SCHOOL OF INFORMATION TECHNOLOGY

MASTER OF INFORMATION TECHNOLOGY

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

INDIVIDUAL ASSIGNMENT

COVER PAGE

Name of Student	Leeto Modutoane
Student Number	16081928
Name of Module	Life Cycle & Maturity Models for IT
Module Code	MIT 850
Name of Lecturer	Dr. Patricia E.N. Lutu
Date of Submission	08-August-2016
Contact telephone number	071 984 6883/011 223 7106
E-mail address	Leeto.modutoane@gmail.com
Declaration:	*I declare that this assignment, submitted by me, is my own work and that I have referenced all the sources that I have used.*
Signature of Student	[illegible]

Date received	
Signature of Administrator	

Question 1 (a)

Executive/Political leader’s support and allocation of inadequate resources

From Charette, (2005), I have learned that allocation of inadequate resources can impede the success of a project, and it can have negative consequences. The article indicates that one problem related to business factors is the need to cut down on costs. IT executives expect IT departments to do more and faster with less funds which is a problem while IT resources are increasing in price. They do not see IT projects as an investment but as pure unnecessary costs that must be controlled. The article also gives an example of the airport automated baggage-handling system where local political leaders had held the project to one unrealistic schedule after another, which delayed the project and the opening of the airport.

Based on the above statement it is clear that government or organisational politics can disrupt project success as they can impede project's schedule, cost, and quality. As a result project managers often become dishonest by providing unrealistic expectations, overpromising what the project will do, how much would it cost, when it will be completed. When this happens, developers are the ones who have to make up for the shortfall which puts them under pressure which leads to the likelihood of error and, ultimately, failure.

The lesson learned here is that lack of executive and political support can lead to project failure. Stakeholders need to support IT and project managers who on the other hand need to support other technical staff for the project to become a success. There needs to be openness, honesty, communication and collaboration amongst all participants, politicians, company executives, IT and Project Managers, systems and business analysts, programmers, testers, clients and users.

The project manager is the key resources to the project’s success

According to Charette, (2005), to determine project failure we look at business management, project management and organisational culture. Project Mangers play a very big role in software projects and can also be a major source of errors that may lead to project failure. Bad choices by project managers are possibly the single greatest cause of software project failures and their mistakes are normally not as easy to fix as, their mistakes can lead to havoc. Big software project failures may resemble the worst conceivable aeroplane crash where the pilot had little experience and flew into a storm with an untested aircraft. This according to my interpretation means that you cannot give a big IT project to an inexperienced project manager.

The lesson learned here is that for a big project to be a success you need well trained and experienced project managers. Based on past experience, a lot of organisations out there see a project manager as an unnecessary role as they often assign the business analyst to also act as a project manager. This is a lesson that project management is a key role and it must be given to experienced project managers with the right experience.

Project Size matters and we need to learn from previous mistakes

Charette, (2005) says that large project failures can destroy the company’s entire future, and failure of such project is about 15 to 20 percent. Studies indicate that large projects fail 3 to 5 times more often that the small ones. The larger the project the more complex it becomes. Greater complexities leads to extra error possibilities because no one clearly understands the different components of the entire project.

Software failure is predictable and avoidable, the lesson learned here is that we know why projects fail, there is plenty of research on project failure yet we are still unable to or unwilling to learn from previous mistakes. In large projects we need to pay extra attention and as the article indicates there also 134 recommendations from the DOD and all of this recommendations must be applied in order to prevent project failure. Another lesson is that the bigger the project the higher is the risk of failure and experienced personnel can minimise the possibility of risk.

Badly defined system requirements

The article indicates that some organisations like the software development firm Praxis requires that the customer be fully and actively involved in the project especially when it comes to understanding and defining the customer's requirements. Praxis makes sure that there is understanding between them and the client before any development starts on what is desired, what is feasible, and what risks are involved with the given resources.

The lesson here is that before any development work is started the system and user requirements are clear and based on my past experience this can lead to serious project delays and users distancing themselves from the finished product.

Question 1 (b)

i. In my opinion the waterfall process was not the suitable process for the Taurus project simply because it was a very complicated project, involving novel technology and an ambitious schedule for which no protocols existed. Charette, (2005), the fact that there were no protocols in existence and Novel technology was used it meant that requirements were meant to change and therefore a more iterative process would have been more suitable.
Taurus was expected to go live in in October 1991 yet by the expected delivery date the schedule had slipped 100% moreover the security, legal and technical requirements were much more complex than had been expected. At that point it was clear that the expectations could not be met yet development continued for almost 2 years before the project was abandoned.
The project team were repetitively expected redo work due to changing requirements. There were too many diverse constituencies involved including banks, brokers, company registrars, international custodians and they all wanted different things which affected the requirements and design processes.

This shows that the waterfall process was not suitable because (Base36, n.d.) says that waterfall is good in projects were the expected outcome, size, cost and timeline of the project are well know in advance which is not the case with Taurus as there were no proper procedures in advance to map the expected processes, there were so many stakeholders with different expectations which led to frequently changing requirements, a more iterative process would have been more suitable.

ii. In my opinion there waterfall process was not also followed properly, waterfall methodology is a sequential design process, with successive stages Boehm, (1988) and following an order of steps with iterations between the preceding and the successive steps not with the more remote steps which was not properly followed in Taurus as Drummond, (1999) indicates that the project director decided to design the outwards part of the system, first leaving the central architecture until last which is the core of the system especially in complex systems such as Taurus with multiple components.
The use of waterfall therefore did not result in a successful software project because the right waterfall processes were not followed properly and the other main reason is because of the changing requirements which cannot be handled well through waterfall processes, as stated above a more iterative process would have worked better.

iii. My opinion is that the Spiral model would have worked much better, and the Taurus project might have been a success if the Spiral process was used, else it would have been realized earlier that the project will never be a success.
This is because Boehm, (1988) says that the spiral process is more risk driven, it places more emphasis on risk and therefore the risks leading to failure would have been realized much earlier. Another good thing about it is that as part of the risk identification process alternative solutions are also identified and alternatives to packages like Vista, issues with Taurus would have been identified earlier, if vista is not working the alternative could have been considered much earlier, prevented failure and resulted in less costs.
Boehm, (1988) also continues to say that SPIRAL is a viable framework for integrated software system development which is exactly the Taurus architecture, and another advantage of Spiral being that it focusses on elimination of errors and unattractive alternatives in early stages, and through all the points above from Boehm, (1988)'s article.I would say the Spiral Process would have worked much better than the waterfall process.

Question 2 (a)

Communication
There was a communication problem, because users were not involved and they felt as if they were not part of the project, in the LAS case there was no attempt to sell the system to users, to reduce resistance to change and thus reduce anxiety due to the change in normal processes. Such a major change to the organisational culture requires careful management and communication from management to users and clients which was not enough with regards to the LAS case study. Dalcher, (1999)

Planning
The problem here is that there was no experienced project managers that could manage the work schedule and risks, they had little experience in PRINCE the proposed project management methodology and also there was no quality assurance on the system Dalcher, (1999). Planning from project management's perspective and

quality processes we very poor, with the magnitude of the project it was really doomed to fail. This project required dedicated quality assurance and project management teams for it to work.

Modelling
Processes were not mapped properly, because the implementation of a new system should enhance processes and not make the situation worse for the users, in the LAS case-study all allocations and communications will be handled by the computer which enhanced certain processes and reduced paperwork but on the other hand other processes were worsened, for an example the ambulances were now allocated to locations away from their home stations which led to delays as they would get stuck in traffic after the end of the shift, this is the time which is going to waste as the ambulances are now stuck in traffic instead of attending to calls while they are trying to get back to the home station. Dalcher, (1999). In this case processes were not well understood, and therefore not well implemented because the user, ambulance drivers were never fully involved in order to understand the user requirements.

Construction
The problem with construction is that the allocated 11 months for re-development was not enough, especially that is was non-negotiable. It's a very complicated automated system with minimal human participations so the proposed time would just pressurise the development team and lead to failure. There is no way that the code to build such an immense system would be generated in such a short space of time. Dalcher, (1999)

Deployment
The problem with deployment was that it was decided to roll-out the system all at once, the team decided to go for a big bang implementation instead of introduce the system in pilot phases. For such a big system it would have been better to introduce the modules in phases to reduce risk and anxiety from the users. Dalcher, (1999)

Question 2 (b)

Tracking and Control
The problem here is that there is no experienced staff and suppliers in project management to maintain the project schedule, PRINCE was selected as the project management methodology and training was provided to the inexperienced resources but the problem is that very little of the gained knowledge was used or any alternative methodologies to guide the project schedule. Dalcher, (1999)

Quality assurance
Dalcher, (1999) indicated that Systems Options had not performed any quality assurance on the system, no test plans existed, no integration tests were ever and unit tests were conducted on their own code and a project of such magnitude can never be a success without proper processes to ensure its quality, there should have been a dedicated quality assurance team to deal with the quality of the system especially that it has to do with people's lives.

Technical reviews
Dalcher, (1999) indicated that the system had known errors before the go live date on Monday 26 October 1992 but the system was till put to live with about 81 outstanding issues, having performed no stress testing on the full three phase system, no backup plans were in place. This issues should have been fixed before going live or at least the system should have been thoroughly tested before being put on live, stress and integration tests should have been performed before and also on going live the current manual process should not be abounded immediately is should run in parallel with the new system until it can be fully trusted, serving as a backup should the system fail.

Question 3(a)

i. **Essence** – this are difficulties inheritable in the nature of software itself and forms part of the characteristics of software, it refers to conceptual errors in most cases related to the specification, design and testing of software constructs.
   Accidents – this are the difficulties that are not inherent, can be created and fixed by developers.
ii. **Complexity** – software is very complex for their size unlike many other human constructs, no two parts are the same in software unlike with hardware computers, buildings where the use of repeated elements is normal.
   Conformity – software is expected to conform because it interfaces with other human institutions and systems, and because it's the component that has only came to the scene recently and it is the most conformable.

Changeability – software is embedded or developed for users, laws, machine vehicles e.t.c and all these change regularly. Software must always comply and confirm to the changes in all this other products.
Invisibility – software is invisible and un-visualizable, there may be charts and architectures but inter-workings of software can never be visualized.

iii. **Accidental Complexity** – this is the complexity brought by abstract programs which lack constructs already embodied in high-level languages. Abstract programs lack constructs built into high-level languages which complicates programs as programmers are expected to write the constructs as part of the program.
Slow Turnaround of batch programming – this is an accident difficulty that is attacked by time-sharing, as it preserves immediacy. The difficulty is related to the slowness of batch programming when compiled and executed, which takes up a lot of time. This is an interruption to the programmer's consciousness which is very costly in time. The system response time is very slow.
Using Programs together – "this is the accidental difficulty of using programs together, by providing integrated libraries, unified file formats, and piles and filters" This is the difficulty to integrate multiple programs or components in the same program

Question 3(b)

i. **Ada and other high-level languages advances**: its contribution included features such as modularization and step by step solutions and thus advancing the structure of software programming. Brooks, Jr, (1986)
Object Oriented Programming: there are to ideas under the concept, abstract data types and hierarchical types. Abstract types refers to objects and hierarchical types refer to classes which can have sub-classes. This is where a software project can be broken down into objects and classes to reduce code redundancy. Brooks, Jr, (1986)
Artificial Intelligence: can be described as the use of computer to solve problems that could originally only be solved though applying human intelligence.
Expert Systems: "programs with a general inference engine and a rule base, built to take assumptions and input and explore logical consequences through the inference derivable from the rule base, yielding conclusions and advice and offering to explain its results by retracting its reasoning for the user." Brooks, Jr, (1986)
Automatic Programming: this refers the solving of a problem from a statement of problem statement through automated code generation of a program. Brooks, Jr, (1986)
Graphical Programming: this refers to the use of computer graphics like UML to design programs. Brooks, Jr, (1986)
Program Verification: it's a mathematical proof used to establish that a program meets its specification and thus reduce program testing load, but can also never eliminate program testing as it is very important process. Brooks, Jr, (1986)
Environments and Tools: these refers to better programming tools and environments, research for example into language specific smart editors and integrated database systems. Brooks, Jr, (1986)
Workstations: it's about the more powerful workstations with increased memory capacity, computers power and what benefits would they bring to the software development world. Brooks, Jr, (1986)

ii. **MVC**: Model View Controller model separates concerns, allows for code reuse which finally help developers fight against code redundancy and achieve ease of maintenance with less code to maintain.
Business Intelligence: the best solutions to get companies to be more efficient and help company executes answer business questions quicker and eliminate guess work. This replaces the database and excel reports for managers which may be difficult and time consuming for executives to interpret.
Web Services/WCF: are platform independent and enable multiple systems to communicate easily over the internet.
Mobile Development: Mobile development enabled us to communicate and work from anywhere and whenever we want by being able to access our emails and work from anywhere, this made life much easier and more convenient.
Mobile Devices: mobile development doesn't mean much without devices such as mobile phones and tablets, this also enable us to study and work from anywhere, whenever we want.
E-Commerce: ecommerce enable us to anywhere and anytime.

Bibliography

Base36. (n.d.). *Agile & Waterfall Methodologies – A Side-By-Side Comparison*. Retrieved 08 06, 2016, from ase36.com: http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison/

Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer, 21*(5), 61-72.

Brooks, Jr, F. P. (1986). No Silver Bullet - Essence and Accident in Software Engineering. *The IFIP Tenth World Computing Conference* (pp. 1-16). Amsterdam: The Mythical Man-Month Anniversary edition with 4 new chapters.

Charette, N. R. (2005, September). Why software fails. *IEEE Spectrum*, *42*(9), 42-49.

Dalcher, D. (1999). Disaster in London: The LAS Case study., (pp. Proceedings of the IEEE Conference and Workshop on Engineering of Computer-Based Systems (ECBS)). London.

Drummond, H. (1999). Are we any closer to the end? Escalation and the case of Taurus. *International Journal of Project Management*, *17*(1), 11-16. Retrieved 08 06, 2016

Pressman, R. S. (2005). Software engineering. *R.H. Thayer & M.J. Christensen (eds), Software Engineering, 1*. Retrieved August 06, 2016

Royce, W. W. (1970). Managing the development of large scale software systems: concepts and techniques. *Proceedings of Wescon.*