs.org, November 2013.

[108] Charles Max Wood Jamison Dance Tom Beatty, AJ O'Neal. Can html5 and javascript really replace flash. http://javascriptjabber.com/011-jsj-can-html5-and-javascript-really-replace-flash/, April 2012.

[109] tomfa@github. wellvis project repository. https://github.com/tomfa/ww, November 2013.

[110] Pivotal tracker. Project management tool. https://www.pivotaltracker.com/, November 2013.

[111] Trello. Task organiser software. https://trello.com/, November 2013.

[112] Regular Expression Tutorial. Learn how to use regular expressions. http://www.regular-expressions.info/tutorial.html, November 2013.

[113] GNU/Linux tutorials. An extra user account. http://www.debian.org/doc/manuals/debian-reference/ch01.en.html, November 2013.

[114] Typhoon. objc dependancy injection container. http://www.typhoonframework.org/, November 2013.

[115] Udemy. Infograpic: Ruby vs python vs php. https://www.udemy.com/blog/modern-language-wars/, November 2013.

[116] vim. Gpl license. http://www.vim.org/about.php, November 2013.

[117] VIM. "the editor". http://www.vim.org/, November 2013.

[118] virtualenv. Mit license. http://www.virtualenv.org/en/latest/virtualenv.html#status-and-license, November 2013.

[119] Virtualenv. Tool to create isolated python environments. http://www.virtualenv.org/, November 2013.

[120] Microsoft Visio. "flowchart software". http://office.microsoft.com/en-001/visio, November 2013.

[121] Microsoft Visio. License. http://office.microsoft.com/en-us/microsoft-software-license-terms-for-microsoft-office-2010-HA101817777.aspx, November 2013.

[122] vsftpd. Gpl license. https://security.appspot.com/vsftpd.html, November 2013.

[123] vsftpd. Very secure ftp deamon. https://security.appspot.com/vsftpd.html, November 2013.

[124] w3schools. Javascript. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, November 2013.

[125] Scott Wambler. Introduction to test driven development (tdd). http://www.agiledata.org/essays/tdd.html, November 2013.

[126] Webfaction. Hosting for developers. https://www.webfaction.com/, November 2013.

[127] Debian Wiki. Sudo. https://wiki.debian.org/sudo, August 2013.

[128] Khronos Wiki. Webgl and opengl. http://www.khronos.org/webgl/wiki/WebGL_and_OpenGL, November 2013.

[129] Khronos Wiki. Webgl frameworks. http://www.khronos.org/webgl/wiki/User_Contributions#Frameworks, November 2013.

[130] Wikipedia. Apache subversion. http://en.wikipedia.org/wiki/Apache_Subversion, November 2013.

[131] Wikipedia. Bring your own device. http://en.wikipedia.org/wiki/Bring_your_own_device, November 2013.

[132] Wikipedia. Concurrent versions system. http://en.wikipedia.org/wiki/Concurrent_Versions_System, November 2013.

[133] Wikipedia. Http persistent connection. http://en.wikipedia.org/wiki/HTTP_persistent_connection, November 2013.

[134] Wikipedia. "microsoft visio". http://en.wikipedia.org/wiki/Microsoft_Visio, November 2013.

[135] Wikipedia. Stateless protocol. http://en.wikipedia.org/wiki/Stateless_protocol, November 2013.

[136] Chad Williams. Building with django: In good company. http://www.fiveq.com/blog/programming/building-django-good-company/, April 2012.

Part IV

Appendix

Appendix A

Welldrilling

In this chapter we will give an general overview of the well drilling process, which is necessary to understand the situation that our system will assist. The following sections will describe how the actual drilling is done at sea (section A.1) and the three main planning parts in the software (section A.2).

A.1 Basic Introduction to Well Drilling

On Norwegian continental shelf, there are a large number of oil fields. Two important ones are Gullfaks and Ekofisk. There are multiple platforms in each field, and each of the platforms usually have multiple wells.

In Norway, there are around 300 operations where oil wells are constructed or repaired at any given time. Each operation include about 15 people, which sums up to about 4500 people. In other words, a formidable amount of Norwegian workers can be found in this part of the oil industry. It is toward these companies, and regarding these operations that this software will be targeted.

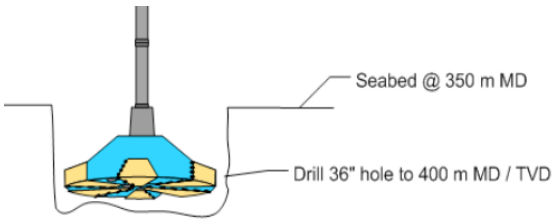


Figure A.1: A drilling head

Due to the big operational expenses and large financial risks linked to oil drilling, it is critical that a well is planned thoroughly before operation. The objective of the planning is to minimize the chances for unscheduled events, both under the making and the operation of the well. This is done by modelling drilling paths, materials and

forces that the well must endure. Iterations on paths and materials are done to improve the plan. In addition to this planning part, corrections on offset from plan during drilling operation are the main uses of this software.

When a plan is finished and approved, it will contain plans for the well during construction, operation and plugging (closing the well post-producing). The well will then be constructed as specified in the plan. Offsets from the plan needs to be identified and corrected during construction.

Drilling is done in sections since the formations will cave in like sand on the beach if you drill too deep. Each section is secured by lowering a casing and cementing it in place before the next section start. This allow us to get a "fresh start" at every section, where on need not think of the formations in the previous segments. Drilling a section at a time also allows us to simplify the work process into more or less independent stages.

A general, simplified procedure could be as follows (taken from [9]):

1. Drill a large hole (e.g. 36")
2. Run casing (e.g. 30") – this works as a mould in addition to structural support
3. Cement the 30" casing in place
4. Drill a smaller hole through the 30" casing, e.g. 24"
5. Run e.g. 20" casing.
6. Cement the casing in place
7. The 3 step procedure above is repeated until the target in the reservoir has been met/achieved

When repeating the steps, the well construction may resemble an extendable telescope. At the seabed, the top of the well is installed with a "well head", which has different security mechanisms to prevent blowout and spill. (See figure A.2)

A.2 Overview of Workflow

A.2.1 Contruct Well Path

The first part of the planning, is constructing a path for the well to follow. The path must obviously stay clear of any previously drilled wells or paths reserved for future wells. Curvatures and inclination should be minimized to reduce wear on equipment and reduction in drilling force. The main input of the user is depth, inclination and direction at certain points in the well. The program then uses a user-selected algorithm to construct a smooth transition in the path between those points.

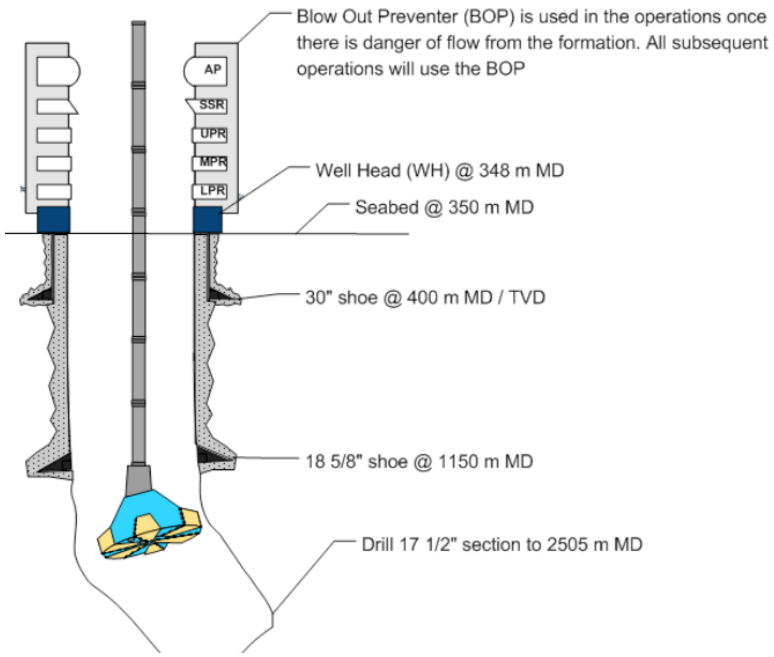


Figure A.2: An oil well

A.2.2 Work String Forces

The drill consists of a string that is lowered from a rig. At the end of the string, there is a drilling head (blue in figure A.3) that can be rotated to drill in a certain direction to change the direction of the well. This part of the workflow focuses on verifying that the path and equipment are compatible. Among other things, this includes making sure that the drill pipe, casings and tubing can reach the end point (called the target) and retrieved to the surface, that the work string won't break during operation and that cementing operation can be performed.

If the work string cannot make the path safely, a new path has to be considered, and/or the planned tools has to be changed. This is a process that is iterated a few times.

A.2.3 Casing and Tubing-Design

During the drilling, the work string will touch the walls, and wear on the installed casings. Heating and cooling will cause elongation and contraction, and pressure from liquids will also tear both the casings and the inner tubings. The purpose planning casing wear is to make sure that the planned material are of high enough grade for this wear, also under trying scenarios.

The bullet points below show what the purpose of this part of the software is:

- Verify that the installed casing can endure the planned operations.
- Verify the maximum pressures possible from the exposed formation.
- Verify that the casing can endure the pressure of the section. This is pressure-tested before each new section is drilled
- Verify if cuttings injection in the annulus (if planned) will collapse the casing.
- Verify that the pressure increase from heating of liquids will not collapse the casing.
- Verify that the pressure increase from heating of liquids will not collapse the tubing.
- Verify that the load from elongation/contraction of steel due to heating/cooling.

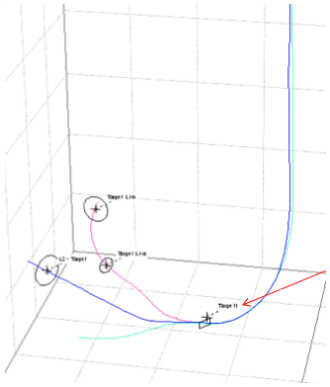


Figure A.3: A graphical representation of the planned well path in the current software

Appendix B

Supporting Documents

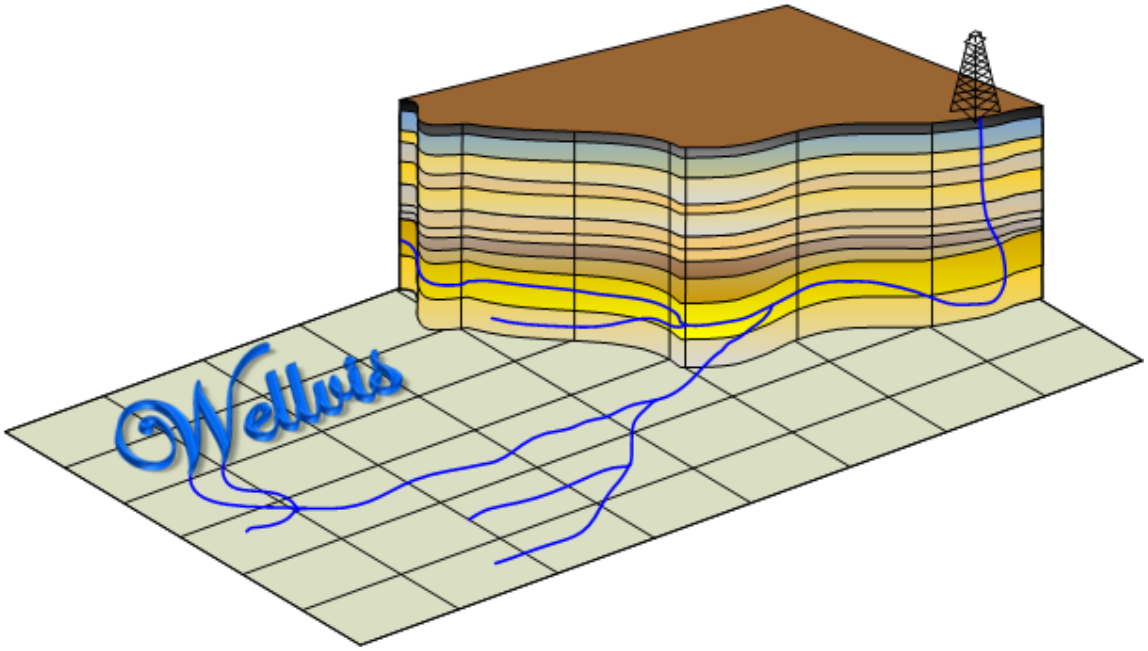
NTNU
Norwegian University of
Science and Technology

Basic Survey Calculation

Support for development of Well Path Software

Group 7 , TDT4290 "Customer Drive Project"

October 2013



Author: Bjorn Brechan

NTNU
Department of Petroleum Engineering
and Applied Geophysics

NTNU

Title:

Basic Survey Calculations

Support for development of Well Path Software
Group 7, TDT4290 “Customer Drive Project”

Assignment / scope:
Group 7 in TDT4290 “Customer Drive Project” Autmn 2013 chose to develop an application that constructs Well Paths for drilling oil and gas wells.

This document provides the basic theory and Math to construct well paths with 3D trajectory - Calculations to be used in the software developed by Group 7.

The basic theory for calculating well paths is the introduction to the nomenclature and parameters used by and in the equations for the 3D curvatures.

1 Summary

Searching for equations that develop precise well bore coordinates went from SPE8338 (M. Y. Planeix and R. C. Fox). They kicked off the method and call it “Exact Mathematical Formulation to plan 3D Directional Wells”. Their method was further developed by G. B Guo, S. Miska and R. L. Lee at New Mexico Institute of Mining and Technology, who documented this in SPE23576. These updated calculations are called “Radius-of-Curvature Method” and is discussed in detail in this document.

As offset, the Minimum Curvature Method is discussed. This method is not exact enough to develop and calculate coordinates for wells with 3D trajectories. However, in drilling operations, it is the preferred method when actual coordinates are logged by tools in the drill string.

The work performed includes a spread sheet to cover the methods calculations – “Planning 3D Trajectory - Method1 Complete – with Tangent.xlsx”. This sheet covers the scope set for functionality of the software to be developed by group 7. In this excel sheet, each of the three varieties of Method 1is entered and verified to work exact:

Method 1:
3D - Inclination / Azimuth / TVD

#	Name:	Input	Input parameters	Calculates
A	3D-M1 – Inc	Inclination	Build rate, Turn rate and Inclination	Azi, MD, TVD, NS, EW
B	3D-M1 – Azi	Azimut	Build rate, Turn rate and Azimut	Inc, MD, TVD, NS, EW
C	3D-M1 - TVD	TVD	Build rate, Turn rate and TVD	Inc, azi, MD, NS, EW

Table 1 - 3D Curvature Methods

Table of contents

List of figures:

List of Tables:

2 Acknowledgement

Would like to thank IDI for taking the assignment and especially Group 7, who have worked hard to achieve the prototype of Wellvis Well Path.

(Hope you had as much fun as us in Wellvis ☺ -and: should any of you need another assignment, please let us know)

Thank you!

On behalf of Wellvis,
Bjorn

3 Introduction

This document was written for the support of the work Group 7 (Autumn 2013) performed in TDT4290 "Customer Driven Project". Therefore, the content is written for non-drilling engineers (even though they catught the essence and consept of drilling very fast).

This document focuses on how to construct well paths with exact coordiantes – it explanes how a well path is constructed and calculated.

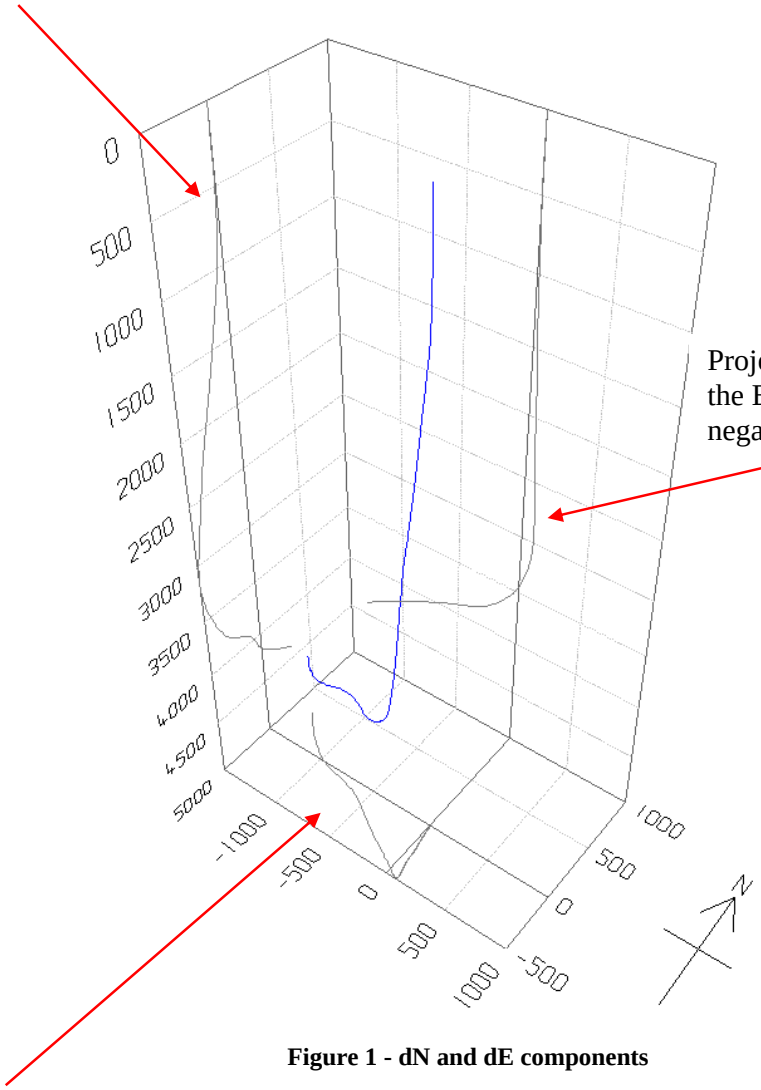
A Glossary with terms often used in surveying and directional drilling has been added in appendix.

3.1 Definitions

"Survey calculations are used to predict the position of the wellbore relative to the surface location"

- MD = Measured depth – Length of the wellbore measured by the drill string
- TVD = True vertical depth – Vertical component of the measured depth
- North = North component of the horizontal departure
- East = East component of the horizontal displacement
- Δ (delta) = the difference in…
- Cl = Course length (measured length - from … to…)
- $_{\text{Subscript}}$ 1 = The upper survey of two survey points
- $_{\text{Subscript}}$ 0 = The upper survey of two survey points (same as over)
- $_{\text{Subscript}}$ 2 = The lower survey of the two survey points
- No $_{\text{Subscript}}$ = The upper survey of two survey points (same as over – often used with $_{\text{Subscript}}$ 0. E.g I and I_0 – they are pairwise the same as I_1 and I_2)
- I = Inclination from vertical
- A = Azimuth of the survey (0 to 360 degrees)
- RC = Radius of curvature
- VS = Vertical section
- DLS = Dogleg severity
- DEP = The departure in the horizontal plane

Projection on this wall reflects the North component



Projection on this wall reflects the East component – here it is negative (westwards)

Figure 1 - dN and dE components

Projection on the floor reflects "Plan View"

There is another view often used: "Vertical section". Select a vertical plane (often the general direction of the well), and then project the well path on to it.

4 HSE&Q

Not many papers exist on how to develop exact calculations of well paths (coordinates) – the bulk of papers focuses on how to calculate a drilled well path and the accuracy of the coordinates.

Much engineering is perfumed before the well is drilled, so to have an accurate survey is important to get the right results, not only for anti collision but also for casing wear and other applications.

This work is a step in the direction of user friendly software with accurate wellbore coordinates.

For Group 7:

Thr author think you will find the information in here very simple and easy to grasp. And if you have doubts about that - always remember:

1) You have the Excel sheets with all calculations to look at if something looks uncomprehendable
2) You can always ask the author of the report for help / explanation
3) ...so don't worry ☺

5 Basics of constructing a well path

All well paths are developed from 1 fixed and known point to the next. This is also how they are measured to know exactly where the well is going during drilling / making the hole (well).

Normally the known starting point is the Well Head – where the well starts. For the purpose of simplicity, this will be coordinate 0 degrees north, 0 degrees East and 0 m deep (TVD)

From this point, the user of the software will start to develop the well path towards a target. This target will be something that the user receives from an external source (a group of people who are specialized in modeling the reservoir: "Subsurface Team" – often referred to as "the enemy").

The well path is then constructed 1 step at the time going from the known starting point (Well Head) towards the target (specific place in the reservoir given to the user as coordinates that can be expressed: ΔN, ΔE and ΔTVD from the reference point / Well head). There are a number of thing for a user to think about when making a well path, but it all comes down to being able to:

1. Create a tangent (vertically or deviated if following after a curvature).
2. Create a 3D curvature
3. Create a 2D curvature (same as above but with no change in either vertical or horizontal direction)

With these features available when constructing a well path, a user can go in any desired direction (avoiding problems) when going/drilling towards the target.

5.1 Some basic definitions

Any point on the wellbore can be described as a function of distance from the reference point as a set of coordinates: [X, Y Z] or more precise: [ΔN, ΔE, ΔTVD].

When discussing and developing the equations used to calculate coordinates for a 3D curve, the following applies for this document:

X = ΔE

Y = ΔE

Z = ΔTVD

Note from Figure 2 that there is a vector called "U", which is a tangent to the wellbore in the point S. Note also that the red arrows indicate the positive direction of each axis / coordinate - [ΔN, ΔE, ΔTVD].

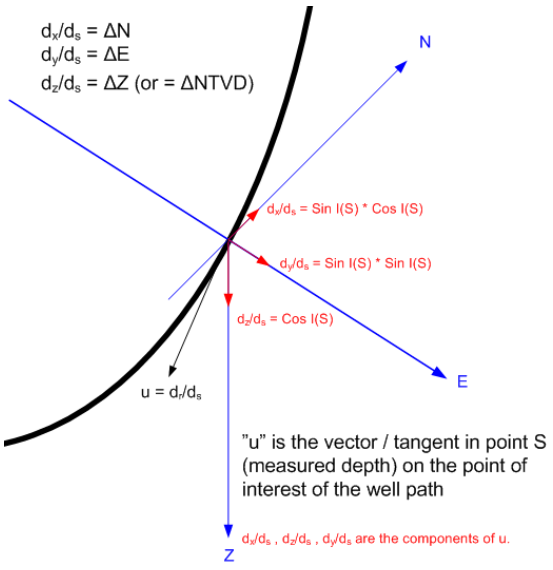


Figure 2 - Definition of coordinates

To better understand the further discussions regarding directions, it can help to imagine a situation looking down the well, with the nose pointing south. Then straight up will be north, to the right hand will be East and so forth. The reason for this simple exercise is to figure that looking down on the 3D well, it will be like a thread going from center and (often) curving in multiple directions. An example of this is the "floor" in figure 1, which is extracted in figure 3 below – the well goes first (almost) straight south, then mostly westwards while turning north.

In drilling, the North is defined as 0 ("and 360") degrees, East as 90 degrees, South as 180 and West as 270 degrees. So a better description of the well projection on the floor in the figure will then be first 180 degrees, then 300 degrees.

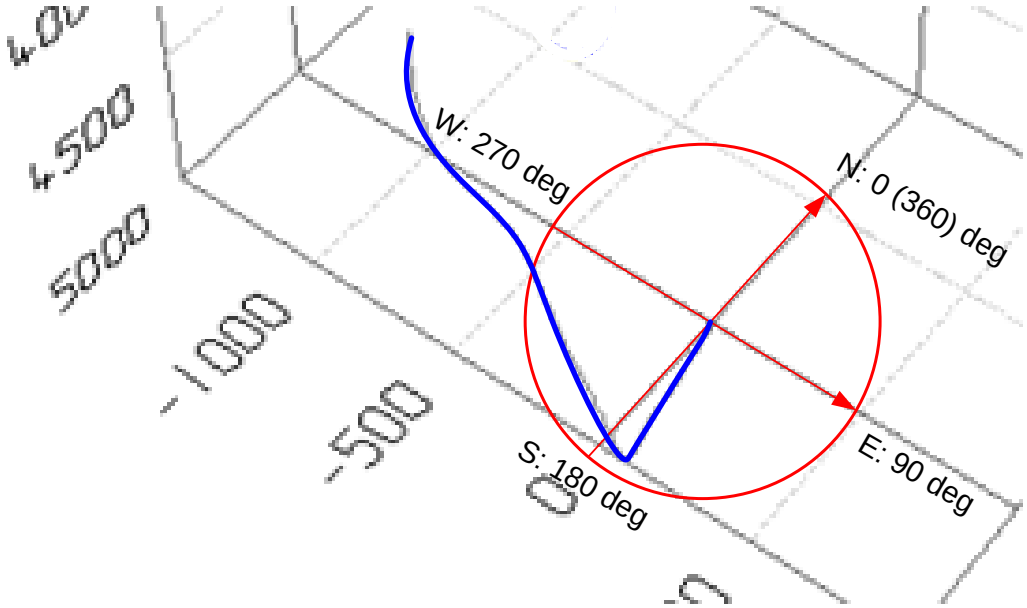


Figure 3 - Degrees and compass directions

After this rather awkward exercise, it can be easier to understand that e.g. adding azimuth angles should not be more than 359,99° or less than 0°. The logic / math must catch when e.g. compass direction of the well goes from 10° to 350° that it is a 20° turn west and (normally) not a 340° east.

The logic developed for this in the supporting spread sheet, covers ~90% of the cases occurring. But for the M1 TVD case, Excel will develop a circular reference at any attempt to adjust (force) the turn west or east.

This must be considered an in introduction for the chapter discussing the calculations.

..and Z or TVD is positive downwards.

Description of Inclination and Azimuth:

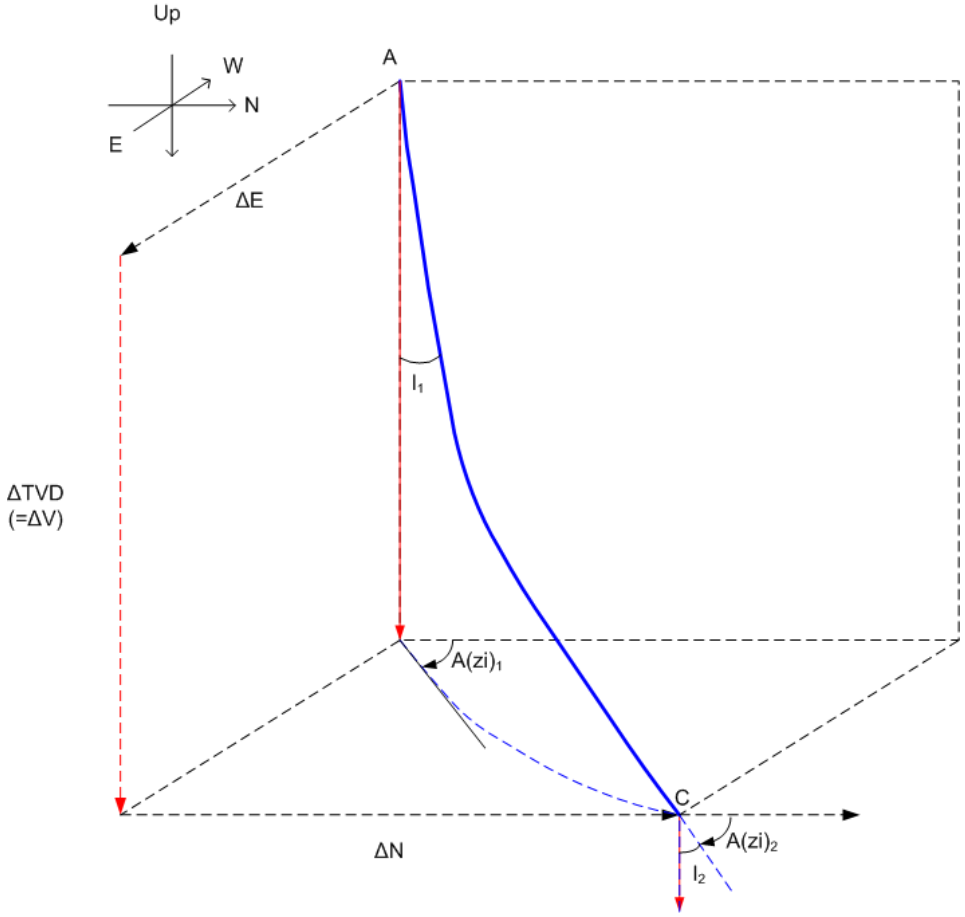
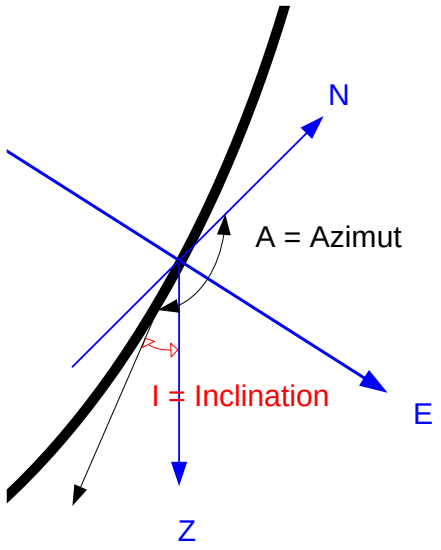


Figure 4 - Definition: Inclination and Azimuth



6 The Radius-of-Curvature Method

Repeating figure 2, now with focus on the equations and their significance.

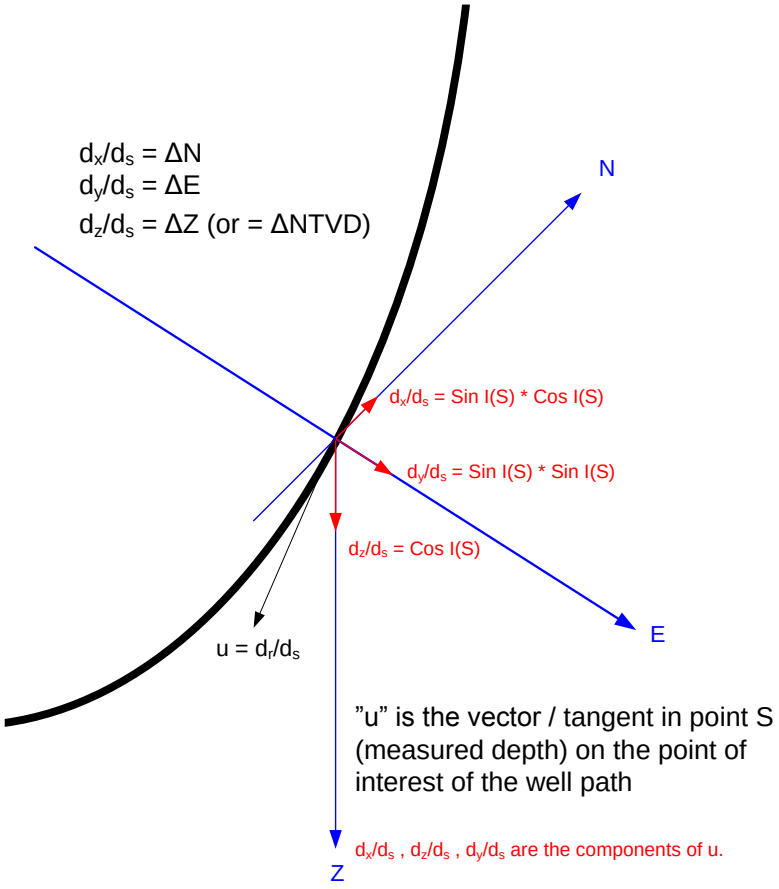


Figure 5 - Unit vectors: X, Y and Z

The tangent in point S is described by coordinates. These coordinates are the "decomposing" unit vectors - d_x/d_s, d_y/d_s and d_z/d_s - along the N, E and Z axis.

So, integrating the below listed expressions for each of these unit vectors given in SPE 23576 will yield the equations given on the next page.

North:	d_x/d_s = Sin I(S) * Cos I(S)
East:	d_y/d_s = Sin I(S) * Sin I(S)
Z (dTVD):	d_z/d_s = Cos I(S)

Where (S) means Length (nomenclature adopted from the SPE paper) – i.e. integrating to get the coordinates going from one point on the curve (well path) to the next.

This is the expression describing the North coordinate:

$$\int_{S_0}^{S} \frac{\mathrm{dx}}{\mathrm{ds}} = F_x(S) = \frac{1}{T^2 - B^2} * \{T * (Sin\,[I] * Sin[A] - Sin\,[I_0] * Sin[A_0]) + B * (Cos\,[I] * Cos[A] - Cos\,[I_0] * Cos[A_0])\}$$

Eq 1

This is the expression describing the East coordinate:

$$\int_{S_0}^{S} \frac{\mathrm{dy}}{\mathrm{ds}} = F_y(S) = \frac{1}{T^2 - B^2} * \{-T * (Sin\,[I] * Cos[A] - Sin\,[I_0] * Cos[A_0]) + B * (Cos\,[I] * Sin[A] - Cos\,[I_0] * Sin[A_0])\}$$

Eq 2

This is the expression describing the Z coordinate:

$$Z = Z_0 + \frac{1}{B} * (Sin[I] - Sin\,[I_0])$$

Eq 3

Remember that the point calculated from is always known. So, if the inclination in the end point and the B (build rate [deg/30m]) is known, the Z coordinate can be determined.

6.1 Other equations

Another definition that needs explaining is the term “Dogleg” [°]. This is means “curve” and is often in 3D. When drilling, often the first part of the well starts with a vertical section (until the formation is firm enough to withstand the extra strain from a deviated hole). From there, the well path is directed towards the target. To go from vertical we build angle from vertical. This is called B for Build Up Rate (BUR). This will create a curve.
If there is a need to go in a different compass direction, the well path will be turned. To change compass direction will also create a curve. This is called T for Turn Rate.

When Turning and Building at the same time (when drilling), a 3D curved well path will be made. The effective dog leg of this 3D curve can be seen in the figure below:

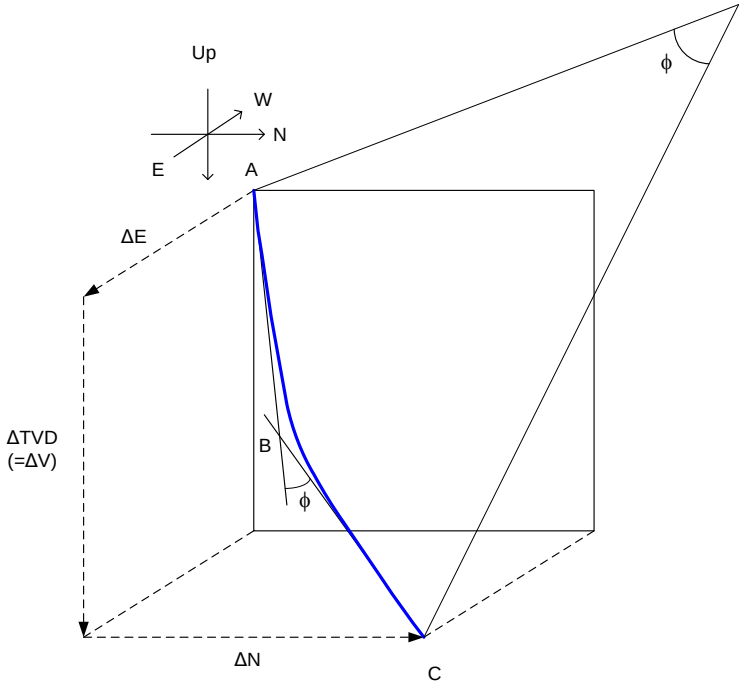


Figure 6 - Dogleg angle

ϕ is the dog leg angle.
Note that the curve is bent and cannot be decomposed in an angle like ϕ for the Azimuth (compass) / Turn and for the increase in inclination / Build. Any such method will result in inaccurate coordinates when going from point A to C as shown in figure 6.

For people in drilling to understand how much the well path is changing direction (3D), the term Dogleg must be defined to a length. This results in a new definition: Dogleg severity [°/30m] – often abbreviated to DLS. So it is simply the ϕ (dog leg angle) divided into lengths of 30 meter:

DLS = (ϕ * 30)/CL , where CL is the physical length (measured depth - MD) of the curvature.

Equation to describe the relationship between T (Turn) and B (Build) exists:

$DLS = Sqrt[B^2 * T^2 * Sin^2(I_2)]$

To describe the similar angle as ϕ for 3D curvatures for Turn and Build alone, the term Radius of Curvature (RC) is shown in the figure below as θ_I (I for Inclination):

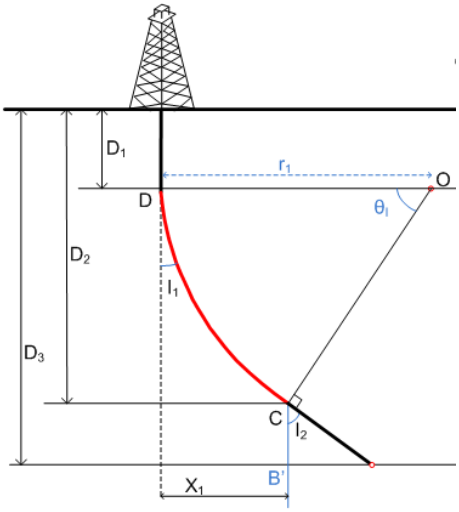


Figure 7 - Radius of curvature - RC

The formula for RC_I (marked as r_1 in figure 7) is:

$$RC_I = (180 * 30) / (\pi * B)$$

Where B is the Build Rate.

The same expression can be used for a turn (Azimuth angle):

$$RC_A = (180 * 30) / (\pi * T)$$

Where T is the Turn Rate.

To calculate the physical length (Course Length – CL) of a curvature:

$$CL = [RC * \pi (I_2 - I_1)]/ 180$$

6.2 How to Calculate M1 – Inclination

Input is:

a) Starting point – all info known (Inc, Azi, Z and dN, dE)
b) User input for end point:
    a. Inclination in end point
    b. Turn rate T
    c. Build rate B

Step 1: Calculate RC_I – notice the excel formula in the top.

F19 =180*100/H29/(PI()*B11)

	A	B	C	D	E	F	G
1							
2					T and Azimuth-support:		
3	NOTE: B different from T!				Turn west => dA =	Turning East	
4					Turn east => dA =	180	
5	INPUT:				USE THIS dA:	180	
6							
7		12,51	Azi1				
8		5	Inc1				
9	I_2	65	Inc2			DL	Deg/m
10	T	3	Turn Rate/30m ====>			180	0,1
11	B	1	BUR/30m =====>			60	0,03333333
12		3,310134	H_{az}/30m ====>			198,6080254	0,11033779
13		2,896989	DLS - Wilson (from BUR & T)			173,8193294	0,09656629
14							
15	CALCULATIONS:						
16	dMD:						
17		CL=	5905,5118	[ft]	dMD=2π*RC*(inc2-inc1)/360		
18			1800	[m]			
19					RC	5639,348377	foot
20	Azi2:					1718,873385	m

$RC_I = (180 * 30) / (\pi * B)$

With RC_I known:

Step2: Calculate CL

C17 =F19*2*PI()*(B9-B8)/360

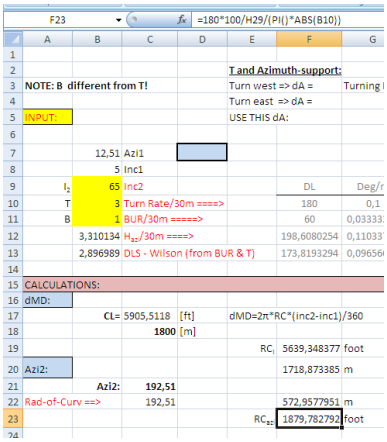
	A	B	C	D	E	F	G
1							
2					T and Azimuth-support:		
3	NOTE: B different from T!				Turn west => dA =	Turning	
4					Turn east => dA =		
5	INPUT:				USE THIS dA:		
6							
7		12,51	Azi1				
8		5	Inc1				
9	I_2	65	Inc2			DL	Deg/
10	T	3	Turn Rate/30m ====>			180	0,1
11	B	1	BUR/30m =====>			60	0,0333
12		3,310134	H_{az}/30m ====>			198,6080254	0,1103
13		2,896989	DLS - Wilson (from BUR & T)			173,8193294	0,0965
14							
15	CALCULATIONS:						
16	dMD:						
17		CL=	5905,5118	[ft]	dMD=2π*RC*(inc2-inc1)/360		
18			1800	[m]			
19					RC_I	5639,348377	foot
20	Azi2:					1718,873385	m
21		Azi2:	192,51				
22	Rad-of-Curv ==>		192,51			572,9577951	m
23					RC_{az}	1879,782792	foot

185

$CL = [RC * \pi (I_2 - I_1)]/ 180$

With CL known:

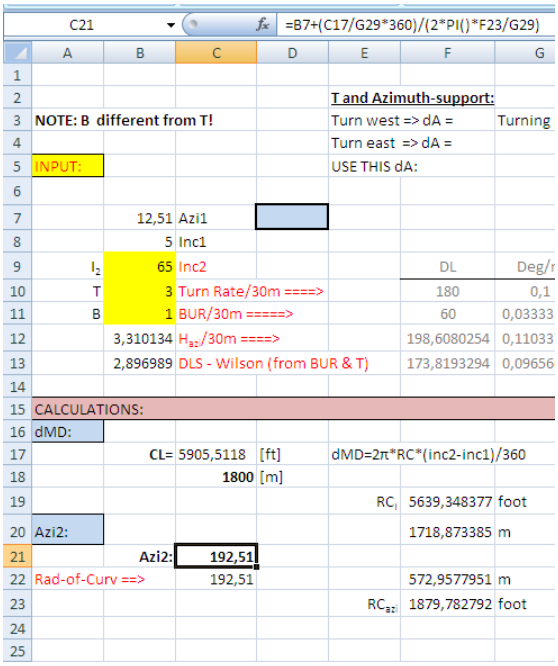
Step 3: Calculate RC_A



$RC_A = (180 * 30) / (\pi * T)$

With RC_A known:

Step 4: Calculate the Azimuth in the end point - Azi_2 :



$Azi_2 = Azi_1 + [(CL*180) / (\pi * RC_A)]$

Now you are ready for step 5:

This step prepares the input for the equations 1 through 3 (on page 15) that calculates the dN, dE and dZ. First for Turn rate T:

a) Convert from [°/30m] to get the Dogleg [°]: $DL_T = T*CL / 30$
where CL is the Course Length [m]

b) Calculate the radian of the DL_T [°] ==> DL_T [Radians]

c) Finally, normalize the DL_T for meters (divide by the Course Length of the curve):

DL_T/CL [Radians/m]

Then do the same for Build rate B:

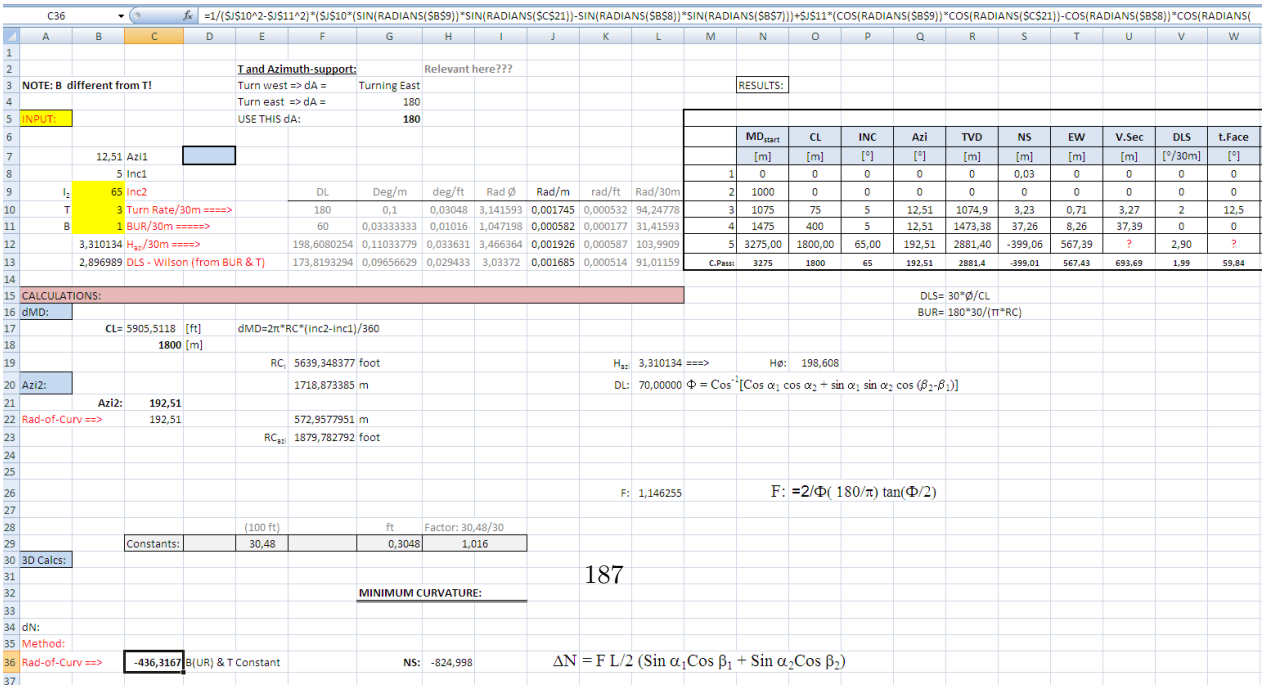
a) Convert from [°/30m] to get the Dogleg [°]: $DL_B = B*CL / 30$
where CL is the Course Length [m]

b) Calculate the radian of the DL_B [°] ==> DL_B [Radians]

c) Finally, normalize the DL_B for meters: DL_B/CL [Radians/m]

Now for the final steps – calculate the dN, dE and dZ:

All input are above for the different coordinates. An example for dN is attached below:



Comparing the equations for dN:

$$\int_{S_0}^{S} \frac{\mathrm{dx}}{\mathrm{ds}} = F_x(S) = \frac{1}{T^2 - B^2} * \{T * (Sin\,[I] * Sin[A] - Sin\,[I_0] * Sin[A_0]) + B * (Cos\,[I] * Cos[A] - Cos\,[I_0] * Cos[A_0])\}$$

=1/(J10^2-J11^2)*(J10*(SIN(RADIANS(B9))*SIN(RADIANS(C21))-
SIN(RADIANS(B8))*SIN(RADIANS(B7)))+J11*(COS(RADIANS(B9))*COS(RADIANS(C21))-
COS(RADIANS(B8))*COS(RADIANS(B7))))

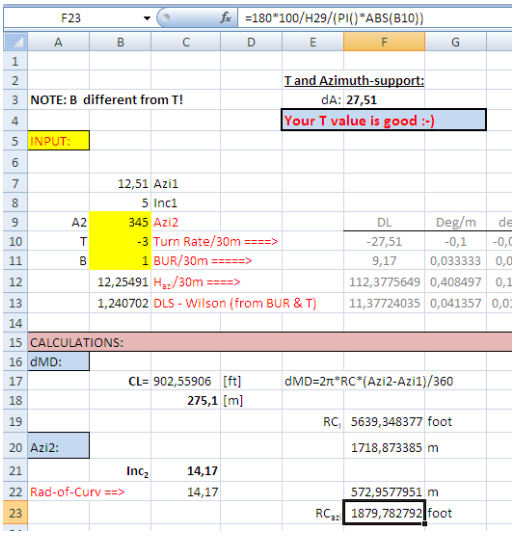
1) Turn and Build are user inputs
2) I is here the Inclination in the new point (same as I_2, nomenclature is adopted from the SPE paper)
3) A is here the Azimuth in the new point
4) I_0 and A_0 refers to same but in the starting point (known values, they are the end point of the last section calculated)
5) Using the right units is important – B and T should enter the calculations in [Radians/m] to get the dN contribution (going from the known point to the end of the curvature) in meters. [Radians/ft] will give the same in ft.
6) Remember: since this is the dN, it should be added / accumulated with the N-coordinate in the starting point (N_0) to get the N-coordinate in the new point: $N = N_0 + dN$
7) **NOTE:** A dedicated chapter has been prepared for discussing sensitivities and considerations to parameters and results. Once you feel you have an overview of this / the methods / the theory, it is a good idea to have a look at it. It will influence the (calculated) results.

6.3 How to calculate M1 – Azimuth

Input is:

a) Starting point – all info known (Inc, Azi, Z and dN, dE)
b) User input for end point:
    a. Azimuth in end point
    b. Turn rate T
    c. Build rate B

Step 1: Calculate RC_A

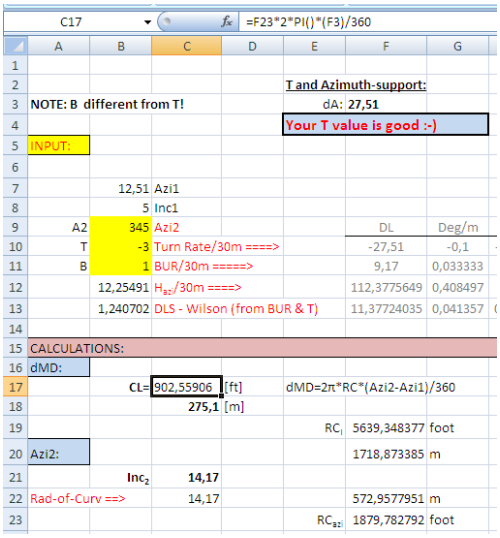


$RC_A = (180 * 30) / (\pi * T)$

(Sorry for some of the calculations being in [ft] – they could all be in [m] – something that dvels from searching for the right units and getting the calculations correct)

With RC_A known:

Step2: Calculate CL

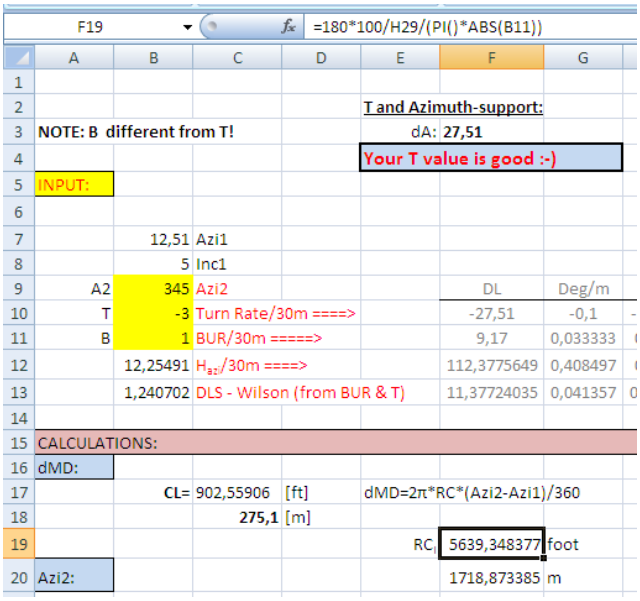


189

$CL = [RC * \pi (I_2 - I_1)]/ 180$

With CL known:

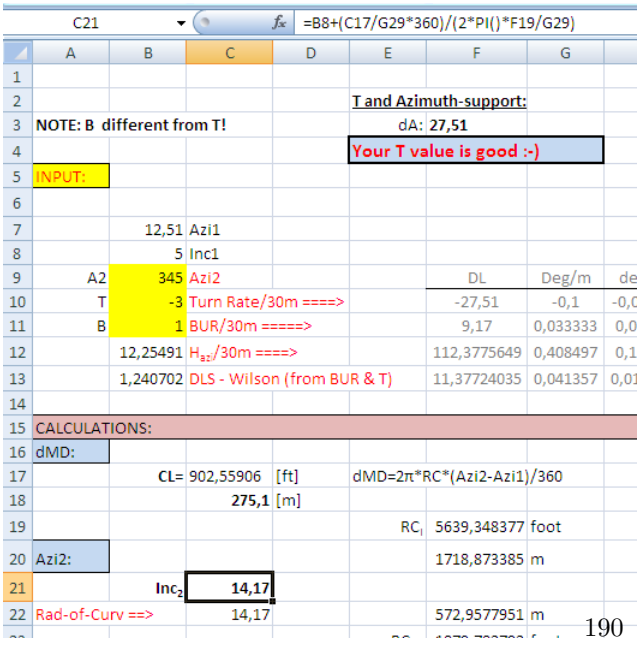
Step 3: Calculate RC_I



$RC_I = (180 * 30) / (\pi * B)$

With RC_I known:

Step 4: Calculate the Inclination in the end point - Inc_2 :



$Inc_2 = Inc_1 + [(CL*180) / (\pi * RC_I)]$

Now you are ready for step 5, which is a repetition of same step in M1 Inc.

This step prepares the input for the equations 1 through 3 (on page 15) that calculates the dN, dE and dZ. First for Turn rate T:

a) Convert from [°/30m] to get the Dogleg [°]: $DL_T = T*CL / 30$
where CL is the Course Length [m]
b) Calculate the radian of the DL_T [°] ==> DL_T [Radians]
c) Finally, normalize the DL_T for meters (divide by the Course Length of the curve):

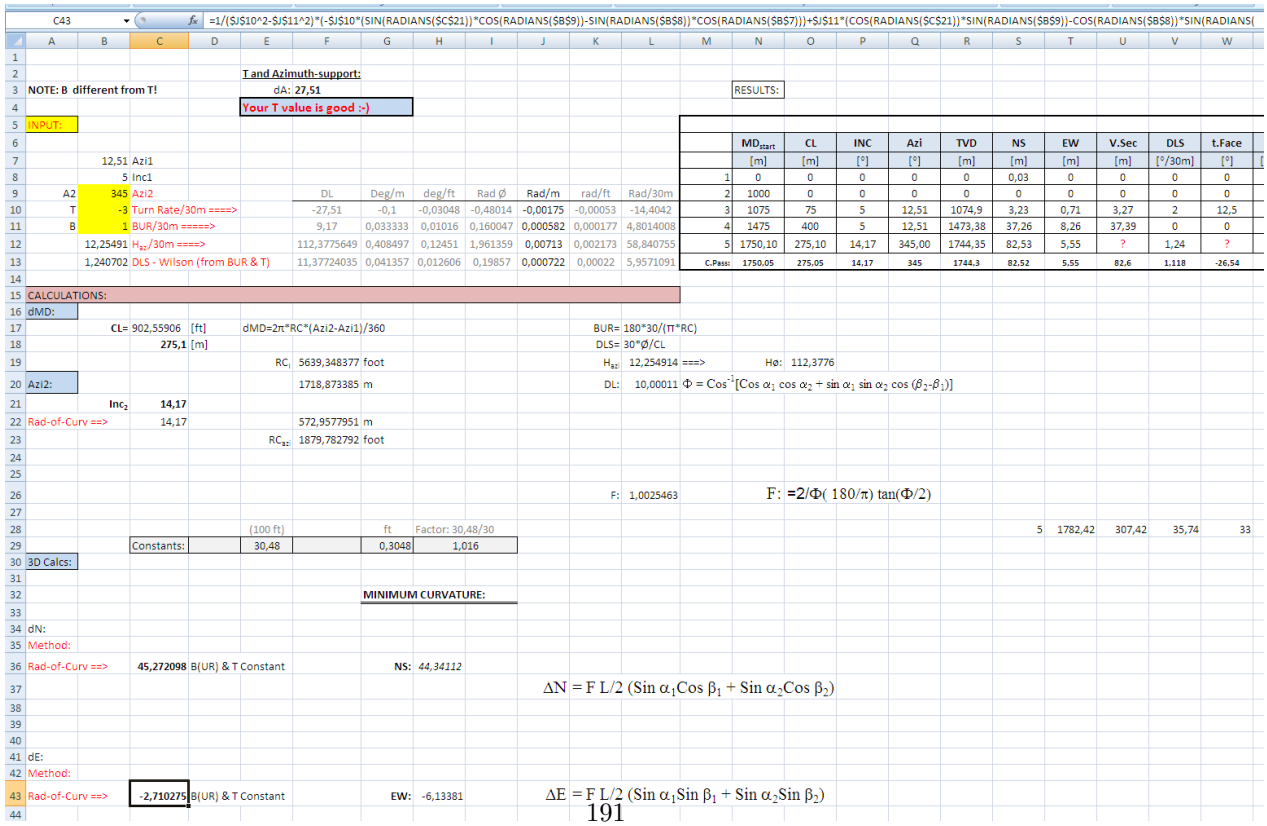
DL_T/CL [Radians/m]

Then do the same for Build rate B:

a) Convert from [°/30m] to get the Dogleg [°]: $DL_B = B*CL / 30$
where CL is the Course Length [m]
b) Calculate the radian of the DL_B [°] ==> DL_B [Radians]
c) Finally, normalize the DL_B for meters: DL_B/CL [Radians/m]

Now for the final steps – calculate the dN, dE and dZ:

All input are above for the different coordinates. An example for dE is attached below:



	MDstart	CL	INC	Azi	TVD	NS	EW	V.Sec	DLS	t.Face
	[m]	[m]	[°]	[°]	[m]	[m]	[m]	[m]	[°/30m]	[°]
1	0	0	0	0	0	0,03	0	0	0	0
2	1000	0	0	0	0	0	0	0	0	0
3	1075	75	5	12,51	1074,9	3,23	0,71	3,27	2	12,5
4	1475	400	5	12,51	1473,38	37,26	8,26	37,39	0	0
5	1750,10	275,10	14,17	345,00	1744,35	82,53	5,55	?	1,24	?
C.Pass:	1750,05	275,05	14,17	345	1744,3	82,52	5,55	82,6	1.118	-26,54

DL	Deg/m	deg/ft	Rad Ø	Rad/m	rad/ft	Rad/30m
-27,51	-0,1	-0,03048	-0,48014	-0,00175	-0,00053	-14,4042
9,17	0,033333	0,01016	0,160047	0,000582	0,000177	4,8014008
112,3775649	0,408497	0,12451	1,961359	0,00713	0,002173	58,840755
11,37724035	0,041357	0,012606	0,19857	0,000722	0,00022	5,9571091

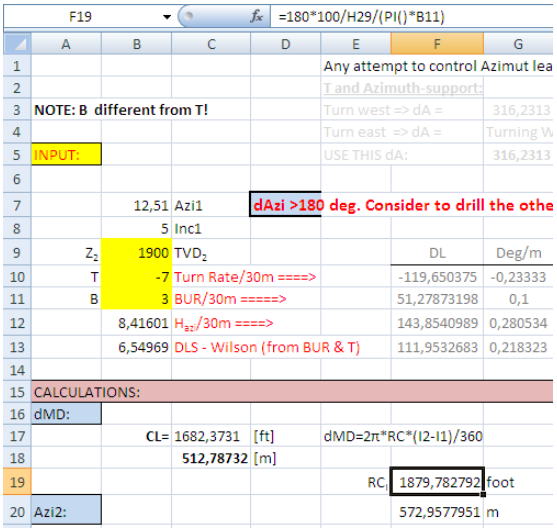
5	1782,42	307,42	35,74	33	

6.4 How to calculate M1 – TVD

Input is:

a) Starting point – all info known (Inc, Azi, Z and dN, dE)
b) User input for end point:
    a. TVD in end point
    b. Turn rate T
    c. Build rate B

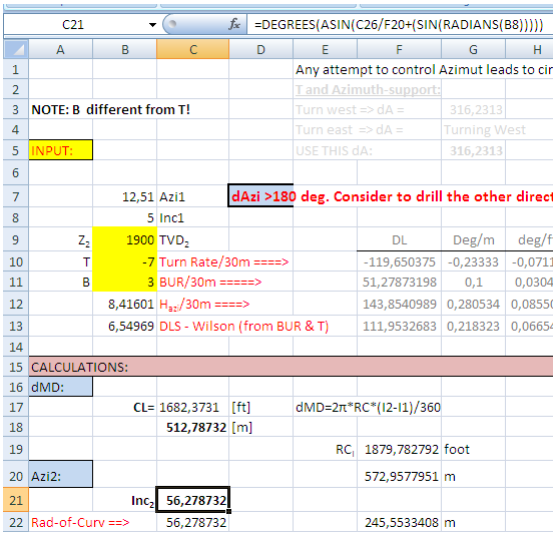
Step 1: Calculate RC_I



$RC_I = (180 * 30) / (\pi * B)$

With RC_I known:

Step2: Calculate the Inclination in the end point - Inc_2 :

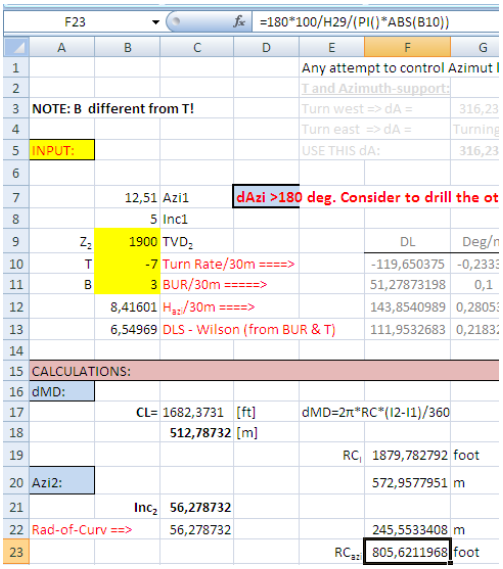


192

$Inc_2 = Sin^{-1}[(TVD2 – TVD1) / RC_I + Inc_1]$

With Inc_1 known:

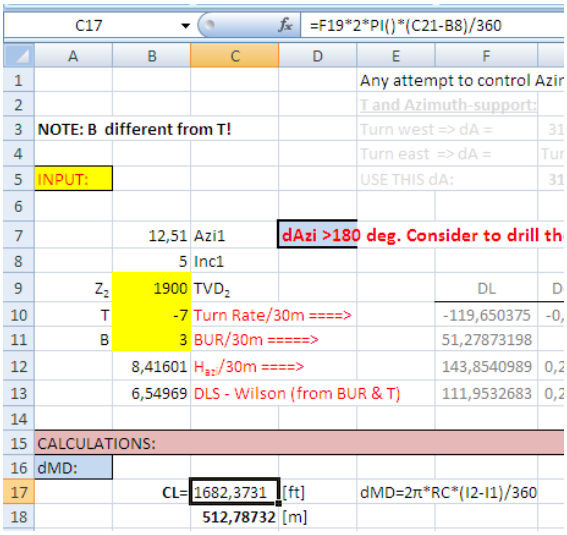
Step 3: Calculate RC_A



$RC_A = (180 * 30) / (\pi * T)$

With RC_A known:

Step 4 Calculate CL



$CL = [RC * \pi (I_2 - I_1)]/ 180$

With CL known:

Now you are ready for step 5, which is a repetition of same step in M1 Inc and M1 Azi.

This step prepares the input for the equations 1 through 3 (on page 15) that calculates the dN, dE and dZ. First for Turn rate T:

a) Convert from [°/30m] to get the Dogleg [°]: $DL_T = T*CL / 30$
where CL is the Course Length [m]

b) Calculate the radian of the DL_T [°] ==> DL_T [Radians]

c) Finally, normalize the DL_T for meters (divide by the Course Length of the curve):

DL_T/CL [Radians/m]

Then do the same for Build rate B:

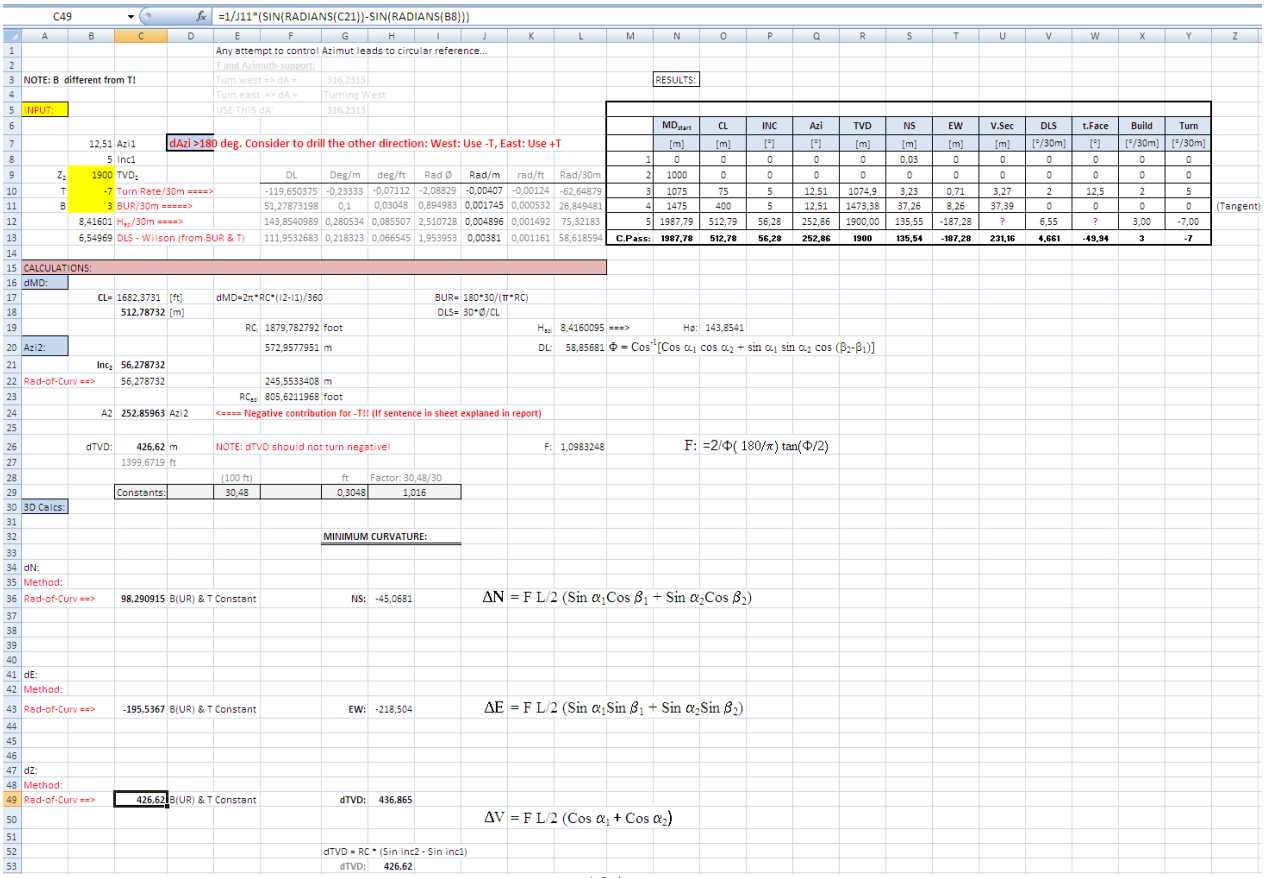
d) Convert from [°/30m] to get the Dogleg [°]: $DL_B = B*CL / 30$
where CL is the Course Length [m]

e) Calculate the radian of the DL_B [°] ==> DL_B [Radians]

f) Finally, normalize the DL_B for meters: DL_B/CL [Radians/m]

Now for the final steps – calculate the dN, dE and dZ:

All input are above for the different coordinates. An example for dZ is attached below:



7 Parameters sensitivities and considerations

Implementing the above sequences for calculating the coordinates for a 3D curve segment of a well path will give accurate results. However, the calculations as they stand will get you in trouble – as all math and logic if not used with caution.

7.1 Angles when turning

As discussed in chapter 5.1 (page 12 of this report), the Azimuth should stay above 0° and below 360°. Also, there should be some logic to the user input: If the user wants to turn to the west, the turn rate T should be negative. There should be no override programmed, so that the well path always takes the shortest route. If a user is planning a large turn due to a constraint or formation obstacle, it would be annoying to nudge the well path in many steps towards the target. Therefore:

Positive T: Always turn with the clock or more correct: Eastwards (from north)
Negative T: Always turn counter clockwise or more correct: Westwards (from north)

This will easily get you in trouble if you pass north from either side. Therefore, some logic to the true dAzi going from start point to end point should be applied.

These are the 2 exceptions that need special attention:

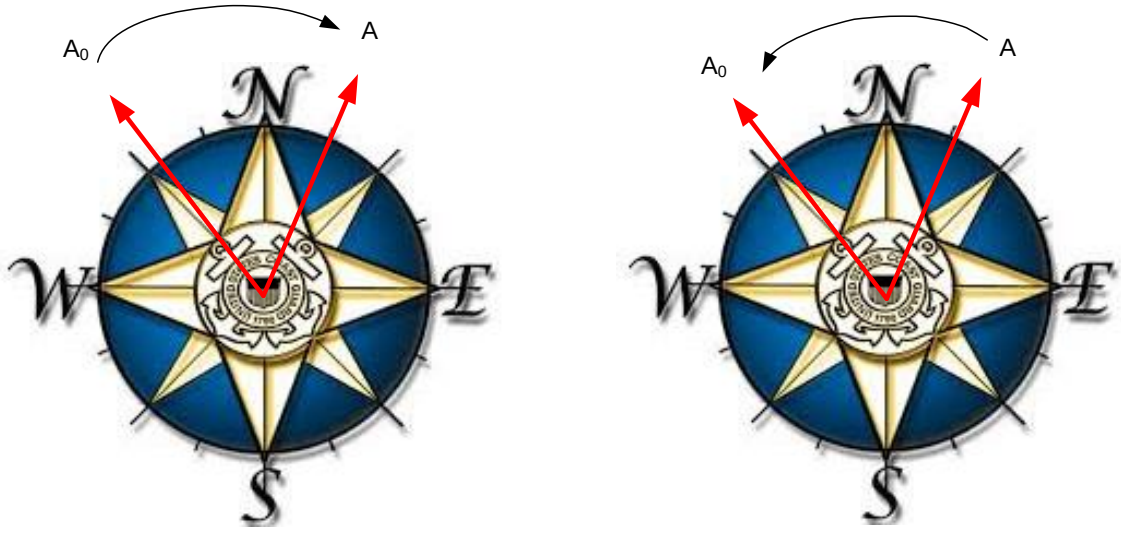


Figure 8 - Exemptions, compass directions

So the logic (attempted to be) applied in the Excel sheet is presented below, but may be programmed differently if easier.

For T (East):
The normal case is that A > A_0. Therefore there is an "If-sentence" verifying if the Turn is positive first, then if the A < A_0 (it will be the special case to the left in figure 8 above, and) the calculations that are normally dA = A - A_0 changes to dA = 360 - A_0 + A.

For –T (West):
Basically the same as above – just reversed.
The normal case is that A_0 >A. Therefore there is an "If-sentence" catching the direction of the turn (confirmed to be west), then if the A_0 <A (it will be the special case to the right in figure 8 above, and) the calculations that are normally dA = A_0 - A changes to dA = 360 – A + A_0.

7.2 Turn and Build

You may have noticed that for T = B will get you into trouble since the formulas for dN and dE all start with 1 / (T^2 – B^2) which then becomes Div-0 in Excel. One way it could be caught and handled while programming:

If user input is T = B: e.g. T=2 and B=2
Let the T (or B) change with +/- 0,01 (or smaller?). Then use / report the average of the 2 results. This means going through the calculations twice and there may be an easier way. However, this method has been verified to work.
It was verified using 1,999 for either and the results became "identical" with the offset comparison calculations (i.e. the results were not averaged and still found acceptable).

One thing to note about B (Build Up Rate – BUR) is that it can become negative. This means that the well path will (create a) drop. In the offset model / calculation, this seems to be handled sub optimally. The calculations easily turn into "unable to reach TVD", which seems a bit strange. The logic, as the author sees it, should be that the well path automatically stops to drop when inclination is 0, which would be the goal of the user. Then, if the 0 degree inclination depth is shallower than the user input, this should be the end of the curvature and the user can add a tangent from there.

So there is a challenge – logic must be built to catch the depth when the well path is turning to 0 deg inclination and the stop there – not abort calculations. The user

7.3 Angles for inclination

Inclination cannot be outside [0°, 360°>. The same logic as for Azimuth does not apply for inclination, which is in a vertical plane. The normal operating range is [0°, 90°], however, when drilling horizontally along a formation, the well path may rise above the horizontal and reach inclinations of 130°. Anything close to or more than 180° (straight upwards!) does seem very strange to anyone in drilling and is highly unlikely to be drilled.

Then, drilling more than 180° Inclination will imply that the well path must have made a loop. This will not go well in operations (forces). Some error message should appear when angles higher than 130° are calculated, but as for Azimuth, a hardline is the [0°, 360°>.

7.4 2D curvatures

This has not been tested in the excel sheet, however, the suggestion is that either B or T is set to 0. In the supporting excel sheet, this does not work because the result is “#Div 0” for some cells.

Again the suggested solution is to set the B or T that is 0 to +/- 0,001 and average the result. And as described in Chapter 7.2, this calls for the calculation to be done twice. Group 7 may find a better way of solving this when programming, but with Excel there is not.

8 Reporting

Reporting should be standardized with the following:

a) User interface for catching the input per method and variety
b) The different methods should be repeatable so the well path can be calculated in segments
c) Calculated results – in a table
d) Graphical visualization of the well path
e) Printing survey listings should be enabled

8.1 User interface

The user interface to input the different steps making the well path could look something like this:

	MD	CL	Inc	Azi	TVD	N	E	DLS	Build	Turn
	[m]	[m]	[◦]	[◦]	[m]	[m]	[m]	[◦/30m]	[◦/30m]	[◦/30m]
1										
2										
3										
4										
5										
(etc)										

Table 2 – User input

It would be good if this matrix would allow the user to segments to the well path manually by entering numbers in the cells:

- TVD refers to the "Z" in the first test model made by group 7
- N refers to "X" (or y) in the first test model made by group 7
- E refers to "Y" (or x) in the first test model made by group 7

If this is work-intesive, it should not be followed up. Segments natural to plan in this way would be tangents. This could easily be added by catching input and running through calculations such as for the 2D and 3D curvatures.

Below is an example of a plan for a well path calculated with 5 segments.

	MDstart	CL	INC	Azi	TVD	NS	EW	V.Sec	DLS	t.Face	Build	Turn
	[m]	[m]	[°]	[°]	[m]	[m]	[m]	[m]	[°/30m]	[°]	[°/30m]	[°/30m]
1	0	0	0	0	0	0,03	0	0	0	0	0	0
2	1000	0	0	0	0	0	0	0	0	0	0	0
3	1075	75	5	12,51	1074,9	3,23	0,71	3,27	2	12,5	2	5
4	1475	400	5	12,51	1473,38	37,26	8,26	37,39	0	0	0	0
5	1750,10	275,10	14,17	345,00	1744,35	82,53	5,55	?	1,24	?	1,00	-3,00
Offset	**1750,05**	**275,05**	**14,17**	**345**	**1744,3**	**82,52**	**5,55**	**82,6**	**1,118**	**-26,54**	**1**	**-3**

Table 3 - Example of calculated segments

Table 2 is taken from the Excel support sheet, which contains the results from an offset model at the bottom line using the same input. All 3 methods have the same. They have been tested for 7 to 15 different scenarios each, and should be as precise as described in the table. The offset results and the calculations are there for comparison to results in Group 7s' programmed model / software. Also, in appendixs there is a full listing of 3 wells for Group 7 to compare and calibrate results with.

Some things to note:

1) The author of this report has not looked into how exact calculations work for vertical section. These will be added to this report, which will be revised with this info later. It is not essential for Group 7 to add this to the model / software.
2) The same goes for Tool face.
3) DLS: the offset model calculates this inaccurate. How it has been decided for the scope of Group 7s' model / software, the DLS is not a direct user input. It could be instead of B and T, which can be deducted from DLS (refer to DLS = Sqrt[B^2 * T^2 * Sin^2(I_2)]).

8.2 Repeatable calculations – Wells planned in segments

As with the early prototype presented by Group 7, it is necessary to use all 3 methods for 3D (and 2D) calculations – in addition to adding tangents.

As with the Group 7 Prototype, lines would contain the results from calculations of each segment of the well as it develops towards the target.

Example of a typical plan:

Line 1: Vertical to e.g. 1000 m (dN = 0, dE =0) CL: 1000m
Line 2: 3D turn and build towards the target – in preparation for the next line CL: 200 m
Line3: Tangent, "transport segment" (at a Sail Angle) CL: 1500 m
Line4: Adjusting direction to land out correct - enter target CL: 200 m
Line5: Drilling reservoir to target(s). This could comprise multiple lines to enter all targets

Line 1 and 3 would use the tangent calculation, and line 2 and 4 would use the 3D curvature calculation (one or more methods).

Note that the excel spread sheet only calculate 1 line and there is a sheet for each of the 3 calculation varieties.

8.3 Graphical visualization of the well path 199

Group 7s' visualization of the well path under construction is good – nothing to add at this time. The report "System Specification" will be updated with the interface made.

8.4 Reporting

Once a plan for a well path has been established, many engineering calculations will be performed using the coordinates of the well path. Displaying and printing listing of coordinates such as MD, CL, INC, Azi, TVD, NS, EW and DLS are frequently used. It is a hope that this could be in place in Group 7s model once finished.

Also exporting these data to send to engineers in 3rd party companies supplying equipment and services are frequently done – a format of common use that is easy to import and export is preferable (ascii?).

9 Tangents

Tangents are great! Inclination and Azimuth are constant per definition, which simplifies the math.

Equations are as follows:

$$dN = dMD * Sin(I) * Cos(A)$$

$$dE = dMD * Sin(I) * Sin(A)$$

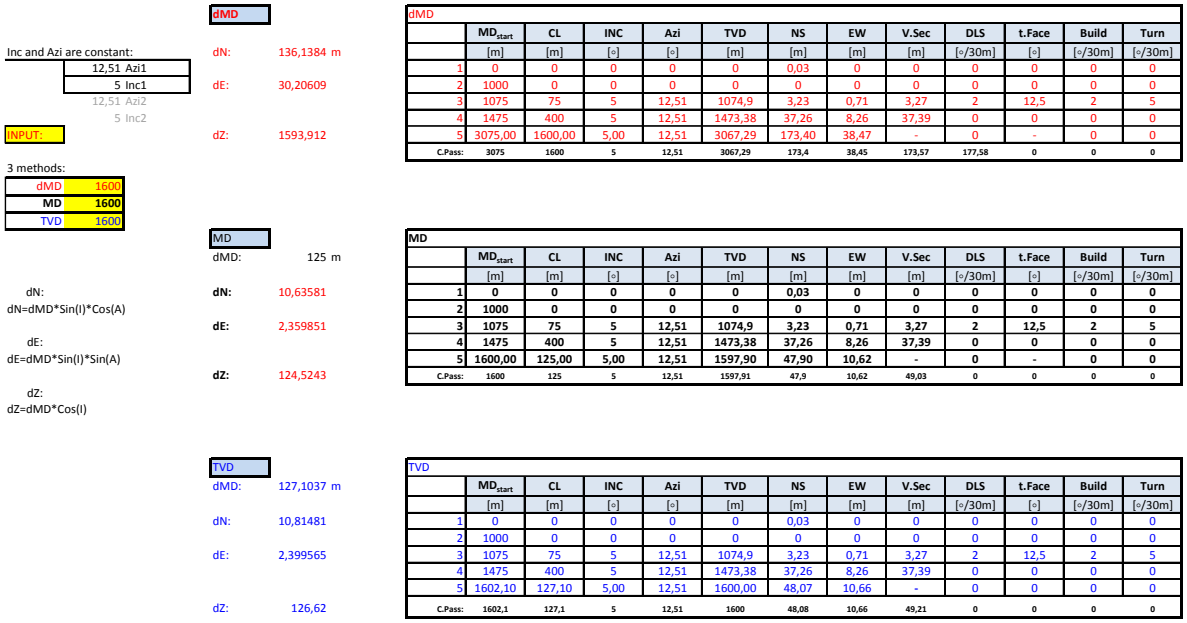
$$dZ = dMD * Cos(I)$$

Methods could be 3 – as per defined by possible user input:

1) Enter CL (or sometimes referred to as dMD)
2) Enter new MD
3) Enter new TVD

These 3 methods are shown in the spread sheet (same page) – see figure 9 on the next page. The math is very simple and can be seen in the spread sheet. In general – just remember to accumulate the values for the coordinates and depths.

Some error catching:
The user should not be allowed to enter a depth that is shorter than the existing plan.



dMD

	MD_{start} [m]	CL [m]	INC [°]	Azi [°]	TVD [m]	NS [m]	EW [m]	V.Sec [m]	DLS [°/30m]	t.Face [°]	Build [°/30m]	Turn [°/30m]
1	0	0	0	0	0	0,03	0	0	0	0	0	0
2	1000	0	0	0	0	0	0	0	0	0	0	0
3	1075	75	5	12,51	1074,9	3,23	0,71	3,27	2	12,5	2	5
4	1475	400	5	12,51	1473,38	37,26	8,26	37,39	0	0	0	0
5	3075,00	1600,00	5,00	12,51	3067,29	173,40	38,47	-	0	-	0	0
C.Pass:	3075	1600	5	12,51	3067,29	173,4	38,45	173,57	177,58	0	0	0

MD

	MD_{start} [m]	CL [m]	INC [°]	Azi [°]	TVD [m]	NS [m]	EW [m]	V.Sec [m]	DLS [°/30m]	t.Face [°]	Build [°/30m]	Turn [°/30m]
1	0	0	0	0	0	0,03	0	0	0	0	0	0
2	1000	0	0	0	0	0	0	0	0	0	0	0
3	1075	75	5	12,51	1074,9	3,23	0,71	3,27	2	12,5	2	5
4	1475	400	5	12,51	1473,38	37,26	8,26	37,39	0	0	0	0
5	1600,00	125,00	5,00	12,51	1597,90	47,90	10,62	-	0	-	0	0
C.Pass:	1600	125	5	12,51	1597,91	47,9	10,62	49,03	0	0	0	0

TVD

	MD_{start} [m]	CL [m]	INC [°]	Azi [°]	TVD [m]	NS [m]	EW [m]	V.Sec [m]	DLS [°/30m]	t.Face [°]	Build [°/30m]	Turn [°/30m]
1	0	0	0	0	0	0,03	0	0	0	0	0	0
2	1000	0	0	0	0	0	0	0	0	0	0	0
3	1075	75	5	12,51	1074,9	3,23	0,71	3,27	2	12,5	2	5
4	1475	400	5	12,51	1473,38	37,26	8,26	37,39	0	0	0	0
5	1602,10	127,10	5,00	12,51	1600,00	48,07	10,66	-	0	0	0	0
C.Pass:	1602,1	127,1	5	12,51	1600	48,08	10,66	49,21	0	0	0	0

Figure 9 - Tangent calculations

10 Further work

1) Targets
2) Anti collision
3) More methods for calculating segments – 3D and 2D (expand Method 1)
4) Add functionality to add multiple sections: "Method 2" in table below
5) Importing well plans from other models – e.g. from ascii or other standard format data
6) "Intelligent calculations": the user ads a segment (line in /) to the plan by selecting a target (as end point). Also, user defines that the well should line up for the next target (i.e. Inc and Azi in the end point are pointing towards the next target).

Method 2
3D segments:

Name:	Users focus parameter	Input parameters	Calculates
Curve tangent curve	DLS 1&2	dTVD, dN & dE, Inc Azi, OR: Target	(on hold) ("all parameters")
curve & curve	DLS 1 or 2	dTVD, dN & dE, Inc Azi OR: Target	(on hold) ("all parameters")

Table 4 - Further 3D calculations

Appendix A Glossary

Term	Also named	Unit	Definition
Azimuth	Azi, Azm, Drift	degrees	Compass heading of trajectory, angle between wellpath and North axis (true north or magnetic North) measured in plan view, clockwise from North. 0º for heading North, 90º for heading East, 180º for heading South, 270º for heading West. Also called "Drift direction".
Bend		degree	Angle between Downhole mud motor and drill bit, used to achieve build or turn while sliding. Usually a mud motor is "dialled" at 1.8º to 2.8º on the build section, and 1.5º to 1.8º on a horizontal section.
Build		º or meters	Increase of inclination; in horizontal drilling also decrease in TVD as a direct result of building angle. As opposed to drop.
Build rate		º/30m	Inclination angle difference between two consecutive survey points, extrapolated to 30m meters (or 100 feet); measured in vertical plane. Typical build rates in a build section of a horizontal well are 6-8º/30m.
Build section	Build		Section of a well where the wellpath is changed to a different inclination, usually from vertical (0º) to horizontal (90º).
Closure	Closure distance	meters	Length of wellpath projected on horizontal plane (plan view) .
Closure direction	Closure azimuth		
Dogleg	Dogleg severity, DLS	º/30m	Combination of build and turn rate; normalized measure of curvature (bend and turn) at a certain point. High doglegs pose difficulty in tripping and running casing or liner.
Drop		º or meters	Decrease of inclination; in horizontal drilling also increase in TVD as a direct result of dropping angle. In an "S" shape well, the section where a well is steered back to vertical after a build and hold section. As opposed to build.
Easting	-W/+E	meters	Horizontal distance between survey point and North-South axis in plan view.

Term	Also named	Unit	Definition
Geosteering			Directionally controlled drilling of a well, usually with the intent of keeping the hole in the pay zone, achieving maximum exposure to the reservoir.
Heel	Landing point		First point in a horizontal well trajectory where the inclination reaches 90º. Usually an intermediate casing is set at this landing point to isolate the pay zone.
High side	HS		Toolface oriented up (or 0º) in horizontal drilling when attempting to increase inclination angle (build wellpath).
Hold section	Hold		Section of a well where the wellpath is maintained along a pre-defined inclination and azimuth. If the inclination ids different from 0º (vertical) or 90º (horizontal), it is also called a tangent section.
Horizontal section	Lateral section		Section of a wellpath where the inclination is maintained at around 90º, usually with the intent to keep the hole in the pay zone in order to maximize exposure to a reservoir. Typically inclinations are maintained in the 87 to 93º range. If steeper inclinations are planned, a more general term used is “Lateral section”.
Inclination	Inc, Incl	degrees	Angle between a tangent to the wellpath and a vertical line, measures the deviation from vertical at a certain point, measured with pendulum, accelerometer or gyroscope. Inclination increases in the build section, is 90º in a perfectly horizontal well, and can exceed 90º in a horizontal well that points up. A vertical well has inclinations close or equal to 0º.
Kick off point	KOP	meters	Measured depth from where a well is steered (usually beginning of the build section).
Landing	Landing point, Heel	205	Landing is the process of increasing the TVD and inclination until a wellpath reaches horizontal or near-horizontal position in a desired reservoir formation. A **soft landing** is a landing where a specific inclination (usually 80 to 85º) is maintained until a formation top is confirmed, and the wellpath is turned to horizontal only after that reservoir top was confirmed. The landing point often

Term	Also named	Unit	Definition
			coincides with the heel (the point where inclination reached 90º, horizontal), and many times a casing string is run at this point.
Low side	LS		Toolface oriented down (or 180º) in horizontal drilling when attempting to decrease inclination angle (drop wellpath).
Measured Depth	MD	meters	Depth of well measured as the bit goes.
Mud motor	Downhole motor		Mud driven 39ownhole motor which rotates the bit below the bend.
MWD	Measured while drilling		Set of physical parameters measured in a wellbore while drilling the well. Typically measurement of inclination, azimuth, gamma, sometimes resistivity (if more parameters are recorded, the operation is called LWD, Logging while drilling).
Northing	+N/-S	meters	Horizontal distance between survey point and East-West axis in plan view.
Rotary steerable			Device used to geosteer while rotating from surface, usually driven by mobile pads.
Rotating			Rotation of drilling bit is driven from surface (rotary table or top-drive), usually higher rate of penetration, but no steering (except for rotary steerable devices).
Sidetrack	Leg		A leg of a well that is started from a point along the wellpath that was previously drilled. The bottomhole assembly is positioned with the bend facing the desired direction, a through is formed by working the pipe, followed by time drilling to achieve proper separation.
Sliding	Slide	206	Drilling without rotating the entire drillstring (by way of the rotary table or top-drive), rotation of bit occurs only below the bend, achieved by the mud driven 39ownhole motor.
Station depth	MD	meters	Survey tool depth, measured depth where survey is recorded, usually 10-15m behind the bit.

Term	Also named	Unit	Definition
Subsea	SS, Subsea level	meters	Vertical distance between see level and survey point.
Survey	Directional survey		A geometric measurement of inclination and azimuth at a certain depth (station depth) along the well trajectory. A series of other parameters (such as TVD, VS, dogleg, etc are then determined using calculations such as *minimum curvature method*.
Tangent section	Tangent, hold section		Section of a well where the wellpath is maintained at a certain inclination, with the intent of advancing in both TVD and vertical section. Short tangent sections are build for housing submersible pumps for example.
Time drill			Procedure where well is oriented in order to start a sidetrack.
Toe			Total depth of a horizontal well. A toe-up profile is achieved when the inclination is more than 90º throughout the horizontal section (the well rises, "builds"), a **toe-down** profile is achieved when the inclination angle is below 90º in the horizontal section (the well points down, "drops", and the toe is located at a lower TVD than the heel).
Toolface		degree	Orientation of motor bend: in horizontal drilling measured relative to vertical line (up is 0º, down is 180º, right is 90º or 90R, left is 270º or 90L); in vertical drilling relative to magnetic north (N is 0M, E is 90M, S is 180M, W is 270M).
Total Depth	TD, Final TD, FTD	meters	Measured depth to which a well is drilled.
True Vertical Depth	TVD	meters	Vertical distance between KB and survey point.
Trajectory	-	- 207	A curve or surface that passes through a given set of points or intersects a given series of curves or surfaces at a constant angle. See also "Wellpath" below.
Turn rate	Walk rate	º/30m	Azimuth angle difference between two consecutive survey points, extrapolated to 30m meters (or 100 feet); measured in

Term	Also named	Unit	Definition
			horizontal plane.
Turn section	Turn		Section of a well where the wellpath is steered in the horizontal plane to a different azimuth.
Vertical Section	VS	meters	Horizontal distance from wellhead to survey point, measured along a pre-defined azimuth in the horizontal plane.
Vertical section plane	Vetrical section azimuth	degrees	Pre-defined azimuth angle along which the VS is calculated, usually angle between North and a line uniting wellhead and Total Depth, measured in plan view.
Wellpath	Trajectory		The trajectory of a directionally drilled well in three dimensions.
Horizontal section	Lateral section		Section of a wellpath where the inclination is maintained at around 90º, usually with the intent to keep the hole in the pay zone in order to maximize exposure to a reservoir. Typically inclinations are maintained in the 87 to 93º range. If steeper inclinations are planned, a more general term used is "Lateral section".

A useful place to search for terms in “oil language:
http://www.glossary.oilfield.slb.com/

Appendix B Minimum Curvature – deduction

Minimum Curvature is based on the method called "Balanced Tengential Method". The difference will follow at the end of this chapter.

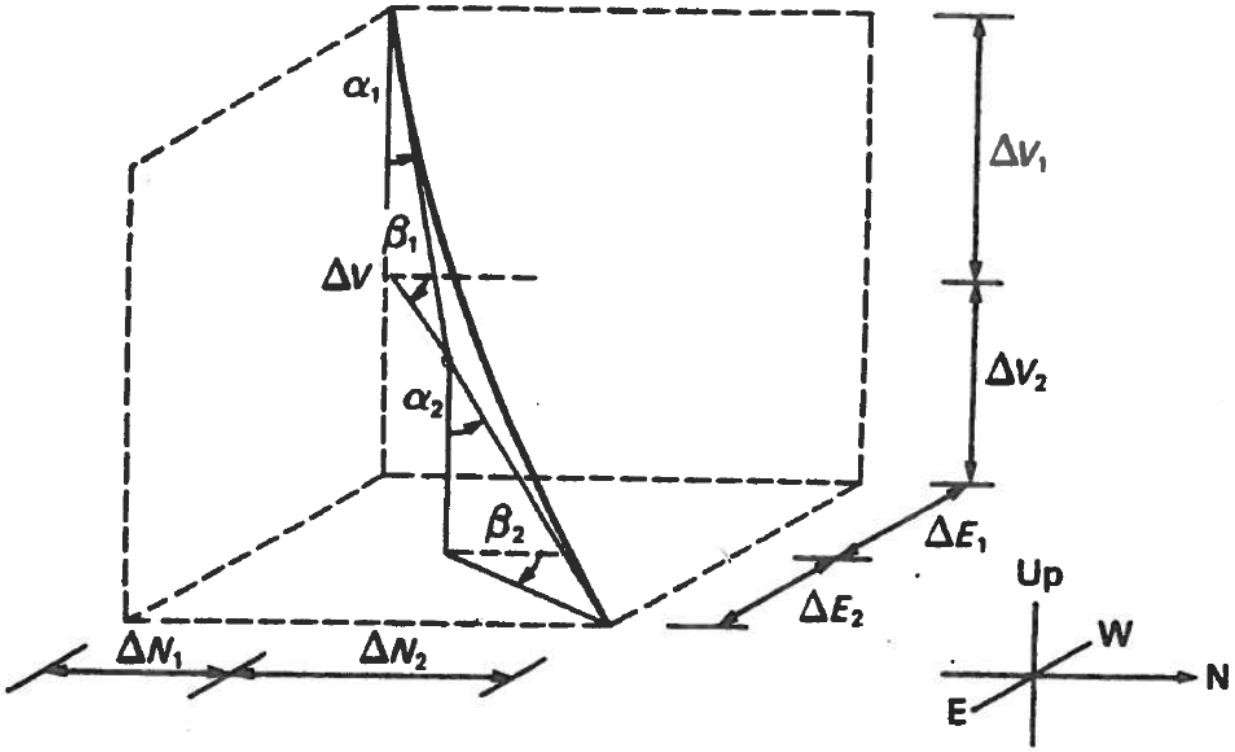


Figure 10 - Balanced Tangent

This method assumes that the actual weilpath can be approximated by two straight line segments of equal length. The upper segment is defined by α_1 and β_1, while the lower segment is defined by α_2 and β_2. The length of each segment = L/2. From the figure above it can be seen that:

$\Delta V = L/2 (\text{Cos } \alpha_1 + \text{Cos } \alpha_2)$

$\Delta N = L/2 (\text{Sin } \alpha_1 \text{Cos } \beta_1 + \text{Sin } \alpha_2 \text{Cos } \beta_2)$

$\Delta E = L/2 (\text{Sin } \alpha_1 \text{Sin } \beta_1 + \text{Sin } \alpha_2 \text{Sin } \beta_2)$

"Balanced Tengential Method" can be further improved by applying a ratio factor – this is the Minimum Curvature Method. Rather than assuming that the actual weilpath is approximated by two straight line segments, this method replaces the straight lines by a circular arc. This is done by

applying a ratio factor based on the amount of bending in the weilpath between the two stations (dog-leg angle). The dog-leg angle Φ can be calculated from:

$\Phi := \text{Cos}^{-1}[\text{Cos}\ \alpha_1 \cos \alpha_2 + \sin \alpha_1 \sin \alpha_2 \cos(\beta_2-\beta_1)]$

The figure below, shows how the ratio factor F is calculated:

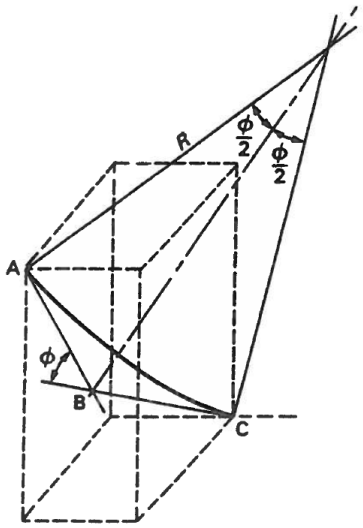


Figure 11 - Fraction coeffisient

$$F = \frac{AB + BC}{arc\ AC}$$

and

$$AB = BC = R \tan(\frac{\emptyset}{2})$$

and

$$\frac{AC}{2\pi R} = \frac{\emptyset}{360} <=> AC = \frac{\pi R \emptyset}{180}$$

Therefore,

$$F = \frac{2}{\emptyset}\left(\frac{180}{\pi}\right) tan\left(\frac{\emptyset}{2}\right)$$

This ratio factor is then applied to the results of iW, AN and E as given by the balanced tangential method. The equations for the minimum curvature method can be summarized as follows:

$\Delta V = F\ L/2\ (\text{Cos}\ \alpha_1 + \text{Cos}\ \alpha_2)$
$\Delta N = F\ L/2\ (\text{Sin}\ \alpha_1 \text{Cos}\ \beta_1 + \text{Sin}\ \alpha_2 \text{Cos}\ \beta_2)$
$\Delta E = F\ L/2\ (\text{Sin}\ \alpha_1 \text{Sin}\ \beta_1 + \text{Sin}\ \alpha_2 \text{Sin}\ \beta_2)$

F: =2/Φ(180/π) tan(Φ/2)

Φ: $= \text{Cos}^{-1}[\text{Cos}\ \alpha_1 \cos \alpha_2 + \sin \alpha_1 \sin \alpha_2 \cos (\beta_2-\beta_1)]$

210

Where:

ΔV: Is displacement from last survey in vertical direction
ΔN: Is displacement (projected in horizontal plane) from last survey, northerm component
ΔE: Is displacement (projected in horizontal plane) from last survey, eastern component
L: Length drilled since last survey
Φ: Dog leg angle
α_1: Inclination at last survey (=a1)
α_2: Inclination at new survey (=a2)
β_1: Azimuth at last survey (ref to north) (=b1)
β_2: Azimuth at new survey (ref to north) (=b2)

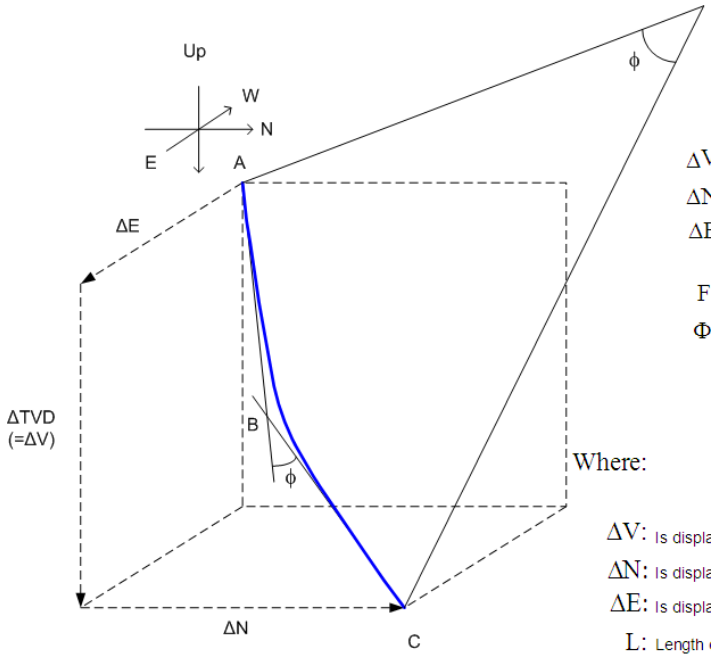
Appendix C Offset model – Minimum Curvature Method

Basically, only 1 of the models used to measure well paths while drilling will be discussed here – the Minimum Curvature. This is known in the oil industry as one of the most accurate methods for the purpose of measuring well paths during drilling.
Other exsisting models:

- Tangential
- Balanced Tangential
- Average Angle
- Radius of Curvature

As stated before: all survey calculations are based on going from a known point to another. In the example / drawing above (and below), point A will have subscript "1" will have known coordinates, and the calculations will identify point C which is indicated with subscript 2.
I is inclination, which is can be found in the figure above indicating the angle from vertical.
A(zi) is the compass direction (from north).
In the subsequent calculations, the inclination (I in the drawing) will have be denoted α and the Azimut will be denoted β.
Point B in the drawing below is just a help point (cross point of the two tangents in A and C).

The general idea for "minimum curvature" is to use "F" to correct from the 2 tangents over to a "circle sector" since that is the actual well path. More about the deduction of the minimum curvature method in appendix B.



Minimum curvature method

$\Delta V = F\ L/2\ (\cos \alpha_1 + \cos \alpha_2)$
$\Delta N = F\ L/2\ (\sin \alpha_1 \cos \beta_1 + \sin \alpha_2 \cos \beta_2)$
$\Delta E = F\ L/2\ (\sin \alpha_1 \sin \beta_1 + \sin \alpha_2 \sin \beta_2)$

$F: = 2/\Phi(180/\pi) \tan(\Phi/2)$
$\Phi: = \cos^{-1}[\cos \alpha_1 \cos \alpha_2 + \sin \alpha_1 \sin \alpha_2 \cos (\beta_2 - \beta_1)]$

Where:

ΔV: Is displacement from last survey in vertical direction
ΔN: Is displacement (projected in horizontal plane) from last survey, northerm component
ΔE: Is displacement (projected in horizontal plane) from last survey, eastern component
L: Length drilled since last survey
Φ: Dog leg angle
α_1: Inclination at last survey (=a1)
α_2: Inclination at new survey (=a2)
β_1: Azimuth at last survey (ref to north) (=b1)
β_2: Azimuth at new survey (ref to north) (=b2)

Figure 12 – Overview of Minimum Curvature

Appendix C.1 Equations for calculating a 3D curvature

"Minimum Curvature" is used when drilling – not for accurate well planning. These equations are:

$\Delta V = L/2\ (\cos \alpha_1 + \cos\ \alpha_2)$
$\Delta N = L/2\ (\sin \alpha_1 \cos \beta_1 + \sin \alpha_2 \cos \beta_2)$
$\Delta E = L/2\ (\sin \alpha_1 \sin \beta_1 + \sin \alpha_2 \sin \beta_2)$

$F:=2/\Phi(180/\pi) \tan(\Phi/2)$	Relation / correction coeffisient
$\Phi = \cos^{-1}[\cos \alpha_1 \cos \alpha_2 + \sin \alpha_1 \sin \alpha_2 \cos (\beta_2 - \beta_1)]$	Dog Leg

One way to explane the equations would be to go 1 method at the time. The methods would follow the desired functionality of the software – what options would a used need? For this document and project, the functionality will be kept to a minimum that will cover the industry practise.

The methods will be followed up by examples in Excel. When changing the required input for the calculation, the remaining parameters are calculated. This can work as verification when programming the calculations.

NOTE1:

Looking at the 3D curve, Minimum Curvature will not give an accurate estimate of the length of the well path directly.

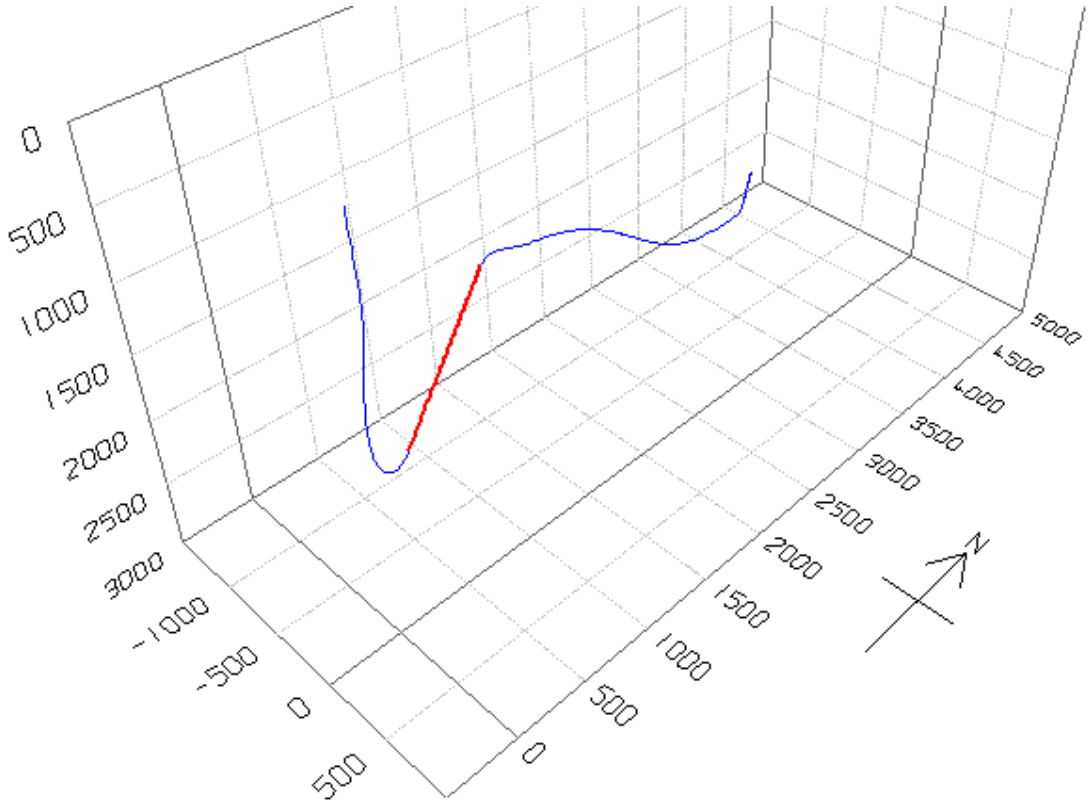
NOTE2:

Due to the fact stated in note 1 above, the equations have not been followed up closely and there is one occasion where the result is negatve when it should be positive. Since this method was added for offset only, it will not be investigated further by the author.

Appendix D Survey listings

This chapter has been added for Group 7 to have some offset data to do calibration of results.

Appendix D.1 Well 1



Plan:

MD (m)	CL (m)	Inc (°)	Azi (°)	TVD (m)	NS (m)	EW (m)	Dogleg (°/30m)	Build (°/30m)	Turn (°/30m)	Section Type
1100,00		35,67	135,49	1057,22	-86,50	128,03	0,000	0,000	0,000	Tie Line
2025,24	925,24	88,09	323,54	1731,67	222,93	-19,61	4,000	1,700	-5,575	DT5 CH Tang
3748,70	1723,46	88,09	323,54	1789,10	1608,31	-1043,20	0,000	0,000	0,000	(ditto)
4031,86	283,16	68,10	356,48	1848,81	1862,49	-1138,83	4,000	-2,118	3,490	OPT AL DLS
4205,86	174,00	68,10	356,48	1913,73	2023,63	-1148,74	0,000	0,000	0,000	(ditto)
4296,92	91,05	80,00	354,00	1938,71	2110,70	-1156,04	4,000	3,922	-0,818	(ditto)
4433,24	136,33	80,71	7,81	1961,66	2244,74	-1153,91	3,000	0,155	3,039	OPT AL DLS
4578,78	145,54	80,71	7,81	1985,17	2387,04	-1134,39	0,000	0,000	0,000	(ditto)
4749,83	171,05	78,00	25,00	2017,00	2547,67	-1087,21	3,000	-0,475	3,015	(ditto)
5019,46	269,62	75,19	52,580	2080,68	2750,12	-924,97	3,000	-0,313	3,069	OPT AL DLS
5181,89	162,43	75,19	52,58	2122,21	2845,53	-800,240	0,000	0,000	0,000	(ditto)
5512,30	330,41	86,00	15,000	2178,10	3111,89	-624,04	3,500	0,982	-3,412	(ditto)
5616,77	104,47	89,86	4,170	2181,88	3214,66	-606,70	3,300	1,110	-3,110	OPT AL DLS
5800,98	184,21	89,86	4,140	2182,31	3398,38	-593,31	0,000	0,000	0,000	(ditto)
6023,73	222,75	80,00	340,00	2202,26	3616,28	-623,24	3,500	-1,329	-3,255	(ditto)
6167,84	144,11	78,16	325,44	2229,70	3741,70	-687,85	3,000	-0,382	-3,031	DT5 CH Tang
6892,93	725,10	78,16	325,44	2378,43	4326,15	-1090,42	0,000	0,000	0,000	(ditto)

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
43,5	0	0	43,5	35,17	7,4	0	0
152	0	356,6	152	35,17	7,4	0	0
291,1	2,77	356,6	291,05	38,53	7,2	3,3	0,597
310	2,28	353,88	309,93	39,36	7,13	4,12	0,8
320	2,23	2,41	319,92	39,75	7,12	4,5	1,017
330	2,2	1,34	329,91	40,13	7,13	4,88	0,153
360	2,35	9,36	359,89	41,32	7,25	5,99	0,352
390	2,46	16,61	389,86	42,54	7,53	7,11	0,323
420	2,61	19,17	419,83	43,8	7,94	8,23	0,188
450	2,24	12,04	449,81	45,02	8,29	9,32	0,477
480	1,74	2,44	479,79	46,05	8,43	10,29	0,599
510	0,8	354,54	509,78	46,71	8,43	10,93	0,954
540	0,5	192,69	539,78	46,79	8,38	11,02	1,285
570	0,74	161,41	569,78	46,48	8,41	10,71	0,406
600	2,1	142,12	599,77	45,87	8,81	10,01	1,423
630	4,26	137,01	629,72	44,62	9,91	8,53	2,176
660	7,1	136,68	659,57	42,45	11,94	5,93	2,84
690	9,18	137,25	689,27	39,35	14,84	2,2	2,082
720	10,94	139,17	718,8	35,43	18,32	-2,45	1,791
750	12,92	140,61	748,15	30,69	22,31	-8,04	2,002
780	15,45	140,64	777,24	25	26,98	-14,7	2,53
790	16,5	140,62	786,85	22,88	28,72	-17,19	3,15
800	17,62	140,5	796,41	20,61	30,59	-19,85	3,362
810	18,54	140,44	805,92	18,22	32,56	-22,66	2,761
820	19,39	140,43	815,37	15,71	34,63	-25,6	2,55
830	20,19	140,4	824,78	13,1	36,79	-28,66	2,4
840	21,04	140,31	834,14	10,39	39,03	-31,85	2,552
850	21,85	140,19	843,45	7,58	41,37	-35,15	2,434
860	22,62	140,14	852,71	4,68	43,8	-38,56	2,311
870	23,35	140,22	861,91	1,68	46,3	-42,09	2,192
880	24,12	140,22	871,07	-1,42	48,87	-45,72	2,31
890	24,89	140,06	880,17	-4,6	51,53	-49,47	2,319
900	25,63	139,95	889,21	-7,87	54,27	-53,31	2,224
910	26,38	139,77	898,2	-11,22	57,1	-57,26	2,262
920	27,12	139,51	907,13	-14,65	60,01	-61,31	2,248
930	27,91	139,07	916	-18,15	63,03	-65,45	2,447
940	28,78	138,65	924,8	-21,73	66,15	-69,68	2,678
950	29,71	138,28	933,52	-25,38	69,39	-74,03	2,842
960	30,63	137,7	942,17	-29,12	72,76	-78,48	2,895

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
970	31,55	137,22	950,73	-32,92	76,25	-83,03	2,858
980	32,53	136,86	959,21	-36,81	79,86	-87,69	2,995
990	33,6	136,67	967,59	-40,78	83,6	-92,47	3,225
1000	34,65	136,44	975,87	-44,85	87,46	-97,37	3,174
1010	35,59	136,28	984,05	-49,02	91,43	-102,39	2,833
1020	36,28	136,01	992,14	-53,25	95,49	-107,49	2,124
1030	36,53	135,73	1000,19	-57,51	99,63	-112,65	0,901
1040	36,16	135,56	1008,25	-61,75	103,77	-117,78	1,15
1050	35,57	135,62	1016,35	-65,93	107,87	-122,85	1,773
1060	35,08	135,61	1024,51	-70,06	111,91	-127,85	1,47
1070	34,89	135,67	1032,7	-74,16	115,92	-132,82	0,579
1080	35	135,58	1040,9	-78,26	119,93	-137,78	0,364
1090	35,3	135,47	1049,07	-82,36	123,96	-142,76	0,92
1100	35,67	135,49	1057,22	-86,5	128,03	-147,78	1,111
1110	34,36	135,09	1065,41	-90,58	132,07	-152,73	4
1140	30,42	133,73	1090,74	-101,83	143,54	-166,47	4
1170	26,51	132	1117,1	-111,57	154,01	-178,49	4
1200	22,62	129,72	1144,38	-119,74	163,42	-188,74	4
1230	18,78	126,57	1172,44	-126,3	171,74	-197,17	4
1260	15,02	121,88	1201,14	-131,23	178,92	-203,73	4
1290	11,41	114,23	1230,35	-134,51	184,93	-208,38	4
1320	8,17	100,2	1259,91	-136,1	189,74	-211,12	4
1350	5,94	72,57	1289,69	-136,01	193,32	-211,92	4
1380	5,97	33,25	1319,54	-134,24	195,65	-210,79	4
1410	8,24	6,04	1349,32	-130,8	196,74	-207,72	4
1440	11,49	352,23	1378,87	-125,7	196,56	-202,73	4
1470	15,11	344,69	1408,06	-118,96	195,12	-195,85	4
1500	18,87	340,05	1436,75	-110,63	192,43	-187,11	4
1530	22,72	336,92	1464,79	-100,73	188,5	-176,55	4
1560	26,6	334,66	1492,05	-89,33	183,36	-164,23	4
1590	30,52	332,94	1518,4	-76,47	177,01	-150,2	4
1620	34,45	331,58	1543,7	-62,22	169,51	-134,53	4
1650	38,4	330,47	1567,83	-46,64	160,87	-117,3	4
1680	42,35	329,54	1590,68	-29,81	151,15	-98,59	4
1710	46,31	328,74	1612,14	-11,83	140,4	-78,49	4
1740	50,28	328,03	1632,09	7,24	128,66	-57,11	4
1770	54,25	327,41	1650,45	27,29	115,99	-34,54	4
1800	58,22	326,84	1667,12	48,23	102,45	-10,9	4
1830	62,19	326,32	1682,02	69,96	88,11	13,7	4
1860	66,17	325,84	1695,09	92,36	73,04	39,14	4
1890	70,15	325,39	1706,24	115,34	57,32	65,29	4

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
1920	74,13	324,95	1715,44	138,77	41,01	92,04	4
1950	78,11	324,54	1722,64	162,55	24,2	119,24	4
1980	82,09	324,14	1727,8	186,55	6,98	146,76	4
2010	86,07	323,74	1730,89	210,67	-10,59	174,48	4
2025,24	88,09	323,54	1731,67	222,93	-19,61	188,59	4
2040	88,09	323,54	1732,16	234,79	-28,37	202,26	0
2070	88,09	323,54	1733,16	258,91	-46,19	230,04	0
2100	88,09	323,54	1734,16	283,02	-64,01	257,82	0
2130	88,09	323,54	1735,16	307,14	-81,83	285,6	0
2160	88,09	323,54	1736,16	331,25	-99,64	313,37	0
2190	88,09	323,54	1737,16	355,37	-117,46	341,15	0
2220	88,09	323,54	1738,16	379,48	-135,28	368,93	0
2250	88,09	323,54	1739,16	403,6	-153,1	396,71	0
2280	88,09	323,54	1740,16	427,71	-170,91	424,49	0
2310	88,09	323,54	1741,16	451,83	-188,73	452,27	0
2340	88,09	323,54	1742,16	475,94	-206,55	480,05	0
2370	88,09	323,54	1743,16	500,06	-224,37	507,83	0
2400	88,09	323,54	1744,16	524,17	-242,18	535,61	0
2430	88,09	323,54	1745,16	548,29	-260	563,38	0
2460	88,09	323,54	1746,16	572,4	-277,82	591,16	0
2490	88,09	323,54	1747,16	596,52	-295,64	618,94	0
2520	88,09	323,54	1748,16	620,63	-313,45	646,72	0
2550	88,09	323,54	1749,16	644,75	-331,27	674,5	0
2580	88,09	323,54	1750,16	668,86	-349,09	702,28	0
2610	88,09	323,54	1751,16	692,98	-366,91	730,06	0
2640	88,09	323,54	1752,15	717,09	-384,72	757,84	0
2670	88,09	323,54	1753,15	741,21	-402,54	785,62	0
2700	88,09	323,54	1754,15	765,32	-420,36	813,39	0
2730	88,09	323,54	1755,15	789,44	-438,18	841,17	0
2760	88,09	323,54	1756,15	813,55	-455,99	868,95	0
2790	88,09	323,54	1757,15	837,67	-473,81	896,73	0
2820	88,09	323,54	1758,15	861,79	-491,63	924,51	0
2850	88,09	323,54	1759,15	885,9	-509,44	952,29	0
2880	88,09	323,54	1760,15	910,02	-527,26	980,07	0
2910	88,09	323,54	1761,15	934,13	-545,08	1007,85	0
2940	88,09	323,54	1762,15	958,25	-562,9	1035,63	0
2970	88,09	323,54	1763,15	982,36	-580,71	1063,4	0
3000	88,09	323,54	1764,15	1006,48	-598,53	1091,18	0
3030	88,09	323,54	1765,15	1030,59	-616,35	1118,96	0
3060	88,09	323,54	1766,15	1054,71	-634,17	1146,74	0
3090	88,09	323,54	1767,15	1078,82	-651,98	1174,52	0

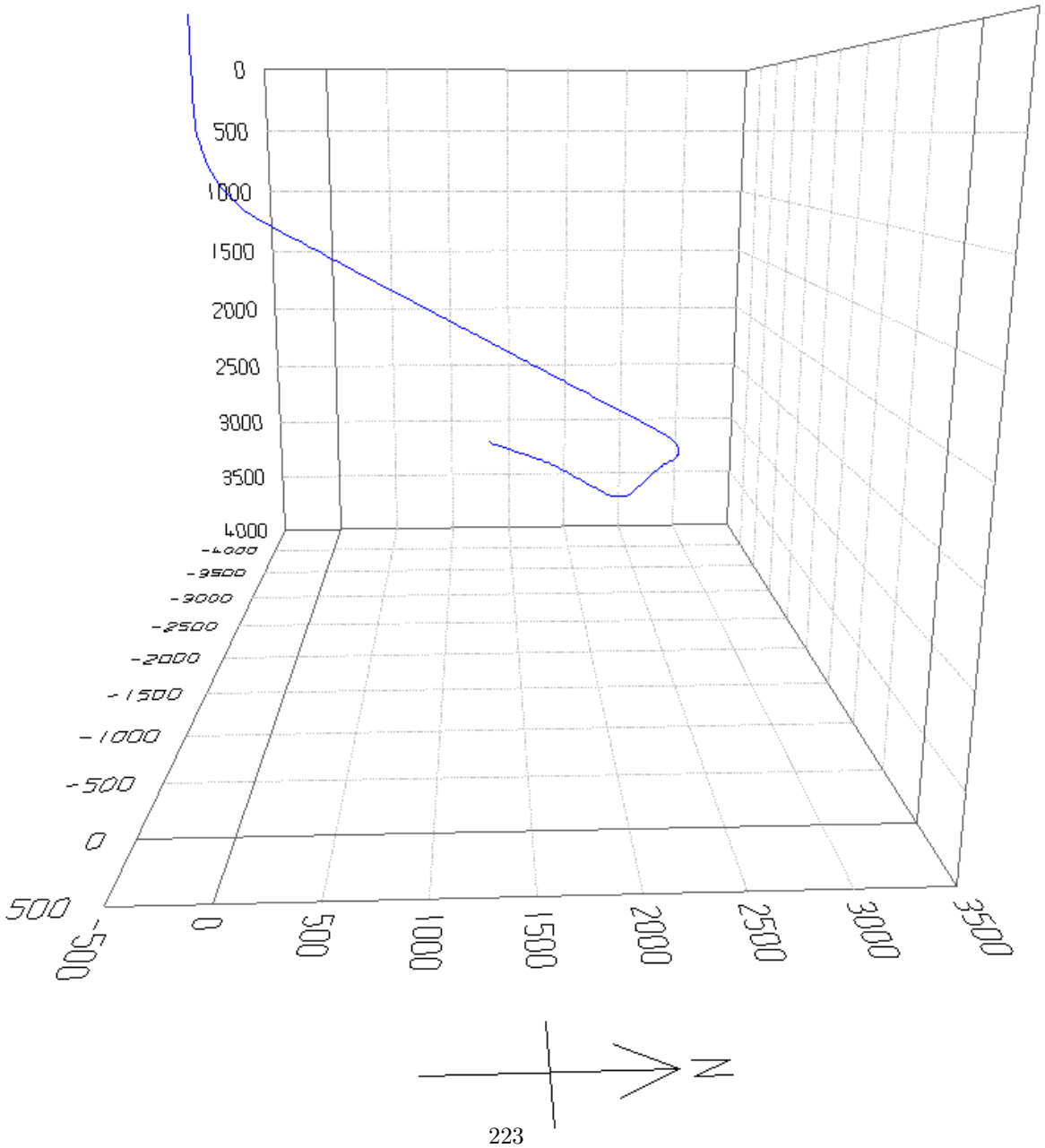
MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
3120	88,09	323,54	1768,15	1102,94	-669,8	1202,3	0
3150	88,09	323,54	1769,15	1127,05	-687,62	1230,08	0
3180	88,09	323,54	1770,15	1151,17	-705,44	1257,86	0
3210	88,09	323,54	1771,15	1175,28	-723,25	1285,64	0
3240	88,09	323,54	1772,15	1199,4	-741,07	1313,41	0
3270	88,09	323,54	1773,15	1223,51	-758,89	1341,19	0
3300	88,09	323,54	1774,15	1247,63	-776,71	1368,97	0
3330	88,09	323,54	1775,15	1271,74	-794,52	1396,75	0
3360	88,09	323,54	1776,15	1295,86	-812,34	1424,53	0
3390	88,09	323,54	1777,15	1319,97	-830,16	1452,31	0
3420	88,09	323,54	1778,15	1344,09	-847,98	1480,09	0
3450	88,09	323,54	1779,15	1368,2	-865,79	1507,87	0
3480	88,09	323,54	1780,15	1392,32	-883,61	1535,65	0
3510	88,09	323,54	1781,15	1416,43	-901,43	1563,42	0
3540	88,09	323,54	1782,15	1440,55	-919,25	1591,2	0
3570	88,09	323,54	1783,15	1464,66	-937,06	1618,98	0
3600	88,09	323,54	1784,14	1488,78	-954,88	1646,76	0
3630	88,09	323,54	1785,14	1512,89	-972,7	1674,54	0
3660	88,09	323,54	1786,14	1537,01	-990,51	1702,32	0
3690	88,09	323,54	1787,14	1561,12	-1008,33	1730,1	0
3720	88,09	323,54	1788,14	1585,24	-1026,15	1757,88	0
3748,7	88,09	323,54	1789,1	1608,31	-1043,2	1784,46	0
3750	87,99	323,68	1789,14	1609,36	-1043,97	1785,66	4
3780	85,73	326,99	1790,79	1633,99	-1061	1813,74	4
3810	83,48	330,31	1793,61	1659,49	-1076,54	1842,3	4
3840	81,26	333,67	1797,59	1685,73	-1090,5	1871,18	4
3870	79,06	337,06	1802,72	1712,59	-1102,82	1900,26	4
3900	76,91	340,5	1808,97	1739,94	-1113,44	1929,39	4
3930	74,79	344,01	1816,3	1767,64	-1122,31	1958,42	4
3960	72,74	347,58	1824,69	1795,55	-1129,38	1987,22	4
3990	70,75	351,24	1834,09	1823,55	-1134,62	2015,64	4
4020	68,83	354,98	1844,46	1851,49	-1138,01	2043,54	4
4031,86	68,1	356,48	1848,81	1862,49	-1138,83	2054,41	4
4050	68,1	356,48	1855,58	1879,29	-1139,86	2070,94	0
4080	68,1	356,48	1866,77	1907,07	-1141,57	2098,27	0
4110	68,1	356,48	1877,96	1934,85	-1143,28	2125,61	0
4140	68,1	356,48	1889,16	1962,64	-1144,99	2152,95	0
4170	68,1	356,48	1900,35	1990,42	-1146,69	2180,29	0
4200	68,1	356,48	1911,54	2018,2	-1148,4	2207,63	0
4205,86	68,1	356,48	1913,73	2023,63	-1148,74	2212,97	0
4230	71,25	355,79	1922,11	2046,21	-1150,26	2235,22	4

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
4260	75,17	354,97	1930,78	2074,83	-1152,57	2263,52	4
4290	79,09	354,18	1937,46	2103,94	-1155,34	2292,41	4
4296,92	80	354	1938,71	2110,7	-1156,04	2299,14	4
4320	80,08	356,34	1942,7	2133,36	-1157,95	2321,56	3
4350	80,21	359,38	1947,84	2162,89	-1159,06	2350,44	3
4380	80,36	2,42	1952,9	2192,45	-1158,59	2378,97	3
4410	80,55	5,46	1957,88	2221,96	-1156,56	2407,06	3
4433,24	80,71	7,81	1961,66	2244,74	-1153,91	2428,46	2,999
4440	80,71	7,81	1962,75	2251,35	-1153	2434,64	0
4470	80,71	7,81	1967,6	2280,68	-1148,98	2462,06	0
4500	80,71	7,81	1972,45	2310,01	-1144,95	2489,48	0
4530	80,71	7,81	1977,29	2339,34	-1140,93	2516,9	0
4560	80,71	7,81	1982,14	2368,67	-1136,91	2544,32	0
4578,78	80,71	7,81	1985,17	2387,04	-1134,39	2561,49	0
4590	80,5	8,93	1987	2397,99	-1132,78	2571,69	3
4620	79,97	11,92	1992,09	2427,06	-1127,43	2598,53	3
4650	79,47	14,93	1997,45	2455,77	-1120,58	2624,65	3
4680	78,99	17,94	2003,05	2484,03	-1112,24	2649,96	3
4710	78,54	20,97	2008,9	2511,77	-1102,44	2674,41	3
4740	78,13	24	2014,97	2538,92	-1091,2	2697,92	3
4749,83	78	25	2017	2547,67	-1087,21	2705,41	3,002
4770	77,69	27,04	2021,25	2565,39	-1078,57	2720,43	3
4800	77,25	30,08	2027,76	2591,11	-1064,57	2741,88	3
4830	76,85	33,13	2034,48	2616	-1049,25	2762,2	3
4860	76,48	36,19	2041,4	2640,01	-1032,65	2781,35	3
4890	76,15	39,26	2048,5	2663,07	-1014,82	2799,26	3
4920	75,86	42,34	2055,76	2685,1	-995,8	2815,9	3
4950	75,61	45,42	2063,15	2706,06	-975,65	2831,2	3
4980	75,4	48,51	2070,66	2725,88	-954,43	2845,14	3
5010	75,23	51,61	2078,26	2744,5	-932,18	2857,68	3
5019,46	75,19	52,58	2080,68	2750,12	-924,97	2861,33	3
5040	75,19	52,58	2085,93	2762,19	-909,19	2869,11	0
5070	75,19	52,58	2093,6	2779,81	-886,16	2880,47	0
5100	75,19	52,58	2101,27	2797,43	-863,12	2891,84	0
5130	75,19	52,58	2108,94	2815,05	-840,09	2903,2	0
5160	75,19	52,58	2116,61	2832,68	-817,05	2914,56	0
5181,89	75,19	52,58	2122,21	2845,53	-800,24	2922,85	0
5190	75,39	51,63	2124,27	2850,35	-794,05	2925,98	3,5
5220	76,19	48,11	2131,64	2869,09	-771,82	2938,63	3,5
5250	77,04	44,62	2138,58	2889,23	-750,7	2952,9	3,5
5280	77,93	41,16	2145,09	2910,68	-730,77	2968,75	3,5

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
5310	78,86	37,71	2151,12	2933,38	-712,11	2986,11	3,5
5340	79,84	34,29	2156,67	2957,23	-694,79	3004,92	3,5
5370	80,85	30,89	2161,7	2982,14	-678,86	3025,11	3,5
5400	81,89	27,51	2166,21	3008,03	-664,39	3046,6	3,5
5430	82,96	24,15	2170,17	3034,79	-651,44	3069,32	3,5
5460	84,05	20,8	2173,56	3062,33	-640,04	3093,17	3,5
5490	85,16	17,47	2176,38	3090,54	-630,25	3118,08	3,5
5512,3	86	15	2178,1	3111,89	-624,04	3137,22	3,5
5520	86,28	14,2	2178,62	3119,32	-622,1	3143,94	3,3
5550	87,39	11,08	2180,28	3148,54	-615,55	3170,63	3,3
5580	88,5	7,98	2181,35	3178,11	-610,58	3198,04	3,3
5610	89,61	4,87	2181,85	3207,91	-607,23	3226,08	3,3
5616,77	89,86	4,17	2181,88	3214,66	-606,7	3232,49	3,3
5640	89,86	4,17	2181,93	3237,83	-605,01	3254,51	0
5670	89,86	4,17	2182,01	3267,75	-602,83	3282,96	0
5700	89,86	4,17	2182,08	3297,67	-600,65	3311,41	0
5730	89,86	4,17	2182,15	3327,59	-598,47	3339,85	0
5760	89,86	4,17	2182,22	3357,51	-596,28	3368,3	0
5790	89,86	4,17	2182,29	3387,43	-594,1	3396,74	0
5800,98	89,86	4,17	2182,31	3398,38	-593,31	3407,16	0
5820	89	2,13	2182,5	3417,37	-592,26	3425,29	3,5
5850	87,63	358,9	2183,39	3447,35	-591,99	3454,27	3,5
5880	86,27	355,67	2184,98	3477,27	-593,41	3483,61	3,5
5910	84,92	352,43	2187,29	3507,02	-596,51	3513,19	3,5
5940	83,59	349,18	2190,29	3536,48	-601,27	3542,92	3,5
5970	82,28	345,91	2193,98	3565,55	-607,69	3572,67	3,5
6000	81	342,62	2198,34	3594,11	-615,74	3602,34	3,5
6023,73	80	340	2202,26	3616,28	-623,24	3625,67	3,5
6030	79,91	339,37	2203,35	3622,07	-625,38	3631,81	3
6060	79,47	336,35	2208,72	3649,4	-636,5	3661,05	3
6090	79,07	333,33	2214,31	3676,08	-649,03	3690	3
6120	78,7	330,29	2220,1	3702,02	-662,93	3718,58	3
6150	78,35	327,25	2226,07	3727,16	-678,17	3746,71	3
6167,84	78,16	325,44	2229,7	3741,7	-687,85	3763,19	3
6180	78,16	325,44	2232,19	3751,5	-694,6	3774,36	0
6210	78,16	325,44	2238,34	3775,68	-711,26	3801,92	0
6240	78,16	325,44	2244,5	3799,86	-727,91	3829,47	0
6270	78,16	325,44	2250,65	3824,04	-744,57	3857,03	0
6300	78,16	325,44	2256,81	3848,22	-761,23	3884,58	0
6330	78,16	325,44	2262,96	3872,41	-777,88	3912,14	0
6360	78,16	325,44	2269,11	3896,59	-794,54	3939,69	0

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
6390	78,16	325,44	2275,27	3920,77	-811,19	3967,25	0
6420	78,16	325,44	2281,42	3944,95	-827,85	3994,8	0
6450	78,16	325,44	2287,57	3969,13	-844,5	4022,36	0
6480	78,16	325,44	2293,73	3993,31	-861,16	4049,91	0
6510	78,16	325,44	2299,88	4017,49	-877,82	4077,47	0
6540	78,16	325,44	2306,04	4041,67	-894,47	4105,02	0
6570	78,16	325,44	2312,19	4065,85	-911,13	4132,58	0
6600	78,16	325,44	2318,34	4090,04	-927,78	4160,13	0
6630	78,16	325,44	2324,5	4114,22	-944,44	4187,69	0
6660	78,16	325,44	2330,65	4138,4	-961,09	4215,24	0
6690	78,16	325,44	2336,8	4162,58	-977,75	4242,8	0
6720	78,16	325,44	2342,96	4186,76	-994,4	4270,35	0
6750	78,16	325,44	2349,11	4210,94	-1011,06	4297,9	0
6780	78,16	325,44	2355,26	4235,12	-1027,72	4325,46	0
6810	78,16	325,44	2361,42	4259,3	-1044,37	4353,01	0
6840	78,16	325,44	2367,57	4283,48	-1061,03	4380,57	0
6870	78,16	325,44	2373,73	4307,66	-1077,68	4408,12	0
6892,93	78,16	325,44	2378,43	4326,15	-1090,42	4429,19	0

Appendix D.2 Well 2



Plan

MD (m)	CL (m)	Inc (°)	Azi (°)	TVD (m)	NS (m)	EW (m)	V,Sec (m)	Dogleg (°/30m)	T,Face (°)	Build (°/30m)	Turn (°/30m)	Section Type
0		0	0	0	0	0	0	0,000	0,000	0,000	0,000	Tie Line
400	400	0	0	400	0	0	0	0,000	0,000	0,000	0,000	Inc Azi MD
1015,8	615,8	61,58	321,28	903,9	234,27	-187,82	246,89	3,000	321,280	3,000	0,000	OPT AL DLS
4253,84	3238,05	61,58	321,28	2445,02	2456,18	-1969,12	2588,44	0,000	0,000	0,000	0,000	(ditto)
4901,45	647,61	61,73	246,32	2790	2576,98	-2462,07	3095,36	3,000	-109,890	0,007	-3,472	(ditto)
4994,29	92,84	52,93	249,83	2840,08	2547,73	-2534,44	3156,36	3,000	162,380	-2,845	1,134	OPT AL DLS
5526,45	532,15	52,93	249,83	3160,85	2401,32	-2933,01	3496,54	0,000	0,000	0,000	0,000	(ditto)
5964,46	438,02	94,9	236,41	3280	2210,98	-3296,69	3790,74	3,000	-19,520	2,875	-0,919	(ditto)
5971,38	01,06,1992	95,07	237,08	3279,4	2207,2	-3302,46	3795,18	3,000	75,870	0,731	2,921	OPT AL DLS
6614,14	642,77	95,07	237,08	3222,57	1859,26	-3839,91	4210,92	0,000	0,000	0,000	0,000	(ditto)
6715,96	101,82	92,53	227,2	3215,8	1796,98	-3920,01	4269,9	3,000	-104,070	-0,748	-2,912	(ditto)
7417,2	701,24	92,53	227,2	3184,8	1320,98	-4434,01	4626,6	0,000	0,000	0,000	0,000	Straight TVD

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
0	0	0	0	0	0	0	0
30	0	0	30	0	0	0	0
60	0	0	60	0	0	0	0
90	0	0	90	0	0	0	0
120	0	0	120	0	0	0	0
150	0	0	150	0	0	0	0
180	0	0	180	0	0	0	0
210	0	0	210	0	0	0	0
240	0	0	240	0	0	0	0
270	0	0	270	0	0	0	0
300	0	0	300	0	0	0	0
330	0	0	330	0	0	0	0
360	0	0	360	0	0	0	0
390	0	0	390	0	0	0	0
400	0	0	400	0	0	0	0
420	2	321,28	420	0,27	-0,22	0,29	3
450	5	321,28	449,94	1,7	-1,36	1,79	3
480	8	321,28	479,74	4,35	-3,49	4,58	3
510	11	321,28	509,33	8,21	-6,58	8,66	3
540	14	321,28	538,61	13,28	-10,65	13,99	3
570	17	321,28	567,52	19,53	-15,66	20,59	3
600	20	321,28	595,96	26,96	-21,61	28,41	3
630	23	321,28	623,87	35,54	-28,49	37,45	3
660	26	321,28	651,17	45,24	-36,27	47,68	3
690	29	321,28	677,78	56,05	-44,93	59,07	3
720	32	321,28	703,62	67,93	-54,46	71,59	3
750	35	321,28	728,64	80,85	-64,81	85,2	3
780	38	321,28	752,75	94,77	-75,97	99,87	3
810	41	321,28	775,89	109,65	-87,91	115,56	3
840	44	321,28	798,01	125,46	-100,58	132,22	3
870	47	321,28	819,03	142,16	-113,97	149,81	3
900	50	321,28	838,91	159,69	-128,02	168,28	3
930	53	321,28	857,58	178	-142,7	187,59	3
960	56	321,28	875	197,06	-157,98	207,67	3
990	59	321,28	891,12	216,79	-173,8	228,47	3
1015,8	61,58	321,28	903,9	234,27	-187,82	246,89	3
1020	61,58	321,28	905,91	237,16	-190,13	249,93	0
1050	61,58	321,28	920,18	257,74	-206,63	271,62	0
1080	61,58	321,28	934,46	278,33	-223,14	293,32	0

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
1110	61,58	321,28	948,74	298,92	-239,64	315,01	0
1140	61,58	321,28	963,02	319,5	-256,14	336,71	0
1170	61,58	321,28	977,3	340,09	-272,65	358,4	0
1200	61,58	321,28	991,57	360,67	-289,15	380,09	0
1230	61,58	321,28	1005,85	381,26	-305,65	401,79	0
1260	61,58	321,28	1020,13	401,84	-322,16	423,48	0
1290	61,58	321,28	1034,41	422,43	-338,66	445,18	0
1320	61,58	321,28	1048,69	443,02	-355,16	466,87	0
1350	61,58	321,28	1062,96	463,6	-371,67	488,56	0
1380	61,58	321,28	1077,24	484,19	-388,17	510,26	0
1410	61,58	321,28	1091,52	504,77	-404,68	531,95	0
1440	61,58	321,28	1105,8	525,36	-421,18	553,65	0
1470	61,58	321,28	1120,08	545,94	-437,68	575,34	0
1500	61,58	321,28	1134,36	566,53	-454,19	597,03	0
1530	61,58	321,28	1148,63	587,11	-470,69	618,73	0
1560	61,58	321,28	1162,91	607,7	-487,19	640,42	0
1590	61,58	321,28	1177,19	628,29	-503,7	662,12	0
1620	61,58	321,28	1191,47	648,87	-520,2	683,81	0
1650	61,58	321,28	1205,75	669,46	-536,7	705,5	0
1680	61,58	321,28	1220,02	690,04	-553,21	727,2	0
1710	61,58	321,28	1234,3	710,63	-569,71	748,89	0
1740	61,58	321,28	1248,58	731,21	-586,21	770,59	0
1770	61,58	321,28	1262,86	751,8	-602,72	792,28	0
1800	61,58	321,28	1277,14	772,39	-619,22	813,98	0
1830	61,58	321,28	1291,42	792,97	-635,72	835,67	0
1860	61,58	321,28	1305,69	813,56	-652,23	857,36	0
1890	61,58	321,28	1319,97	834,14	-668,73	879,06	0
1920	61,58	321,28	1334,25	854,73	-685,23	900,75	0
1950	61,58	321,28	1348,53	875,31	-701,74	922,45	0
1980	61,58	321,28	1362,81	895,9	-718,24	944,14	0
2010	61,58	321,28	1377,08	916,48	-734,75	965,83	0
2040	61,58	321,28	1391,36	937,07	-751,25	987,53	0
2070	61,58	321,28	1405,64	957,66	-767,75	1009,22	0
2100	61,58	321,28	1419,92	978,24	-784,26	1030,92	0
2130	61,58	321,28	1434,2	998,83	-800,76	1052,61	0
2160	61,58	321,28	1448,48	1019,41	-817,26	1074,3	0
2190	61,58	321,28	1462,75	1040	-833,77	1096	0
2220	61,58	321,28	1477,03	1060,58	-850,27	1117,69	0
2250	61,58	321,28	1491,31	1081,17	-866,77	1139,39	0
2280	61,58	321,28	1505,59	1101,76	-883,28	1161,08	0
2310	61,58	321,28	1519,87	1122,34	-899,78	1182,78	0

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
2340	61,58	321,28	1534,14	1142,93	-916,28	1204,47	0
2370	61,58	321,28	1548,42	1163,51	-932,79	1226,16	0
2400	61,58	321,28	1562,7	1184,1	-949,29	1247,86	0
2430	61,58	321,28	1576,98	1204,68	-965,79	1269,55	0
2460	61,58	321,28	1591,26	1225,27	-982,3	1291,25	0
2490	61,58	321,28	1605,53	1245,86	-998,8	1312,94	0
2520	61,58	321,28	1619,81	1266,44	-1015,3	1334,63	0
2550	61,58	321,28	1634,09	1287,03	-1031,81	1356,33	0
2580	61,58	321,28	1648,37	1307,61	-1048,31	1378,02	0
2610	61,58	321,28	1662,65	1328,2	-1064,82	1399,72	0
2640	61,58	321,28	1676,93	1348,78	-1081,32	1421,41	0
2670	61,58	321,28	1691,2	1369,37	-1097,82	1443,1	0
2700	61,58	321,28	1705,48	1389,95	-1114,33	1464,8	0
2730	61,58	321,28	1719,76	1410,54	-1130,83	1486,49	0
2760	61,58	321,28	1734,04	1431,13	-1147,33	1508,19	0
2790	61,58	321,28	1748,32	1451,71	-1163,84	1529,88	0
2820	61,58	321,28	1762,59	1472,3	-1180,34	1551,57	0
2850	61,58	321,28	1776,87	1492,88	-1196,84	1573,27	0
2880	61,58	321,28	1791,15	1513,47	-1213,35	1594,96	0
2910	61,58	321,28	1805,43	1534,05	-1229,85	1616,66	0
2940	61,58	321,28	1819,71	1554,64	-1246,35	1638,35	0
2970	61,58	321,28	1833,99	1575,23	-1262,86	1660,05	0
3000	61,58	321,28	1848,26	1595,81	-1279,36	1681,74	0
3030	61,58	321,28	1862,54	1616,4	-1295,86	1703,43	0
3060	61,58	321,28	1876,82	1636,98	-1312,37	1725,13	0
3090	61,58	321,28	1891,1	1657,57	-1328,87	1746,82	0
3120	61,58	321,28	1905,38	1678,15	-1345,37	1768,52	0
3150	61,58	321,28	1919,65	1698,74	-1361,88	1790,21	0
3180	61,58	321,28	1933,93	1719,33	-1378,38	1811,9	0
3210	61,58	321,28	1948,21	1739,91	-1394,89	1833,6	0
3240	61,58	321,28	1962,49	1760,5	-1411,39	1855,29	0
3270	61,58	321,28	1976,77	1781,08	-1427,89	1876,99	0
3300	61,58	321,28	1991,05	1801,67	-1444,4	1898,68	0
3330	61,58	321,28	2005,32	1822,25	-1460,9	1920,37	0
3360	61,58	321,28	2019,6	1842,84	-1477,4	1942,07	0
3390	61,58	321,28	2033,88	1863,42	-1493,91	1963,76	0
3420	61,58	321,28	2048,16	1884,01	-1510,41	1985,46	0
3450	61,58	321,28	2062,44	1904,6	-1526,91	2007,15	0
3480	61,58	321,28	2076,71	1925,18	-1543,42	2028,85	0
3510	61,58	321,28	2090,99	1945,77	-1559,92	2050,54	0
3540	61,58	321,28	2105,27	1966,35	-1576,42	2072,23	0

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
3570	61,58	321,28	2119,55	1986,94	-1592,93	2093,93	0
3600	61,58	321,28	2133,83	2007,52	-1609,43	2115,62	0
3630	61,58	321,28	2148,1	2028,11	-1625,93	2137,32	0
3660	61,58	321,28	2162,38	2048,7	-1642,44	2159,01	0
3690	61,58	321,28	2176,66	2069,28	-1658,94	2180,7	0
3720	61,58	321,28	2190,94	2089,87	-1675,44	2202,4	0
3750	61,58	321,28	2205,22	2110,45	-1691,95	2224,09	0
3780	61,58	321,28	2219,5	2131,04	-1708,45	2245,79	0
3810	61,58	321,28	2233,77	2151,62	-1724,96	2267,48	0
3840	61,58	321,28	2248,05	2172,21	-1741,46	2289,17	0
3870	61,58	321,28	2262,33	2192,8	-1757,96	2310,87	0
3900	61,58	321,28	2276,61	2213,38	-1774,47	2332,56	0
3930	61,58	321,28	2290,89	2233,97	-1790,97	2354,26	0
3960	61,58	321,28	2305,16	2254,55	-1807,47	2375,95	0
3990	61,58	321,28	2319,44	2275,14	-1823,98	2397,64	0
4020	61,58	321,28	2333,72	2295,72	-1840,48	2419,34	0
4050	61,58	321,28	2348	2316,31	-1856,98	2441,03	0
4080	61,58	321,28	2362,28	2336,89	-1873,49	2462,73	0
4110	61,58	321,28	2376,56	2357,48	-1889,99	2484,42	0
4140	61,58	321,28	2390,83	2378,07	-1906,49	2506,12	0
4170	61,58	321,28	2405,11	2398,65	-1923	2527,81	0
4200	61,58	321,28	2419,39	2419,24	-1939,5	2549,5	0
4230	61,58	321,28	2433,67	2439,82	-1956	2571,2	0
4253,84	61,58	321,28	2445,02	2456,18	-1969,12	2588,44	0
4260	61,37	320,62	2447,96	2460,38	-1972,53	2592,91	3
4290	60,41	317,37	2462,55	2480,16	-1989,72	2615,03	3
4320	59,52	314,06	2477,57	2498,75	-2007,85	2637,71	3
4350	58,72	310,69	2492,97	2516,1	-2026,86	2660,88	3
4380	58,01	307,27	2508,71	2532,17	-2046,71	2684,49	3
4410	57,39	303,79	2524,75	2546,9	-2067,34	2708,47	3
4440	56,87	300,28	2541,03	2560,27	-2088,69	2732,75	3
4470	56,45	296,72	2557,52	2572,22	-2110,71	2757,27	3
4500	56,13	293,13	2574,18	2582,74	-2133,34	2781,96	3
4530	55,92	289,53	2590,95	2591,79	-2156,51	2806,74	3
4560	55,81	285,9	2607,79	2599,34	-2180,15	2831,56	3
4590	55,81	282,28	2624,65	2605,38	-2204,21	2856,35	3
4620	55,91	278,65	2641,49	2609,89	-2228,62	2881,03	3
4650	56,12	275,04	2658,26	2612,85	-2253,32	2905,54	3
4680	56,44	271,46	2674,92	2614,27	-2278,22	2929,81	3
4710	56,86	267,9	2691,42	2614,12	-2303,28	2953,78	3
4740	57,38	264,38	2707,71	2612,43	-2328,41	2977,38	3

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
4770	57,99	260,91	2723,75	2609,18	-2353,54	3000,54	3
4800	58,7	257,48	2739,49	2604,39	-2378,62	3023,21	3
4830	59,5	254,11	2754,9	2598,07	-2403,57	3045,32	3
4860	60,38	250,8	2769,93	2590,24	-2428,32	3066,8	3
4890	61,35	247,55	2784,54	2580,93	-2452,81	3087,61	3
4901,45	61,73	246,32	2790	2576,98	-2462,07	3095,36	3
4920	59,97	246,97	2799,03	2570,56	-2476,94	3107,78	3
4950	57,12	248,07	2814,69	2560,77	-2500,58	3127,64	3
4980	54,28	249,24	2831,59	2551,75	-2523,66	3147,18	3
4994,29	52,93	249,83	2840,08	2547,73	-2534,44	3156,36	3
5010	52,93	249,83	2849,54	2543,41	-2546,2	3166,4	0
5040	52,93	249,83	2867,63	2535,15	-2568,67	3185,58	0
5070	52,93	249,83	2885,71	2526,9	-2591,14	3204,76	0
5100	52,93	249,83	2903,79	2518,65	-2613,61	3223,94	0
5130	52,93	249,83	2921,88	2510,39	-2636,08	3243,11	0
5160	52,93	249,83	2939,96	2502,14	-2658,55	3262,29	0
5190	52,93	249,83	2958,04	2493,88	-2681,02	3281,47	0
5220	52,93	249,83	2976,13	2485,63	-2703,49	3300,65	0
5250	52,93	249,83	2994,21	2477,38	-2725,96	3319,82	0
5280	52,93	249,83	3012,29	2469,12	-2748,43	3339	0
5310	52,93	249,83	3030,38	2460,87	-2770,9	3358,18	0
5340	52,93	249,83	3048,46	2452,62	-2793,37	3377,36	0
5370	52,93	249,83	3066,54	2444,36	-2815,83	3396,53	0
5400	52,93	249,83	3084,63	2436,11	-2838,3	3415,71	0
5430	52,93	249,83	3102,71	2427,86	-2860,77	3434,89	0
5460	52,93	249,83	3120,79	2419,6	-2883,24	3454,07	0
5490	52,93	249,83	3138,88	2411,35	-2905,71	3473,24	0
5520	52,93	249,83	3156,96	2403,1	-2928,18	3492,42	0
5526,45	52,93	249,83	3160,85	2401,32	-2933,01	3496,54	0
5550	55,15	248,87	3174,68	2394,6	-2950,85	3511,72	3
5580	58	247,72	3191,2	2385,34	-2974,11	3531,36	3
5610	60,85	246,64	3206,46	2375,32	-2997,91	3551,32	3
5640	63,71	245,62	3220,41	2364,57	-3022,19	3571,52	3
5670	66,58	244,65	3233,02	2353,13	-3046,89	3591,92	3
5700	69,45	243,72	3244,25	2341,01	-3071,92	3612,45	3
5730	72,33	242,82	3254,07	2328,26	-3097,24	3633,07	3
5760	75,21	241,96	3262,46	2314,91	-3122,76	3653,72	3
5790	78,1	241,11	3269,38	2301	-3148,42	3674,34	3
5820	80,98	240,28	3274,83	2286,56	-3174,14	3694,87	3
5850	83,87	239,47	3278,78	2271,64	-3199,86	3715,25	3
5880	86,76	238,66	3281,23	2256,27	-3225,5	3735,44	3

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
5910	89,66	237,86	3282,17	2240,5	-3251	3755,38	3
5940	92,55	237,06	3281,59	2224,37	-3276,28	3775	3
5964,46	94,9	236,41	3280	2210,98	-3296,69	3790,74	3
5971,38	95,07	237,08	3279,4	2207,2	-3302,46	3795,18	3
6000	95,07	237,08	3276,87	2191,71	-3326,39	3813,7	0
6030	95,07	237,08	3274,22	2175,47	-3351,47	3833,1	0
6060	95,07	237,08	3271,56	2159,23	-3376,56	3852,5	0
6090	95,07	237,08	3268,91	2142,99	-3401,64	3871,91	0
6120	95,07	237,08	3266,26	2126,75	-3426,73	3891,31	0
6150	95,07	237,08	3263,61	2110,51	-3451,81	3910,71	0
6180	95,07	237,08	3260,95	2094,27	-3476,9	3930,12	0
6210	95,07	237,08	3258,3	2078,03	-3501,98	3949,52	0
6240	95,07	237,08	3255,65	2061,79	-3527,07	3968,93	0
6270	95,07	237,08	3253	2045,55	-3552,15	3988,33	0
6300	95,07	237,08	3250,34	2029,31	-3577,24	4007,73	0
6330	95,07	237,08	3247,69	2013,08	-3602,32	4027,14	0
6360	95,07	237,08	3245,04	1996,84	-3627,4	4046,54	0
6390	95,07	237,08	3242,39	1980,6	-3652,49	4065,95	0
6420	95,07	237,08	3239,73	1964,36	-3677,57	4085,35	0
6450	95,07	237,08	3237,08	1948,12	-3702,66	4104,75	0
6480	95,07	237,08	3234,43	1931,88	-3727,74	4124,16	0
6510	95,07	237,08	3231,78	1915,64	-3752,83	4143,56	0
6540	95,07	237,08	3229,12	1899,4	-3777,91	4162,96	0
6570	95,07	237,08	3226,47	1883,16	-3803	4182,37	0
6600	95,07	237,08	3223,82	1866,92	-3828,08	4201,77	0
6614,14	95,07	237,08	3222,57	1859,26	-3839,91	4210,92	0
6630	94,68	235,54	3221,22	1850,5	-3853,05	4221,02	3
6660	93,94	232,62	3218,96	1832,95	-3877,28	4239,22	3
6690	93,19	229,71	3217,1	1814,18	-3900,6	4256,21	3
6715,96	92,53	227,2	3215,8	1796,98	-3920,01	4269,9	3
6720	92,53	227,2	3215,62	1794,24	-3922,96	4271,95	0
6750	92,53	227,2	3214,3	1773,88	-3944,95	4287,21	0
6780	92,53	227,2	3212,97	1753,51	-3966,94	4302,47	0
6810	92,53	227,2	3211,64	1733,15	-3988,93	4317,73	0
6840	92,53	227,2	3210,32	1712,79	-4010,92	4332,99	0
6870	92,53	227,2	3208,99	1692,42	-4032,91	4348,25	0
6900	92,53	227,2	3207,66	1672,06	-4054,9	4363,51	0
6930	92,53	227,2	3206,34	1651,69	-4076,89	4378,77	0
6960	92,53	227,2	3205,01	1631,33	-4098,88	4394,03	0
6990	92,53	227,2	3203,69	1610,97	-4120,87	4409,29	0
7020	92,53	227,2	3202,36	1590,6	-4142,86	4424,55	0

MD (m)	Inc (°)	Azi (°)	TVD (m)	N/S (m)	E/W (m)	V,Sec, (m)	DLeg (°/30m)
7050	92,53	227,2	3201,03	1570,24	-4164,85	4439,81	0
7080	92,53	227,2	3199,71	1549,87	-4186,84	4455,07	0
7110	92,53	227,2	3198,38	1529,51	-4208,83	4470,33	0
7140	92,53	227,2	3197,05	1509,15	-4230,82	4485,59	0
7170	92,53	227,2	3195,73	1488,78	-4252,81	4500,85	0
7200	92,53	227,2	3194,4	1468,42	-4274,8	4516,11	0
7230	92,53	227,2	3193,08	1448,05	-4296,79	4531,37	0
7260	92,53	227,2	3191,75	1427,69	-4318,78	4546,63	0
7290	92,53	227,2	3190,42	1407,33	-4340,77	4561,89	0
7320	92,53	227,2	3189,1	1386,96	-4362,76	4577,15	0
7350	92,53	227,2	3187,77	1366,6	-4384,75	4592,41	0
7380	92,53	227,2	3186,44	1346,23	-4406,74	4607,67	0
7410	92,53	227,2	3185,12	1325,87	-4428,73	4622,93	0
7417,2	92,53	227,2	3184,8	1320,98	-4434,01	4626,6	0

Appendix C

Guides

C.1 How to: Host solution

This section will tell you how to set up a server to host our solution with Django. Ideally, there should be separate hosts for development, testing and production areas.

C.1.1 Server OS

The server we've used Debian Server 7.1, with straight forward (English) installation. Newer Debian versions and Ubuntu server versions should work perfectly fine as well. The requirements for this part are included in default installation: Python, Apache.

C.1.2 User management

When you installed the server, you specified a single user. This user is by default not an administrator (superuser). In order to add yourself as a superuser, run the command `visudo` as a superuser

```
su
visudo
```

This will get you in editmode to specify superusers. A simple way to make yourself user is to include the line

```
your_username   ALL=(ALL:ALL) ALL
```

in the user-part. You might want to make different groups and more users, as well. For this, we reference to [127] and [113]. Please note that these users are the one belonging to the machine, and should not be confused with users in the Wellvis solution.

C.1.3 Packages

The following packages were installed on our development area. While only virtualenv, pip and git are strictly required to run the solution, we recommend all of them.

Package name	Recommendation	Description
openssh-server [77]	Yes	Allows you to remotely ssh to server
screen [55]	Yes	Allows you to keep several open ssh-tabs
vim [117]	Yes	Better editor when editing files directly on server
git [52]	Required	Version control
python-* [16]	Required	Required python tools
pip [84]	Required	Provides you with better package control
virtualenv [119]	Required	Allows you to easily create different environments
netatalk [71]	No	Allows you to use AppleTalk directly to server. Easier development from Mac. Could be used on development server
vsftpd [123]	Required	Secure and small package that allows (S)FTP to server.
denyhosts [18]	Yes	Prevents more than 5 failed loginattempts from same IP.

These packages are installed via the following commands.

```
sudo apt-get install openssh-server
sudo apt-get install screen
sudo apt-get install vim
sudo apt-get install git
sudo apt-get install python-pip python-dev build-essential
sudo pip install pip --upgrade
sudo pip install virtualenv
sudo apt-get install netatalk
sudo apt-get install vsftpd
sudo apt-get install denyhosts
```

C.1.4 Configuration

FTP configuration

We want to make sure that anonymous users are not able to FTP to our server.

```
vim /etc/vsftpd.conf
```

Make sure that `ANONYMOUS_ENABLE` is commented out or set to `NO`
Make sure that `LOCAL_ENABLE=YES`

For the configuration to take effect, we need to restart the ftp module.

```
/etc/init.d/vsftpd restart
```

git configuration

You need to tell git what your Name and e-mail is, before you can start using git. This you do by [53]

```
git config --global user.name ‘‘Your Name’’
git config --global user.email ‘‘your_email@example.com’’
```

C.1.5 Get git-repository

The code repository should be located at main server that acts as a backup and an intersection between different developers code contributions. We have used, and recommend, Github [54] for this.
The repository used in our project is located at project ww with github user tomfa [109]. To get this down to the server, first create a designated folder for the repository

```
mkdir wellvis
cd wellvis
```

Then initate a git repository with

```
git init .
```

...before you pull down the code from the server

```
git remote add origin https://github.com/tomfa/ww.git
git pull origin master
```

You will now have pulled down the master branch from our repository to your local machine. For further details on how to use git, see 10.

C.1.6 Virtual environment

In the repository, there is a folder called env. This folder contains all the required libraries in the correct versions to run the django-application. We use virtualenv to make sure there is a consistency between different servers and users when it comes to packages, versions and setup. You can think of it like the files required for a virtual operating system, except it is a virtual "project". In order to activate the virtual environment instead of the one located at the machine, run:

```
source env/bin/activate
```

This will show `(env)` at the start of every line to indicate that you're inside the virtual environment.

C.1.7 Django configuration

Before being able to run the server with the Django framework, we will also need to modify a settings-file. The settings-file is individual per server where django is deployed, so the our repository does not include it. However, it includes the file `wellvis/wellvis/default_settings.py`, which has the settings used in at our testserver. It should be copied to `wellvis/wellvis/settings.py` and the section DATABASES edited. Django supports Oracle, MySQL, PostgreSQL and SQLite. For more information on how to set up the database, see *Django - How to Install Django* [26].

Once the database settings are set up. You can tell Django to create the necessary tables and fields using the command `python wellvis/manage.py syncdbc`

The settings file naturally include many other interesting settings for the web server. For more information on that, please see *Django - Settings* [24]

C.1.8 Running server

Within the github repository, navigate to the content folder and type

```
python manage.py runserver 0.0.0.0:80
```

The `0.0.0.0` specifies that everyone should be able to access it (from different IPs), while the `:80` specifies that it should be available on that port (80) [22]. Note that the port 80 is by occupied by a default Apache page when you've recently installed Debian. This needs to be disabled first if you wish to use that port.

C.1.9 Resources

The following links can be interesting and teach you more about the section we have just finished.

Learn to use screen	http://www.rackaid.com/resources/linux-screen-tutorial-and-how-to/
Learn to use vim	http://yannesposito.com/Scratch/en/blog/Learn-Vim-Progressively/
Learn to use virtualenv	http://simononsoftware.com/virtualenv-tutorial/
Interactive git game	http://pcottle.github.io/learnGitBranching/

Appendix D

Templates

D.1 Meetings

Scrum Meeting

Date: YYYY-MM-DD HH:MM-HH:MM

Place: (Meeting place)

Present: (Group members present)

What have you done?

Tomas	
Tina	
Tintin	
Pawan	

Have you had any problems?

Tomas	
Tina	
Tintin	
Pawan	

Do you have any questions or things you wonder about?

Tomas	
Tina	
Tintin	
Pawan	

What are you going to do today/till next time?

Tomas	
Tina	
Tintin	
Pawan	

D.2 Status Report

Group 7: Development of Drilling Engineering Software

Summary for week 35
Textual summary of what has happened this week

Work done in this period

- Bullet point of the text above

Report progress

- I.x - Title:
  Short text about what was written in section I.x.

Problems

- Risks that has taken effect

Planning of work for the next period

- Title: Short description of what to do the next week

D.3 Testing Templates

D.3.1 Test template

ID	ID of the test
Description	Description/title of the test
Precondition	Precondition that needs to be fulfilled to perform the test
Feature	What feature of the project does this belong to
Execution	Execution steps
Expected Result	Expected result

Table D.1: Test template

D.3.2 Test result template

ID	ID of the test
Description	Description/title of the test
Tester	Name of tester
Date	Date of the test
Result	Result of the test

Table D.2: Test result template

Appendix E

Status Reports

Group 7: Development of Drilling Engineering Software

Summary for week 36

This week our focus has been on documentation and defining tasks, so we can start implementation. A skeleton for project plan has been set up, and the first part of this has been started (section I1 to I4).

We've decided to use Pivotaltracker as a project management tool and defined the project scope in collaboration with the customer. Three sprints has been set up, each being a three week period, starting the ninth of September.

In addition, we've had a 6 hour group dynamics course, and identified some possible problems down the road and how to avoid them.

Work done in this period

- Customer meeting where we defined the requirements
- Started working on the project plan
- Started working on the first four chapters in the report
- Chose project management tool: Pivotaltracker

Report progress

The following document represents the work done this week, and should be located alongside with this document.

- **1. Introduction**
  We have written an introduction to the project that should make the rest understandable to anyone who reads the report
- **2. Project directive**
  We have decided the scope of the project.
- **3. Planning**
  We have started planning how to do the project
- **4. Preliminary studies**
  Project management tool has been selected. Some libraries has been evaluated and chosen. Version control has been evaluated, though reporting document on this needs finishing.

Problems

- We have not been punctual enough. Future meeting should be made to be 15 minutes past the hour, and we must focus on being there well within that time.

Planning of work for the next period

We will...

- **Learn and teach each other django, git**
  All members needs to learn the usage of django and git for development and version control.
- **Set up development environment**
  Setting up git repository, connecting to Pivotaltracker, setting up development server with individual development areas per user .
- **Implementation**
  Deciding on frameworks, and set up a simple web page with login. If time allows it, we will create some data models for the well drilling project hierarchy.
- **Set up testing environment**
  An individual server has to be set up for deploying versions of test application.
- **Be finished with the document section 1 - 5**
  Update document sections 1 through 5 to reflect the current situation.

Group 7: Development of Drilling Engineering Software

Summary for week 37

This week has focused on the evaluation and learning of different tools.
Git has been chosen as version control system, and a repository is located at github. A debian server for development environment has been set up, and users made for each of the group members. Django has been chosen as web framework (server side), and the first Javascript 3D model has been made!

Work done in this period

- Development environment has been set up (see appendix A, page 47)
- An individual test server has been set up on NTNU site, and will be used for the customer to test system after each sprint.
- Played around and learned with 3D modelling in three.js.

Report progress

- **Appendix B - User guides**
  Sections B.1 - B.4, describing how to connect to and use development server.
- **Appendix A - Technical information**
  Section A.1, describing the properties of the development server.
- **Section 3.3 - Quality ensurance**
- **Section 3.2 - Roles**

Problems

- Illness and responsibilities outside this project has caused low amount of logged hours this week. We expect to pick up these lost hours during the next couple of weeks.

Planning of work for the next period

- **Start implementation.**
- Smooth over document, section 1-5.

Group 7: Development of Drilling Engineering Software

Summary for week 38

This week we have worked on the document, continued experimenting with the 3D graphics. We have almost implemented the database.

Work done in this period

- We have made a database and started implementing it
- We have been able to create a 3D model of axis-planes with grids and their labels/numbers that would be generated dynamically when we provide x,y,z ranges. We also added the functionality to show/hide the grids with a toggle button.

Report progress

- **We have rearranged the sections**
  - Section 5 before is now section 3
- **We have added section 6**
  - Test Plan
- **We have added section 7**
  - Architectural description

Problems

- Low attendance, will be better next week

Planning of work for the next period

- **Document:**
  - Finish section 3
  - Make architectural views
  - Make diagrams like GANTT and use case
  - Remove overlapping text
- **Implementing:**
  - Add the features to add well lines dynamically.
  - Make user-interface.

Group 7: Development of Drilling Engineering Software

Summary for week 39

This week we have worked on finding out which use cases are necessary for the job, and went through the architectural views and how they are set up. We have managed to make a well in the graphical model. Everyone has worked hard on the project to finish the first sprint.

Work done in this period

- We have figured out which use cases are necessary and started making textual description to these.
- We have been able to make a well in the model
- We have made several architectural views

Report progress

- **We have worked on section 3**
  - Added use cases and textual use cases
- **We have added part 2: Sprints**
  - Section 8
  - Section 9
  - Section 10
- **We have added part 3: Conclusion and evaluation**
  - Section 11
  - Section 12
- **We have worked on section 7**
  - Added architectural views

Problems

- Need algorithms for the well-making process
- People needed to apply for jobs because of itDAGENE, which meant that it was hard to get as many hours as needed

Planning of work for the next period

- **Finish sprint 1**
- **Present sprint 1 to customers and get feedback**
- **Finish planning and starting sprint 2**
- **Document:**
  - Finish section 3
  - Make diagrams like GANTT
  - Write documentation for Sprint 1

Group 7: Development of Drilling Engineering Software

Summary for week 40

This week we have presented our first prototype and shown it to our customer and talked about the next sprint. We have given the customer testing environment that they can use to test the program, and defined tasks for the next sprint. Some technical issues have arisen, but these has been solved.

Work done in this period

- Showing of first prototype
- Arranging the test environment and sending out to customer.
- Planning sprint 2

Report progress

- Section 7
    - Done all that can be done for now

Problems

- Disagreements between group members
- The 3d graph can not function when hiding menu. This was solved by creating a separate view for the 3d-module. Might want to keep it this way.
- Testserver database got corrupted. This was because we tried taking a shortcut by importing database from the development environment.
- Some python-dependencies was not properly set up in the virtual environment on the development server. This caused some problems when deploying to test environment.

Planning of work for the next period

- **Implementation**
    - Changing database for sprint 2
    - Integration between front-end and back-end
    - Configurable colors in 3d graphic
- **Documentation**
    - Write about sprint 1
    - Section 3

Group 7: Development of Drilling Engineering Software

Summary for week 41

This week we have made WBS and planned tasks for sprint 2 and started on some of the tasks. We have managed to hide/show of left panel in the 3d module and construct a structure to use the JSON object in the 3D module. We have also made a script that simplifies regular operations and the ones who have been documenting have tried to understand django and such

Work done in this period

- Made WBS and planned tasks
- Hide/show left panel in 3d module
- Made script that simplifies regular operations

Report progress

- Worked on 4.7.3 Pylons, not finished!

Problems

- Disagreements between group members
- Cannot activate virtual environment through a script
- Hard to understand someone elses code since it also will be redone

Planning of work for the next period

- **Implementation**
    - Make idea for frontpage
    - Read from JSON structure
    - Define variables
- **Documentation**
    - Finish 9.7.3

Group 7: Development of Drilling Engineering Software

Summary for week 41

This week has been all about finishing sprint 2 and getting everything ready for the customer meeting and second prototype presentation. We have managed to do most of what we have planned to do, what is left is just some small things that it is okay to focus on in the next sprint.

Work done in this period

- Make well-view
- Read from JSON-structure
- Growl messages
- Complete the Setting/Configuration Panel with Colorpicker
- AJAX sending/receiving

Report progress

- Finished 4.7.3 Pylons
- Working on fixing and almost finishing chapter 4

Problems

- Disagreements between group members
- Low on hours

Planning of work for the next period

- **General**
  - Customer meeting
  - Create tasks
- **Implementation**
  - No-go-zones
- **Documentation**
  - Fixing and finishing a lot of paragraphs

Group 7: Development of Drilling Engineering Software

Summary for week 43

This week we have focused on calculating the minimum curvature and reading a lot about that. We have also started rewriting our report in Latex, and we are taking into consideration the feedback after the writing course and rewriting the report a bit.

Work done in this period

- Started learning Latex
- Programming x, y and z

Report progress

- We are writing everything over to Latex

Problems

- Disagreements between group members
- Low on hours
- Sickness

Planning of work for the next period

We have a customer meeting today, so a lot of what will be done is dependent on this meeting. We need to secure that we have the customers best interest so we will be making new tasks and such according to what we find out from this meeting

- **Documentation**
    - Writing everything over to Latex, and at the same time rewriting a bit.

Summary for week 44

The tangent method for drawing well path is already done. Now working on the 1st method (among three mentioned by client) to make the 3D curve.

Work done in this period

- tangent method for drawing well path

Report progress

- We are writing everything over to Latex

Problems

- Disagreements between group members
- Low on hours
- Sickness

Planning of work for the next period

- **Implementation**
    - Finish methods to make 3d curve
- **Documentation**
    - Writing everything over to Latex, and at the same time rewriting a bit.

Summary for week 45

This week has gone into finishing our last sprint. Our goal is to finish the curvature during these last few days. This week has also been a lot about documenting and delegating tasks and finding out who does what in the documentation, and moving everything from google docs to latex.

Work done in this period

- curvature
- moving and rewriting stuff in latex

Report progress

- We are writing everything over to Latex
- In the report you will see a difference that some chapters have been moved around, and been altered, and not everything has been added yet.

Problems

- Just some questions about who does what

Planning of work for the next period

Now that we only have under two weeks left, this last time will go too writing the document since most of our grade will be based on that. When we have time, or get bored with documenting we will do some tweaking and such with the software. Hopefully most of the document will be finished during this last week so that we only have the presentation to think about the last few days, so that is the goal, but we migt have to work the whole last weekend to make it.

Appendix F

Test Cases

In this section we introduce all our test cases from this project. All of them follow the template we mentioned in 5.5. We also mention which test cases was in which sprint.

F.1 Sprint 1

The test cases from sprint 1 has ID from ID01-ID06.

F.2 Sprint 2

The test cases from sprint 2 has ID from ID07-ID12.

F.3 Sprint 3

The test cases from sprint 3 has ID from ID12-ID14.

ID	ID01
Description	Log in
Precondition	Must have username and password
Feature	Test that it is possible to log in
Execution	1. Write in username and password 2. Press Login
Expected Result	User should now be logged in to the system

Table F.1: Test case ID01

ID	ID02
Description	Add element
Precondition	Must be logged in and have admin-rights
Feature	Test that it is possible to add elements like country, field etc
Execution	1. Go to admin 2. Press "Add" where you want to add element 3. Fill in info
Expected Result	There should now exist a new element

Table F.2: Test case ID02

ID	ID03
Description	Navigate
Precondition	Must be logged in
Feature	Test that it is possible to navigate
Execution	1. Choose country 2. Choose field 3. Choose well
Expected Result	The user should now see the well panel

Table F.3: Test case ID03

ID	ID04
Description	Hide navigation panel
Precondition	Must be logged in
Feature	Test that it is possible to hide the navigation panel
Execution	1. Press toggle-button
Expected Result	The navigation panel should now be hidden and the toggle-button has changed

Table F.4: Test case ID04

ID	ID05
Description	Insert well path points
Precondition	Must be logged in and have chosen to make well path
Feature	Test that it is possible to insert well path points
Execution	1. Insert well path points
Expected Result	The well path starts showing with every new points inserted

Table F.5: Test case ID05

ID	ID06
Description	Remove well path points
Precondition	Must be logged in,have chosen to make well path and have at least one point in the well path
Feature	Test that it is possible to remove well path points
Execution	1. Press remove by the line where you want to remove the points
Expected Result	The well path changes and the point is removed

Table F.6: Test case ID06

ID	ID07
Description	Choose opacity on grid
Precondition	Must be logged in and have a well path
Feature	Configurable 3D view
Execution	1. Choose configuration 2. Choose and change opacity on grid
Expected Result	The grids opacity should be less visual

Table F.7: Test case ID07

ID	ID08
Description	Choose color on x-axis
Precondition	Must be logged in and have a well path
Feature	Configurable 3D view
Execution	1. Choose configuration 2. Choose and change color on x-axis
Expected Result	The x-axis should have a different color now

Table F.8: Test case ID08

ID	ID09
Description	Choose size on labels
Precondition	Must be logged in and have a well path
Feature	Configurable 3D view
Execution	1. Choose configuration 2. Choose and change size on labels
Expected Result	The labels should have a different size now

Table F.9: Test case ID09

ID	ID10
Description	Move rotation point
Precondition	Must be logged in and have a well path
Feature	Configurable 3D view
Execution	1. Left-click 2. Drag until you get the rotation point like you want it
Expected Result	The rotation point should be changed

Table F.10: Test case ID10

ID	ID11
Description	Zoom grid
Precondition	Must be logged in and have a well path
Feature	Configurable 3D view
Execution	1. Scroll to get the right size
Expected Result	The grid's size should have changed

Table F.11: Test case ID11

ID	ID12
Description	Move grid
Precondition	Must be logged in and have a well path
Feature	Configurable 3D view
Execution	1. Right-click 2. Drag until you have the grid where you want it
Expected Result	The grid should be moved

Table F.12: Test case ID12

ID	ID13
Description	Choose fullscreen
Precondition	Must be logged in and have a well path
Feature	Test that fullscreen is available
Execution	1. Visualize well path 2. Choose Fullscreen
Expected Result	The well path should be in fullscreen mode

Table F.13: Test case ID13

ID	ID14
Description	Choose previous well path
Precondition	Must be logged in and have a well path that has been changed
Feature	Test that fullscreen is available
Execution	1. Go to right well 2. Choose “Restore” 3. Choose the version you would like
Expected Result	The well should be changed to an older version

Table F.14: Test case ID14