Martin Chukaleski 19951025-5335 guschuma@student.gu.se

Products, Project and People Management
Final Exam

Part 1

T1.

Our task as a group for the Software Architecture for Distributed Systems was to build a product that could on input take a JSON file that contains data about software architecture diagrams (class diagram, sequence diagram) process them and then simulates them. These functional requirements were required by the stakeholders (teachers) but on top of those we also added other functionalities which improved the user experience. To make things clearer, the "sequence diagram simulator" is a web application produced to visualize the messages between lifelines in the diagram in relation to the given file and to share this simulation with other users. If our simulator is a success and we choose to commercialize it then we would have to make a lot of modifications to the software and hardware also, so that every user is satisfied and dealt with properly. The way the system is currently built mainly due to technical and business constraints it wouldn't be able to handle a large group of clients (for instance 1000), there are a lot of security concerns, the speed of execution declines when more users are online and we can't be sure about its yearly unexpected downtime percentage. These are the main concerns about the product and we can associate them with these quality attributes scalability, security, performance and availability. Applying specific design tactics for each quality attribute will enhance the system, but this also comes with a trade off, some side effects will occur using these tactics so we have make sure to level them out. To improve the scalability we need to add more hardware resources (servers) so they can handle a growing amount of work, this will affect the performance in a positive matter since the load is divided among servers meaning that they can execute the calculations faster than one would do. For the security we would have to use an RSA encryption method for the messages between the clients and servers and also apply mechanisms that detect attacks and recover from them so that data integrity is preserved.The availability will be upgraded with Ping/Echo and Watchdog tactics so we can know when servers are idle and not responding and switch the traffic to others, also the enhanced security will have a positive effect since they are closely related.

T2.

The software that we are providing for a wide customer base should be called a product rather than a service, since it is tangible and we as a group are developing it. The business goal would be to sell it to interested clients, the product is particularly a off-the-shelf software with features required by a large amount of people involved in software architecture and it is not customized for each client [9].

T3.

Our project type is a Software Development project since we are a group of people that are working together and following activities that help us to produce a concept into a software product and those are: Requirements Gathering and Analysis, Design, Development, Testing, and Production Implementation. A software development project is a complex undertaking by two or more people within the boundaries of time, budget, and staff resources that produces new or enhanced computer code that

adds significant business value to a new or existing business process [2]. In my opinion as a group we have fulfilled every single one of the above stated criteria.

T4.

In our project the goal was to get a system developed and also its documentation in a period of almost 3 months, fairly quickly i should add but we are in a educational organization and have to be graded by our teachers in a semester's time frame. From this we can conclude that the time was the most dominant factor in our project and the second factor is the financial one, we were not funded so we had to use software that was open source and our own hardware. Since we were not building a safety critical system like the ones in nuclear power plants or airplanes that means, not that the quality was left behind but it was the least influential factor in the project. The task needed to be done cheap and on time, so in the project management triangle of time/cost/quality i think in figure 1.1 we stand on the left middle purple triangle - Cheap & Quick [4].

Figure 1.1 [3]

T5.

The software development life cycle model that is chosen for a software project has many effects on the project management, for example different types of processes and approaches (traditional vs agile) influence the planning and scheduling for the specific phases and also development practices have impact on the cost estimation and the division of work [9]. Project management life-cycle has its own stages just like the SDLC and they have to be aligned with one another into this particular way: initiation - requirements, planning - design, execution - construction and implementation and closeout when the project is finished. This leads to the explanation of why a different SDLC will influence the planning and scheduling. For the commercial version of our software i would most definitely choose the Iterative and incremental methodology, this model is divided into small parts which helps to make modules easily and it is very simple to use. Every iterative step has these phases: requirement, design, repetitive modular implementation, test and integration and final integration & system test. When the iteration is finished and more functions are added the same iterative cycle is repeated. I chose this model because in my opinion the advantages outweigh the disadvantages and in this day and age it is widely used by companies that have a product that is similar to ours. The pros of this models are: the risks of a delay are reduced because all of the important work is done in the beginning, defects are spotted and resolved early, changes to the project are less costly and are easy to implement, functional prototypes are developed early in the cycle which means that we can get feedback from clients, large scale risks can be identified and dealt with high

priority and many more. To conclude from all of these positive effects the commercial version of our product will be developed in less time, low cost and most importantly with minimal risk, however the downside to using this model would mean that the maintenance will be costly [10].

T6.

Management summary

Problem/Opportunity:

One of the main goals of the product that we will build is to help the teachers and also the students with the software architecture studies, specifically speaking in the part of showing and simulating class, behavioral and deployment diagrams. The idea is that our online simulator website can be used by the teacher to create a group and students can join in, after that phase the teacher can choose from two modes: slave or normal mode. If he chooses the firsts he can upload a diagram and simulate it and every student will be able to see his simulation on the website in real-time and after they can discuss the design, but if he chooses the second mode students will also be able to upload files which will be shared with everyone in the group so each of them can simulate the uploaded files. The ambition that we have is to spread the platform in every educational system or organization where software architecture is used, so that we can ease students studies, aid the teaching process and satisfy the needs of software development organizations.

Alternative Options:

One of the options that was taken into consideration but was later dropped, was after a file has been uploaded and the diagram has been rendered, to have a choice whether to simulate it or make changes to the diagram itself. By making changes i mean adding a new actor or life line, changing the names of some components and their specifications, but this idea was left out because there are already tools that are able to perform this online, also the time and cost for implementing this complex feature would be too high and finally the risks that would have come with it are tremendous. However having this option would result in a more functional product which eventually could lead to more clients trying out and using our software, so if the system is a success and is a profitable one, the plan is to modify the system and add this feature.

Reason for Selecting Current Option:

We have selected this current option for our project because this idea is authentic and currently there is no software that is already built that simulates diagrams from a JSON file, offers a single and group view where clients can share their files, so we can consider this as an advantage and a opportunity to fill in the market gap. Gap is a thin opening where entrepreneurs generally witness a market need that has not been met. The most meaningful feature of our product is that everything is online, the simulations are run on our website, this means that we are giving users a reason to skip the stages of searching for a software, installing it, paying a license for it and many other inconveniences that may occur during that process. Having the system on the cloud, in my opinion will make our software more accessible and usable over the competitors, furthermore that is how we will take our chunk out of the market share.

Business Benefits:

Without a business plan almost every software project would eventually fail. The strategy that we would like to proceed with to generate revenue is similar to that of Spotify, the intention is to have two alternatives upon registration to choose from, either proceed with the usage of our website for free or pay a symbolic monthly fee for a premium user account. When a client picks the first option the website will automatically allocate an ad schema for that account, meaning that earnings will be made for the clicks of ads and also from video ads. The premium features like uploading an unlimited size of files, having unlimited groups in users pool and many more will be only available to premium users. For the educational institutions we will offer the software for a one time fee.

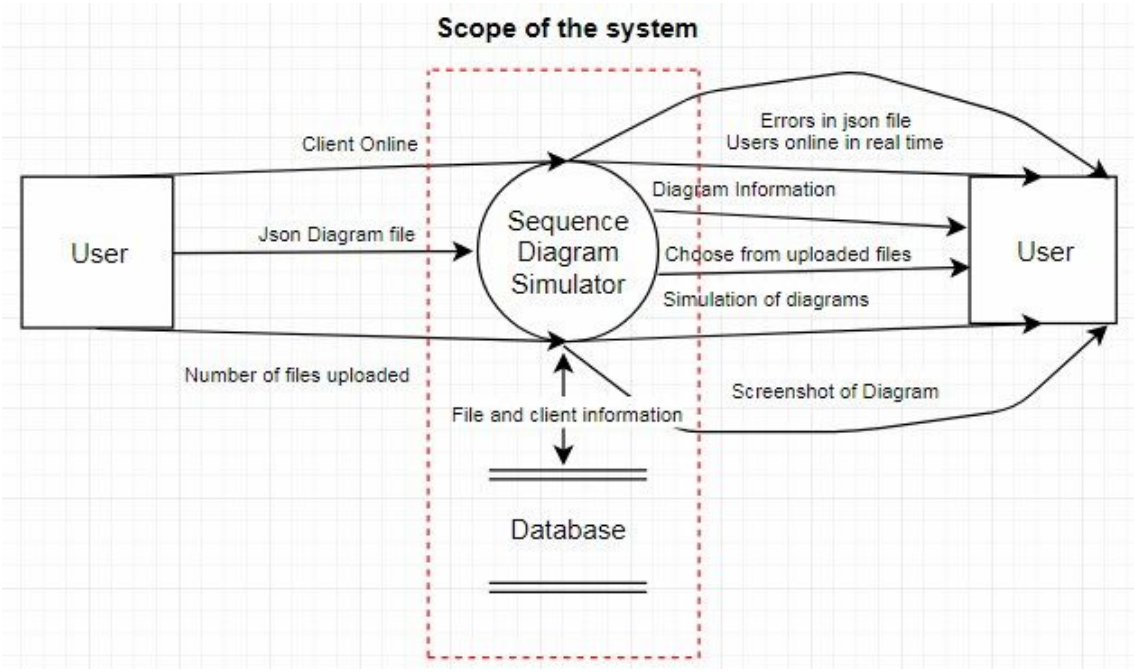
T7.

Yearly income: 90 000 €
Yearly maintenance cost: 10 000 €
Discount rate: 15% r = 0.15
Discount factor - First year: [1/(1+r)]^(t-1) = [1/1.15]^0 = 1
Second year: 0.87
Third Year: 0.76

Item	1st Year	2nd Year	3rd Year
Software Development Cost	210400	0	0
Software Maintenance Cost	-10 000 €	-10 000 €	-10 000 €
Benefit	90 000 €	90 000 €	90 000 €
Cash flow each year	80 000 €	80 000 €	80 000 €
Cash flow each year multiplied by its discount factor	80 000 €	69600 €	60800 €

Management takes about 10% from the total budget [12].
10% * 201400 = <u>20140 €</u>
Total Cost = 201400 + 20140 = <u>221540 €</u>

T8. Figure 1.2



In figure 1.2 we can see the scope of the system and also the external inputs, outputs, logical files and inquiries.Using the function point analysis which is a specific mathematical process we will estimate the size of the project in lines of code, it consists of 7 steps that will be executed in the next section.

External inputs: client online, JSON file containing the diagram information, number of files uploaded
Internal Logical file: database (file locations, users online)
External interface file: JSON file
External Output: errors in JSON file, users online in real time (changes every second) Information about the selected diagram (number of nodes, sequences and messages), choose from uploaded files.
External inquiry: simulation of diagram, screenshot of diagram.

Type	Simple	Average	High	Count	Weight
External Input	3	5	7	**3**	**5**
External Output	4	7	9	**4**	**7**
External Inquire	7	11	15	**2**	**11**
Internal Logical File	5	6	8	**1**	**6**
External Interface File	3	5	7	**1**	**5**

Table 1.1

In table 1.1 we have classified the complexity of each function type, count and chosen the appropriate weight to each of them.
Calculation of unadjusted FPs by summing weightings:

UFP =3x5 + 4x7 +2x11 + 6x1 + 5x1 = 15 + 5 + 6 + 28 + 22 = **76**

Calculation of the Value Adjustment Factor (VAF) after each of the 14 technical complexity factors have been scored from 0 to 5:

VAF = 0.65 + (*Fi* 14 *i*=1 * 0.01) => 0.65 + (45 * 0.01) = **1.10**

The last step is to apply VAF to UFP to calculate the adjusted FPs:
AFP = UFP * VAF = 1.10 * 76 = **84**
LOC = 84 * 54 = **4536** => KLOC = **4.5** (The value 54 is from the website that was provided in our lecture slides for the programming language C#).

The COCOMO method stands short for Constructive Cost Model, and it will be used to calculate the development effort and development time for the project. Before we start with the mathematical formulas we have to fit our project into the three stated complexity types, since it doesn't have rigorous requirements and the development team is small we can classify it as Organic project.

Project type	a	b	c	d
Organic	3.2	1.05	2.5	0.38

Table 1.2 - values for organic projects

Development effort = a * 4.5 ^ b = 3.2 *4.85 = **15.5** persons-months
Development time = 2.5 * (15.5) ^ 0.38 = **7** months
Average staffing: 15.5/7 = 2.2 => +/- 3 persons (In my opinion at least three people are needed for the development team, just in case if team members get ill or can't work by any other unspecified reasons).
The salary for the developers and the manager for the first three years would be:

The amount requested from the sponsor was 221540 € but not all of the money can be used for salaries since a part of the resources may be spent for training developers, hiring consultants, late delivery back up plans. Back up money 40 000 €.
Salary Calculations: 21 540 € for project management (task 7). 221540 - 61540 = 180 000 €
180 000 € / 3 people x 3 years = 20 000 € per year for developers.

T9. Gantt chart for the project schedule

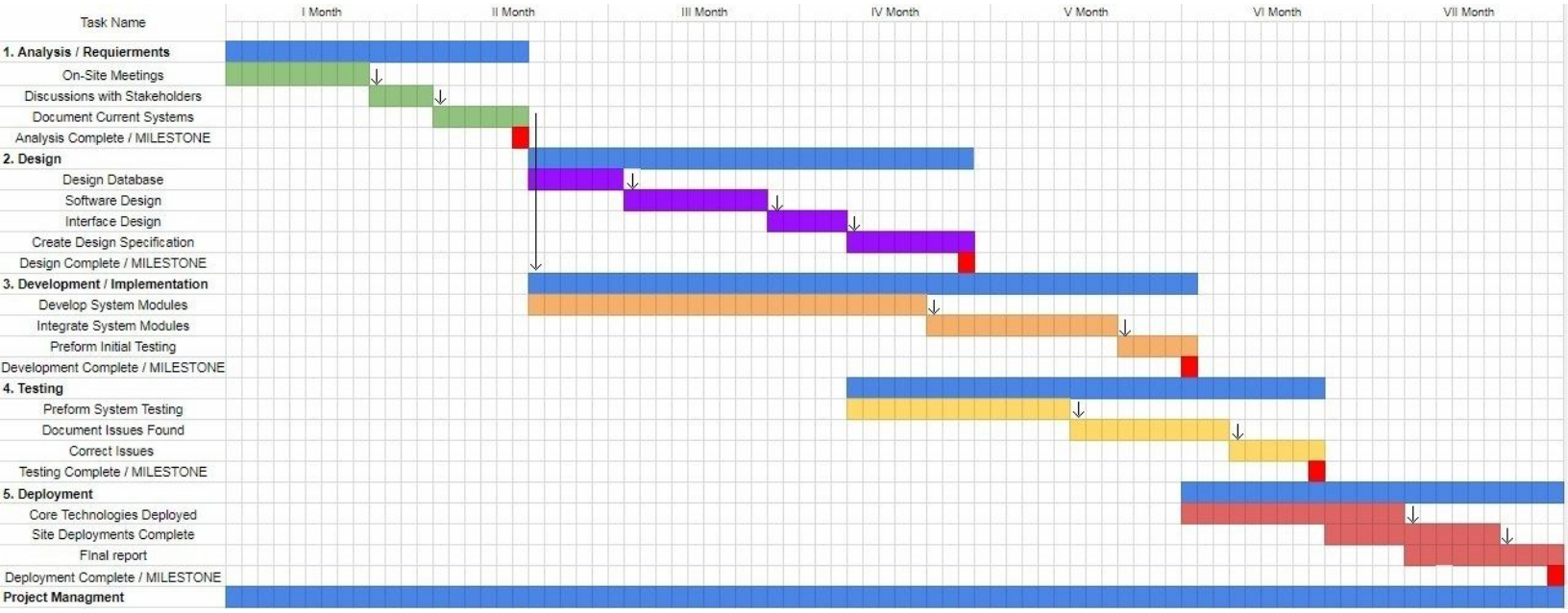
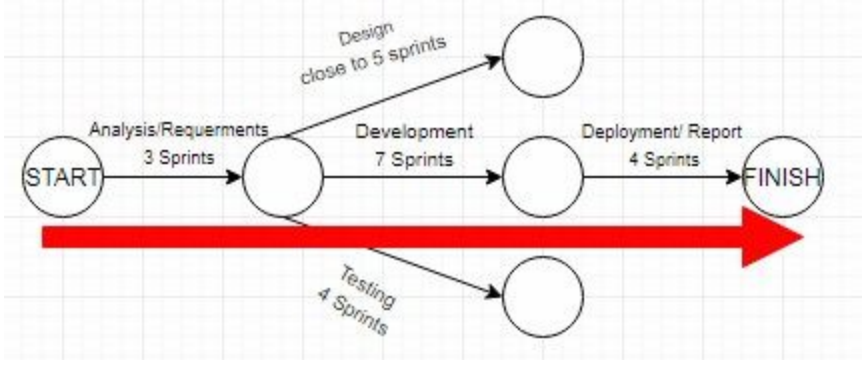


Figure 1.2 (Click Here for the high quality picture)

From figure 1.2 we can see the the time span of the whole development process is split in 7 months and to be more specific according to the iterative and incremental process that we are using those 7 months can be divided into 14 agile sprints. The dependencies on the gantt chart are stated with an arrow pointing downwards meaning that before continuing with the next task. Milestones are marked as red rectangles at the and of every phase, and the deliverables for them are:
I Phase: Definition of Terms (Glossary),Process Flow Diagram and functional requirements specification.
II Phase: Class Descriptions, GUI Mockups, 4 + 1 Architectural View and Network Architecture.
III Phase: Fully working prototype,Updated UML diagrams and documentation and Updated Functional Specifications.
IV Phase: Integration Test results, Regression Test Results, Stress Test Results, Usability Test results , Data Validation and Quality Attribute Scripts.
V Phase: Complete and full documentation and the software product.

Figure 1.3 Critical path:



T10.

In the following section i will identify some of the risks that may occur during the development of the software, they will be assessed and a mitigation strategy will be addressed for every single of them.The assessment is done according to the impact the risks would have, if they have occurred, which would translate to the extended time that is needed for the project to be completed.

Business risk:

1.Costs associated with late delivery

Description: Sometimes in the process of software development the project isn't delivered in time which results in unwanted resources being spent on consulting or other solution methods.

Likelihood: Medium *Impact:* High

Mitigation Strategy: Project Management should split down a project into several sprints of planning, development and feedback, this will allow us to make incremental progress and will provide that we stay on course to accommodate the goals of the project, meaning that we should map out the scope of the project so we have a good idea of all of the elements that are needed to satisfy the goal.

Project risks:

2. Scheduling

Description: Unrealistic time estimates usually do occur in individual software projects and produce complications for developers and managers. Many factors contribute to this risk like the issues that may occur with the development some of the requirements, low budget, non-experienced developers and many more.

Likelihood: Low *Impact:* Medium

Mitigation Strategy: The project manager should monitor the existing project, have stand up meetings after every sprint to make sure that everything is on track and of course have a backup plan if things are not going as planned [11].

3. Gold Plating

Description: Frequently developers want to show off their skills by unnecessarily adding features that have not been specified, going beyond what is covered in the requirements. For example, a programmer might add Flash animations to a login module just to make it look “cool”, this results in a waste of programming hours.

Likelihood: Low *Impact:* Low

Mitigation Strategy: This can be avoided at the time of preparing the detailed scope statement or statement of work (SOW) by ensuring that unwanted material or work are not taken into account and then have a proper Scope Management Plan to review and change the SOW at any stage of the project if it is felt that gold plating has taken place knowingly or unknowingly.

4.Poor communication between team members

Description: The lack of communication during the multiple project phases can arise between project management and developers or even between individual developers and this can lead to conflicts that can harm the organization.

Likelihood: Low *Impact:* Medium

Mitigation Strategy: Firstly, the project manager should gather all the people involved in the project, he should talk with all of them and identify the issues and understand their interests. Secondly, several options for solving this problem should be addressed like scheduling daily status meetings, work retreats, using software communication tools like Slack, etc.

Product risks:

5.Security

Description: During the testing phase it may be realized that there is a serious security flaw that would motivate hackers to breach the system, gain control or steal our data.

Likelihood: Medium *Impact:* High

Mitigation Strategy: Project management needs to be briefed, so they can put more testers and developers on a short notice which would result in increased costs and time. Assuming that the problem is not resolved by employees in our organizations, the next approach would be contacting a security consulting firm to help us for a considerable cost [11].

Part 2

Q1.

Some of the quality tools and methods that are used at the telecom giant Ericsson that have high success are: Agile/Lean, Continuous feedback, RCA (Root Cause Analysis) and Fault slip through analysis. The positive effects of Agile/Lean are that the client is involved in every step of the project so there is a high collaboration between the client and project team.With the usage of time-boxed, fixed schedule Sprints, new features are delivered quickly and frequently, with a high level of predictability, because each Sprint has a finite duration, the cost is predictable and limited to the amount of work that can be performed and also it comes with the liberty to constantly refine and reprioritize the product backlog [5].The continuous feedback is used during a performance appraisal to administer more knowledgeable and accurate performance reviews. RCA and FST are a part of the feedback analysis methods and the first one is used for solving the problems at the root, instead of fixing the distinct issue which eventually may lead to an organisational change. Advantages of the second method are: fewer stopping faults, less redundant testing, learning from earlier mistakes and avoid doing them again, early and cost effective fault detection, shorter lead times and improved delivery precision and many more [6].

Q2.

Speed improved the quality at Ericsson by doing continuous integration on main track, monthly releases, automating the whole flow. The pros of these principles are increased customer usage of latest software, shorter lead time of value to the customer, decreased inflow of customer trouble reports, decreased number of open faults. Continuous delivery & release which is the software engineering approach used by them is responsible for most of above mentioned benefits, in which teams develop software in short cycles, making sure that the software will be reliably released at any point of time. The goal of this approach is at building, testing, and releasing software faster and more frequently and that

leads to a reduced cost, time, and risk of delivering changes by ensuring space for more incremental updates to applications that are in the development process. This adaptation has lead the company to release software updates every month, transition being live in an operator's network from as little as two days, compared to six to nine weeks before and also customers are saving up to 50 percent in operational spending by deploying the software in smaller and more frequent releases using automated acceptance tests [7].

Q3.

First of all, cyclomatic complexity is a metric that was invented by Thomas McCabe and it is used for calculating the complexity of a method by giving a single number.In structured programming the result is found upon the number of loops and if statements in the function. There are many tools like GMetrics, SonarQube, Eclipse Metrics Plugin that do all the work for us, but they may give out different values for the same method. The diverse results are because every tool is implemented differently and also the original paper from McCabe was vague on some details of the metric. So why do we use it, in the context of testing code, it can be used for an estimation of the effort we have to put in to write test cases and i believe that this is a valid use case for cyclomatic complexity. On the other hand, high complexity is directly translated to low readability and immense maintenance cost, but this is not totally correct and i will explain in the next few examples.

***I** method*

```
String getMonthName (int month) {
    switch (month) {
        case 0: return "January";
        case 1: return "February";
        case 2: return "March";
        case 3: return "April";
        case 4: return "May";
        case 5: return "June";
        case 6: return "July";
        case 7: return "August";
        case 8: return "September";
        case 9: return "October";
        case 10: return "November";
        case 11: return "December";
 default: throw error
```

***II** method*

```
String getWeight(int i) {
    if (i <= 0) {
        return "no weight";
    }
    if (i < 10) {
        return "light";
    }
    if (i < 20) {
        return "medium";
    }
    if (i < 30) {
        return "heavy";
    }
    return "very heavy";
}
```

***III** method*

```
int sumOfNonPrimes(int limit) {
    int sum = 0;
    OUTER: for (int i = 0; i < limit; ++i) {
        if (i <= 2) {
            continue;
        }
        for (int j = 2; j < i; ++j) {
            if (i % j == 0) {
                continue OUTER;
            }
        }
        sum += i;
    }
    return sum;
}
```

The first method has a cyclomatic complexity of 14 and that is way over the recommended value for a function, certainly many tests are needed to test this method, but speaking for myself i wouldn't need that much time to understand or change this function since it is only a simple switch case with 11 cases. This raises my point that even though the result was high it didn't lead to low readability or huge maintenance cost. The following two methods both have the same complexity of 5, however as we can see the first one contains 4 if statements and the other has two nested if statements inside two loops, meaning that is way more complex than the other. I think, cyclomatic complexity as a single measurement isn't a good metric that can evaluate the complexity of a function and this is concluded from the previously mentioned examples. To measure it correctly we have to consider using other metrics together with McCabes approach, like length of a function and nesting depth. This way we can bind and compare the different results and have a better understanding of the complexity for a specific method [8].

References:

1. Software Products versus Software Services - LINK
2. Key Phases Software Development - LINK
3. The Project Management Triangle - LINK
4. Project Management for Information Systems (5th Edition) - page 209
5. Benefits of Agile Software Development - LINK
6. Software Development Process - LINK
7. Quality Assurance in Software Development - Jörgen Uddenheim & Lars Stålheim
8. Mccabe Cyclomatic Complexity - LINK
9. Lecture_0 DIT885 HT17 - Gul Calikli Slides by: Björn Olsson
10. Iterative & Incremental development model - LINK
11. Project Management for Information Systems (5th Edition) Chapter 15
12. Lecture_3 DIT885 HT17 - Gul Calikli Slides mostly by: Björn Olsson - Slide 63