Quality Management Report

Group Adapa

Michael Thompson
890301-1974

20/12/13

Introduction

The purpose of this document is to keep track of our progress for the requirements set for Adapa stock watcher system. It will give an overview of the progress towards the final requirements, a test case where the functions are tested, timed and other relevant tests to ensure quality, there is also a conclusion where the overall reliability, performance and understandability of the project is documented. Techniques and standards used for this document are taken from the DIT845 Quality Management course with Miroslaw Staron.

The progress of the requirements will follow a IEEE_Std_610.12-1990 template with some adjustments to fit this project. The IEEE (IEEE_Std_610.12-1990) defines product quality as:

1. The degree to which a system, component, or process meets specified requirements

2. the degree to which a system, component, or process meets customer or user needs or expectations.

In this document it has been decided that this will be demonstrated using simple numbers from 1-10 in a table for each of the two definitions, where a 10 means a function or quality fully meets specified requirement and similar for the end user where a 10 fully meets the needs or expectations of the user.

The document will also explain the pre-release defects. The defect definition is however slightly changed for this project. A defect in this document does not necessarily have to be code related, it should be considered more of a general problem. The Post-release defects will not be appointed for at all in this document. The defects are given a severity, they are the following.

- Severity 1: Crashing the whole system, impacts the architecture
- Severity 2: Functionality of the product is affected, fixes do not impact architecture
- Severity 3: Somewhat affects functionality
- Severity 4: Affects performance, not functionality
- Severity 5: Enhancement request; good to have, but not necessary
- Severity 6: Question

Regarding what severity a defect is given, it should be prioritised differently, for example a defect with the severity 1 should be fixed before a severity 5. The severity 6 might imply that the requirement is not able to be implemented due to lack of time or technical issues.

Functional Requirements

1.1 List of Functional Requirements

* User shall be able to analyse market index details by chart and numbers.
* User shall be able to analyse stock details by chart and numbers.
* User shall be able to view stock in three different views.
* User shall be able to search and filter stocks from selected market.
* System shall supply user with RSS news feeds for available markets.
* User is able to follow desired stocks/Stock Markets in the application.
* System displays news about each available stock.

1.2 Functional Requirements progress

1.2.1 User shall be able to analyse market index details by chart and numbers.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	For the first few sprints focus will on the backend, this function will be implemented at a later sprint. It is not required for us to make our own graphs, so we will use existing graphs from the internet. Which ones we are choosing is yet to be decided.	1	1
17/10/10	Focus is still on backend, development will start in two weeks.	1	1
24/10/10	Focus is still on backend, development will start in one week.	1	1
02/12/13	**Web:** The website currently displays detailed information in one graph. This week the other chart and detailed number view will be implemented. **Desktop:** **Mobile:** Data is currently displayed in graphs, some of the graphs will be redone to function more effectively, numbers will be implemented during this sprint. **Defect Severity 4** Does not affect the system, however performance can be improved.	Web, 5 Desktop, 5 Mobile, 5	Web,5 Desktop, 5 Mobile, 5
06/12/13	A graph has been implemented for all platforms for this matter. For the web and desktop client a stock graph from Hihstock chart has been used to get information about the stock market index from DAX, London stock exchange and Nasdaq. These are also covered in the mobile application using a specific graph API for android. The requirement is fully met.	Web, 10 Desktop, 10 Mobile, 10	Web, 10 Desktop, 10 Mobile, 10

1.2.2 User shall be able to analyse stock details by chart and numbers.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	For the first few sprints focus will on the backend, this function will be implemented at a later sprint. It is not required for us to make our own graphs, so we will use existing graphs from the internet. Which ones we are choosing is yet to be decided.	1	1
17/10/13	Focus is still on backend, development will start in two week.	1	1
17/10/13	Focus is still on backend, development will start in one weeks.	1	1
05/11/13	**Web**: On chart has been found for the webpage for the detailed view of each stock. At the moment the graph only shows fake data. **Defect Severity 2** At the moment there is only a visual graph but not functionality. This affects the functionality of the system, however not the architecture. Functionality will be implemented in the upcoming weeks. **Desktop**: Desktop version is under construction, not ready to implement charts and functionality **Defect severity 2** Desktop version is under construction, it is a little bit early to implement the chart in the desktop version. Affects functionality but does not affect architecture. **Mobile**: Mobile version is under construction, not ready to implement charts and functionality **Defect severity 2** Mobile version is under construction, it is a little bit early to implement the chart in the desktop version. Affects functionality but does not affect architecture.	Web, 3 Desktop, 1 Mobile, 1	Web, 3 Desktop, 1 Mobile, 1

02/12/13	**Web:** The website currently displays detailed information in one graph. This week the other chart and detailed number view will be implemented. **Desktop:** Graph and numbers have been implemented in the desktop client. However the graph has been implemented using Highcharts via HTML which is not apparently allowed, they will be developed into the app! **Defect Severity 2** We will have to rethink on how we are going to implement the graph for the desktop client. This will be arranged as soon as possible! **Mobile:** Data is currently displayed in graphs, some of the graphs will be redone to function more effectively, numbers will be implemented during this sprint. **Defect Severity 4** Does not affect the system, however performance can be improved.	Web, 5 Desktop, 3 Mobile, 3	Web, 5 Desktop, 3 Mobile, 3
16/12/13	All clients now display specific stock in a candlestick diagram and in a table of numbers. The web and desktop client uses graphs from HighCharts and the mobile client uses Stockchartview for android. The table of numbers is just a simple textfield where we load specific information from a request from the database. The requirement is fully met! http://stockchartview.org http://www.highcharts.com	10	10

1.2.3 User shall be able to view stock in three different views.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	This is a requirement from our project owner! Our original plan is to have a linear stock graph displaying the stock market index, a candlestick chart for a specific stock and last a table with relevant numbers. This will be implemented at a later sprint.	1	1
16/12/13	The requirement is fully met!	10	10

1.2.4 User shall be able to search and filter stocks from selected market.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
16/12/13	**Web:** User can either specify which letter the stock begins with, for example, if letter A is selected, all stocks in the selected stock market beginning with the letter A will be shown. It is also possible to search for a specific stock. The search function will match the input from the stock list, only displaying those that match. **Desktop:** It looks through the entire list of stocks and compares the user search input with the company names. Only the matching items will be visible. The filter looks for partly matches, this means that if you would search for "ple" it would show APPLE INC in the list. **Mobile:** The text field triggers a method on every key input. That method compares the letters inserted with symbols loaded in the system. The result is displayed in a drop down list below the search bar. If the user deletes the letters, the list disappears; if the user enters a letters that simply does not give any search results, the list disappears. If a list item is selected, the list disappears and the chosen stock is added to “My Stocks”. In that instance, the system also calls for all the stock data (daily and/or historical) to be downloaded for the just added stock. Requirement fully met.	10	10

1.2.5 System displays newsfeed about economics(RSS)

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	The idea is to fetch a newsfeed about the general economic situation around the world. This will be implemented using an RSS feed.	1	1
17/10/10	Nasdaqs newsfeed is beeing used at the moment. The RSS feed works as intended, however we struggle to get it in to our database. This defect is severity 4, this defect affects the system, our system will however work even without this function. It works as intended, we just need to figure out how to get it in to the database. **Defect Severity 4:** Function somewhat effects the final product.	7	3
05/11/13	Due to problems with getting the information of the RSS feed into the database, it has been decided that the RSS feed will be implemented directly in the front end since it does not really make sense to have it in the backend. **Defect Severity 4.** Function still affects the system, the system will work however without this particular function, It is very nice to have and not to hard to implement, at least on the web client.	5	3
02/12/13	**Web:** An rss feed has been implemented using a open source Rss reader, this works ok but is not optimal since the Rss source code is quite hard to work with. It looks and work how is should though. So far there is only source, a few more will be implemented in a near future. http://www.zazar.net/developers/jquery/zrssfeed/ **Defect Severity 4** The fact that the source code is open source and written in a Javascript which is hard to implement it is not optimal with our code, it does however work as intended. **Desktop:** Yes indeed! Economics, technology and business of Nasdaq feed. **Defect Severity 2** Altthough it might work this is not good enough for launch, should be redone within this sprint. **Mobile:** Yes indeed! Economics, technology and business of Nasdaq feed.	Web, 7 Desktop, 2 Mobile, 10	Web, 5 Desktop, 1 Mobile, 10

16/12/13	All three clients display economic related news using RSS. It was decided that the RSS feed should be implemented directly in the client instead of having it in the ETL. Requirement is fully met.	Web, 10 Dekstop, 10 Mobile, 10	Web, 10 Dekstop, 10 Mobile, 10

1.2.6 User shall be able to follow desired stocks in all applications.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	We would like to give the user the opportunity to save their own stocks for an easy access. The idea is to implement this function via the database, to give the end user the possibility to create an account for this purpose. This will be implemented at a later stage.	1	1
17/10/13	Nothing to report	1	1
24/10/13	Development will start in a few weeks.	1	1
05/11/13	Nothing to report.		
16/12/13	**Web:** The web does not have the ability to login for saving the stocks. Saving stocks to my stock however is possible, the problem however is that it is one for all users. This is not a good solution, hopefully we can fix this prior to project hand in or to the presentation. **Defect Severity 3:** The defect somewhat affects the system, however, it is possible to save stocks to mystocks, a login option would be optimal to enable more users using the web client. **Desktop:** User is able to select stocks to follow after login. They have the ability to add and remove stocks from their list! **Mobile:** Yes, through the cross-platform user database, Login and adding stocks to “my stocks”.	Web, 8 Dekstop, 10 Mobile, 10	Web, 7 Dekstop, 10 Mobile, 10

System displays news about each available stock.

This is an optional requirement

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
16/12/13	The desktop client has this function for the Nasdaq and Dax stock market, however not for the London Stock Exchange. Mobile and web client has not implemented this requirement. Requirement partially met.	Web: 1 Dekstop: 8 Mobile: 1	Web: 1 Dekstop: 8 Mobile: 1

Quality Requirements

2.1 List of quality Requirements

* Data collected from sources shall be accessible in applications maximum five minutes after being fetched.
* Applications shall only receive stock data from backend.
* One view shall be candlestick chart.
* Data should be fetched using Erlang.
* User shall be able to access data via three platforms (mobile, desktop, web).
* System should use at least four different data sources.
* Database shall only consist of well formatted data.
* Database shall contain data up to five years old.
* Database shall not use any SQL language.
* ETL shall discard data not up to standard.
* ETL shall be packaged and run as an OTP application.
* ETL shall be automated, meaning that once started, no more interaction is necessary.
* ETL shall set up database properties necessary to run.

2.2 Quality Requirements progress

2.2.1 Data collected from sources shall be accessible in applications maximum five minutes after being fetched.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	At this moment we are focusing on getting the data into the database and finding sources. This particular part should be done as soon as possible.	2	1
17/10/13	Data is now received into the database. As of now we only get data in manually be running the code itself. **Defect severity 2:** Although not an defect, this particular problem affects the whole thought of the system being automatic.	4	1
24/10/13	The function "timer.erl" has been implemented. This module checks if the requested stock market is open, it takes time differences in account. The system now checks for already available data in the database, if newest feed is already in the database, nothing will happen, if newest feed is not in the database, the database will update. Data will be available directly after inserter into the database. This function is therefor almost done, at least from the back-end point of view. Now we must develop the front-end so the end user will actually get the stored data, **Defect severity 2:** The problem still set to an severity 2, due to the reason that end user cannot access data at present moment. Front end development will begin shortly.	6	3
05/11/13	The back end has been slightly redesigned to make it more effective. The backend is now almost done. Only things remaining is some problems with the parses not getting all the stock symbols, this should be a fairly simple fix. Data is stored in database as soon as it is fetched from our sources, data gets into database really quick, I will time this function at later stage to see if it full fills our requirement. **Defect severity 4.** Not getting all stock symbols affect some of the functionality, the back end works as intended though, which is good.	8	4

16/12/13	As soon as we have uploaded the data into the database it is accessible for the clients. Depending on what stock market is being loaded it takes some time to load the data into the clients. The web and desktop client loads each stock market in an average of 15 seconds. The mobile client does not load the entire stock market list due to the reason to save harddrive space. Each stock is however available via a search! The requirement is fully met.	10	10

2.2.3 Applications shall only receive stock data from backend.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
16//12/13	Our clients only “communicates” with our database, where we specified what data should be included. An exception to this requirement is the stock market index, which is accessed directly from the google to the clients. It does not go through the ETL at all. Requirement fully met.	9	9

2.2.5 ETL shall be implemented in Erlang.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
27/10/13	Currently the symbol getters are not checking if their respective database exists, they do both run even if they've not uploaded any data. I suggest writing something that checks if the database exist, if not, create it. **Defect severity 4.** The defect does affect the system, the solution for the problem however is easy to fix. **Resolved.**		
27/10/13	The bulk API is not currently working like our intentions. Another way of writing the to the DB in a controlled way should be thought of. **Defect severity 2.** Defect can affect entire system but should not affect the architecture. Solution: Batcher has been changed to now spawn a process per message received. Issue solved. **Resolved.**		
	Nasdaq seems to be looping itself, needs to be resolved! Error caused by loop setup. Loop was triggered to soon and many fetching sequences was started the same minute. Problem MIGHT occur again when testing with hardcoded values of M but should not occur when software is released. M is the minute return from erlang:localtime(). Error and solution is applicable on all source server (nasdaq_server, lse_server, and dax_server). **Defect Severity 2** **Resolved.**		

	Logger writes batches of 100 messages or all messages sent if it is idle for five seconds. When spamming logger_server with thousands of messages the file writer crashes because one operation writing is already started. **Defect Severity 2** The defect affects the whole system, if this function is not working our whole backend will crash. Not acceptable, solution is needed right away. **Solution**: loop(List) is now always called with write_log(TempDatabase) as argument and returns an empty list when all messages are written. Messages sent to the logger in the meantime is just queued up and thus, is not lost!Resolved.		
16/12/13	Requirement fully met.	10	Not important for end user.

2.2.6 User shall be able to access data via three platforms (Mobile, Desktop, Web).

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	The idea is to design all of our clients so that they resemble each other. A mock up has been done, which we are all happy with! The clients will not be developed at present moment.	2	2
17/10/13	Development starts in three weeks		
16/12/13	We have a running web, mobile and desktop client. There are some minor adjustments that needs to be done on the clients. These are however more of design issues, they do not affect the functionality of the system in any way. **Defect severity 6:** Mostly design improvements, uncertain if this will be implemented before deadline or for the presentation. Requirement fully met.	10	10

2.2.7 System should use at least four different data sources

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	A constraint from our project owner. As of now we have found one source, marketondemand giving us information for the Nasdaq stock market.	2	2
17/10/13	Two more sources have been discovered, Google apps finance and Yahoo finance. They have been implemented in our back-end and works fine.	4	4
24/10/13	Finding a fourth source is a problem right now. We are currently investigating the last source and making adjustments to the ones we already have.	4	4
16/12/13	We currently have 5 sources. Three of them are for the stock markets. Yahoo for Dax. markitondemand.com for Nasdaq. Gogle finance for LSE and historical data Then we have two sources for the symbols. nasdaq.com for the Nasdaq Symbols myinvestorhub.com for the LSE symbols Requirement fully met.	10	10

2.2.9 Database shall only consist of well formatted data.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	The data we get from our sources must be formatted in the same way before being placed in the database. The solution for this will be to make a parser that checks everything and puts in the same format before inserting to database.	1	1
17/10/13	Our parser module can handle one source good, the other two sources are almost complete and almost works. One big problem is the third source, since we have not yet found a third source we do not know how to parse it to the database. Severely affects the final product! **Defect severity 2:** Although it partially works, this module is most important for the system to function. Without this working the whole project will stop, high focus on fixing this.	4	5
24/10/10	The two not working sources are not functional, problem with the fourth source is still of major concern. **Defect Severity 2:** The fourth source is of major concern. If it is formatted in the same way we might have to fix more with our back-end, this will can be very time consuming.		
16/12/13	In all of our stocker market server modules we have functions that check that server is running and that the symbols contain the data we have specified. This way we can ensure that data inserted into the database will always be correct, Incorrect stocks will be stored in a server logger. Requirement fully met.	10	10

2.2.10 Database shall contain data up to five years old.

Previously: Database contains data from the last 5 years.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	The idea is to have at least 5 years of data to get an good overview about the stock market and it's stock. Should be fairly easy to implement.	1	1
05/11/13	Since this is a school project we do not really need to show that we can store data 5 years back, as long as we get the functionality as intended. It would be possible for us to implement this quality requirement, however it would increase the size of the database very much. This in turn would that mean more data needs to be fetched, leading to longer loading times. It has been decided that we will only use the data we have collected from today and forward. The functionality of the program will be shown neither less. **Defect severity 6.** A nice quality management to implement prior an actual launch, this being a school project it has been decided this is not a vital part of the system. Quality requirement has been changed to **Database shall contain data up to five years old.**	6	6
16/12/13	The ETTL has been up and running on the server since the beginning of november. For all data prior to this we use stock market history from Google. To save size of the clients the historical data will only pulled on request. The requirement is met.	8	8

2.2.11 Database shall not use any SQL language.

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
10/10/13	We have decided not to use any SQL in this program. The reason for this is that we do not think SQL is quick enough for our needs. Our plan is to use couch DB instead.	1	1
17/10/13	The couch DB has been running for a while and it is now clear that we not have to use any SQL for this project. Requirement met!	10	Does not concern end user.

2.2.12 ETL shall discard data not up to standard.

Added 16/12/13

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
16/12/13	This requirement is met via the functions in each server module where functions catch symbols with bad formatted data. Requirement fully met.	10	Does not concern end user.

2.2.13 ETL shall be packaged and run as an OTP application.

Added 16/12/13

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
16/12/13	Since the beginning of december the ETL has been running on as OTP. We did have some problems with the server since we did not have experience of them before. After some reading up everything now works. Requirement fully met.	10	Does not concern end user.

2.2.14 ETL shall be automated, meaning that once started, no more interaction is necessary.

Added 16/12/13

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
16/12/13	The application is running on our server as an OTP, since started we did have to do anything with it! Requirement fully met.	10	Does not concern end user.

2.2.15 ETL shall set up database properties necessary to run.

Added 16/12/13

Date	Progress	To what extent is the requirement fulfilled 1-10	To what extent does the requirement end user expectation 1-10
16/12/13	When the ETL is started it immediately verifies that CouchDB have got all stock databases, symbol databases, and views set up. If any of those are missing, the ETL will create necessary databases and run a symbol checkup to verify that all symbols are up to date. If initial checkup is successfully complete, the main software is launched.” Requirement fully met.	10	Does not concern end user.

Test cases

3.1 Manual test case

17/10/13

Backend has been tested with one source. The stock collector works like it is supposed to, we managed to gather roughly 3000 stocks from the Nasdaq market. There is no dead code in the source code, code is also well commented. It takes a few minutes for the data to actually be transferred to the database.

Problems so far:

Although we manage to collect data, the process itself seems collect our data very slowly. It takes several minutes before all data has been collected. Suggest that we have a look on what can be done on the backend for it to work more efficiently. This affects the whole system in general and is given a high priority to fix.

Only one source is being used at present moment. 2 more sources have been found, and are being implemented. It is unclear if we need to change our parsers or symbol getters. Ideal the symbol getter will look the same for all sources, this will ensure the quality of the backend. Also high priority to fix since it is an requirement for us to have at least for sources. Focus on backend the up coming weeks.

No front-end has been developed yet so it is hard to test the whole program. The front end will be developed as soon as the backend is working properly. Does not have a high priority right now.

25/11/13

ETTL does now collect data from both Nasdaq, DAX and London Stock Exchange. At current moment it takes 6 minutes to collect all the data. The ETL is very stable, it is programmed to keep on running even if parts of it crashes. We now also have a basic functioning design for our web, mobile and desktop client. The web client displays at present moment fake data in one graph, the mobile also displays fake data in a graph and has an RSS-feed running. The shell for the desktop application has been done, no functionality so far though.

Problems so far:

Our main concern at this time is getting our ETL running in our server. We are currently running the server manually, which is an option. It needs to be automated and running on the server as soon as possible. We believe our biggest problem here is our inexperience working with servers. Some minor changes to the ETL are likely to happen as well to fix this problem.

We know that we have to change some of the data format such as the date to work well with our front end solutions. The solution for this problem is easy, the problem though is that we are not exactly sure on what format we needs since it may differ between the different clients.

Some tests indicate that the ETL actually might be a little bit to fast. Not for the actual gathering or storing of the data since our ETL and couch DB can handle the amount of data. What concerns however is the fact that we are not sure if our sources will grant of access. We will slow down the process a little bit. This could be a major issue however, since we cannot control this problem. If our sources deny us access we will have to come up with a backup plan, for now we hope it will work.

Since we are having some troubles everyday with our ETL, it is hard to find time for development of the front end application. Especially for the desktop client. Hopefully the backend is 100% complete this week!

16/12/13

The development process is now almost a 100 % done except from some minor adjustment in the clients and ETL. The ETL now runs as an OTP since a few weeks ago. We had a few errors which were handled internally, for some reason the "dax_server" fails to parse a certain string, we are unsure of why though ,other than that we did not have any crashes on the OTP. We made a good job developing the ETL, using the strengths of Erlang such as concurrency and error handling. We only do requests to the servers when the stock markets are open, saves us on unnecessary requests from the server, preventing us from being blocked. The ETL is very quick, it actually was to quick at some moments that we had to implement some functions to slow it down. Our 5 sources (three for the stock markets, and two for the symbols) all work as they should. We do however request the stock market index directly from google form the clients.

All clients are done except some minor design improvements. I have tried all clients, first impressions is that they are very responsive. I have not manually managed to crash any of the applications yet. For the web and desktop client, loading the stock market takes usually less than 10 seconds. As more data is saved in the database, the loading process might take a little bit longer on occasions. Worst has been around 25 seconds. This usually just happens one or two times before it goes down to 10 seconds again, we believe this problem occurs on the server side. Our server might be a bit unstable, not much we can do about this issue at the moment. The mobile client does not load the entire stock market, thus not having the problem of the loading time. The RSS feed seems to work flawlessly on all platforms. The desktop client and mobile client has got the ability to save my stocks via login, they web does however not have support for this function. At the moment it is possible to save the stocks to mystocks, there is no login however, which means that there is just one list for all users. Hopefully this can be fixed prior to project handing or worst case to the showcase.

3.2 Testers in the code

The ETL has been designed to only store well formatted data into the database *(as discussed in Design Document, section 6.3 JSON structure)*. Our test functions check if the stock symbol exists and also if it contains the correct data. If symbol exists and consists only of correct data it will be saved into the database. If not, current data will be discarded, and the error will be logged. The automated tests does also log errors such as timeouts or if connection is closed remotely, due to spamming. All data fetchers is covered by the same quality tests, and all events is logged through the module *logger_server*.

Timeout might occur if current source is preventing us from fetching, because of spamming, and maximum retries (currently 3) is reached. In case of socket being closed remotely, the fetcher is looped in maximum two minutes.

Conclusion

In this section i will discuss the overall quality of our product.

Reliability:

Our ETL has yet not crashed a single time since we launched it as an OTP application. We have implemented a lot of gourds to catch internal errors to make sure the application is running by itself without crashing.We have not either experienced any crashes on our front end clients. They are merely a source to display the data from the database, thus crashing is of low risk. What could crash the clients are external forces such as backwards compatibility for example android, using a outdated web server or running the desktop client with the wrong OS. At present moment these are forces we can not regard as our fault.

Performance:

The system is really quick. The ETL only requests data when the stock markets are open. Collecting the data is efficient and fast. Loading the data into the clients takes less then 10 seconds. Depending on the amount of data that needs to be loaded it may take a little bit longer though, this only affects the web and desktop since the mobile application does not load the entire stock markets, only my stocks. Loading time can also be an effect of the server, for some unknown reason the loading time increases a lot, we do believe this is a result of the server.

Ease of use:

We expect our end users to have some experience with their client of their choice prior to using the Adapa Stock System. For example we expect users of our Android app to have some experience with android prior to our product. We also assume they have knowledge about stocks. Our clients do all have a simple design with the stock market in focus. The layout is made to make it clear on how to use the applications. Most users should have no difficulty using the applications.

Understandability:

The project is well documented with well updated architecture and good comments to the source code. This will help further development of the product.

Overall we have developed a stable system with a well function ETL running OTP and front ends that are easy to understand and interact with. Crashes are felt with internally without interrupting the runtime. Once started, the backend will keep on working on the server until someone shuts it off. The amount of defects and bugs are more likely to be found in the front end clients rather than the ETL. A lot of time has been spent on the ETL to make it as good as possible.

Comments on the Quality Manager role:

This document has been evolved over time, however some parts are written after development stopped, this meaning that all of the dates in the report does not necessarily project truth, they are however close to actual dates.

Being quality manager for this course has been quite challenging due to the fact that development has been quick, this leading to looking for defects in the code has been difficult. To find most of the defects and bugs would take to much time, therefor I have focused on the bigger picture.