SOFTWARE ENGINEERING PROJECTS THAT FAILED DUE TO POOR REQUIREMENTS ENGINEERING – A CASE STUDY

Week 1

Requirements engineering is perhaps the most crucial part of a software engineering project. It involves one of the most decisive tasks of gathering necessary information in order to come up with correct software. Many projects are misled and fail or at least face some challenges due to poor requirements engineering. This paper discusses real world examples of software projects that failed due to poor or insufficient requirements engineering.

1. TAURUS Project (1993)

With a total estimated failure cost of £480 million, the TAURUS (Transfer and Automated Registration of Uncertified Stock) project was cancelled because of conflicting requirements. It ranks as one of the major losses in the history of business. Set to change the trading and regulatory context of the whole security industry, it was a highly complex project.

The purpose of TAURUS was to convert the London paper-based stock exchange system to electronic transmission. The initial aim and attempt was to produce a simple system which could have been built within 6 months. However, with some foreseen commercial disadvantages, thee requirements changed to produce a proposed system that would satisfy everyone. The industry had a broad variety of departments (banks, brokers, etc.) and almost every department wanted something different. Even after two years, the project still didn't progress and was in conflict. Later, certain parties involved, including the chief executive Peter Rawlins discussed the project was getting unnecessarily complicated. Although the project was making progress gradually, the feasibility reviews indicated that the minimal date of project finalization would be in April 1993. Even so, while the previous requirements were getting fulfilled as the project progressed, newer issues were arising. The involved parties believed the system would work but require more time and money. This signalled that the system would require further 3 more years to complete. The chain of events led to the chief executive announcing the cancellation of the project. The TAURUS project was then officially cancelled in March 1993. (Drummond, 1999)

2. Denver International Airport Baggage Handling System (2005)

The Denver International Airport Baggage Handling System is another major example of a project failed due too poor requirements engineering phase and conflicting decision making. The project had to be abandoned before being replaced by a manual one in 2005. The project cost a lot of money and at some point, it even cost around $1 Million per day just on interest from loans. Although the purpose was to automate the baggage handling system for good efficiency, the complexity and conflicting decision making led to a lot of problems.

The opening day saw the plan implemented only on one concourse instead of three where the other two still used manual system. The key decision makers did not account the risks and complexity involved. Being a large automated system, the desires were over ambitious. This delayed the project's start. Initially, after some way through the project, strategies were changed, and many things had to be reworked. The major mistake was inability to link the overall airport's plans to the automated baggage system's plans. Even so, the decision was made to progress the project. The time was already limited because of the delayed start and conflicting strategies, and thus, the employees tended to skip or wrongly do different parts of the project. This rushed work led to disastrous events where the media disclosed baggage colliding and crashing. Even while the project progressed, many change requests were hastily implemented into the project which further delayed and rushed the buggy platform.

Another point of failure was for the physical building where the baggage system was to be implemented. The building design was already completed before the baggage system, and as such, the baggage team had to redesign the physical structure for the autonomous system. This also costed an estimated $100M. Eventually, the pressure from different parties and the embarrassing highlights presented by the media, led the project to be abandoned completely. The system, never working well, was abandoned finally in 2005 as even the maintenance cost for the automated system exceeded the cost required for a full manual system. (Calleam.com, 2008, Montealegre and Keil, 2000)

REFERENCES

Drummond, H. (1999). *Are we any closer to the end? Escalation and the case of Taurus.* International Journal of Project Management, **17**(1), pp.11–16. Available from DOI: 10.1016/s0263-7863(97)00074-4 [Accessed 27 Oct. 2019].

Calleam.com. (2008). *Denver Airport Baggage Handling System Case Study – Calleam Consulting*. [online] Available at: http://calleam.com/WTPF/wp-content/uploads/articles/DIABaggage.pdf [Accessed 28 Oct. 2019].

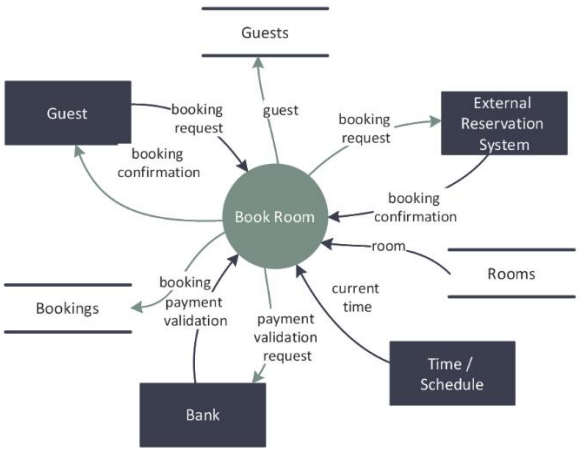
Montealegre, R. and Keil, M. (2000). *De-Escalating Information Technology Projects: Lessons from the Denver International Airport.* MIS Quarterly, **24**(3), pp.417. Available from DOI: 10.2307/3250968 [Accessed 28 Oct. 2019].

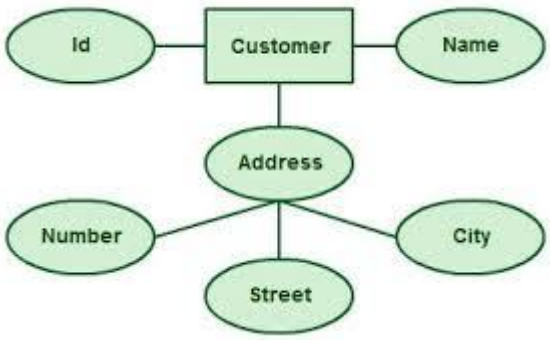
Possible Limitations	1	2	3	4	5	6	7	8	9	10
Needs pre-existing system	X			X						
Needs paper-based pre-existing system	X			X						
Needs customer requirements document						X				
Needs considerable expertise		X					X	X		X
Mainly useful in initial stages	X			X		X	X	X		
Mainly useful in later stages										
Relatively expensive				X			X			X
Inflexible			X			X				
Only gets information from inside heads		X	X					X	X	
Only gets information from outside heads										
Possible Strengths										
Wide Scope		X	X	X				X	X	
Good for getting background information	X	X					X	X	X	X
Good for getting problem domain information	X	X	X	X		X	X	X		X
Good for getting requirements		X	X			X	X			X
Easy to apply	X		X	X					X	
Relatively cheap	X		X			X				
Flexible		X					X	X		X
Can address a large target population			X						X	X
Has high user/client involvement		X	X	X			X	X	X	X

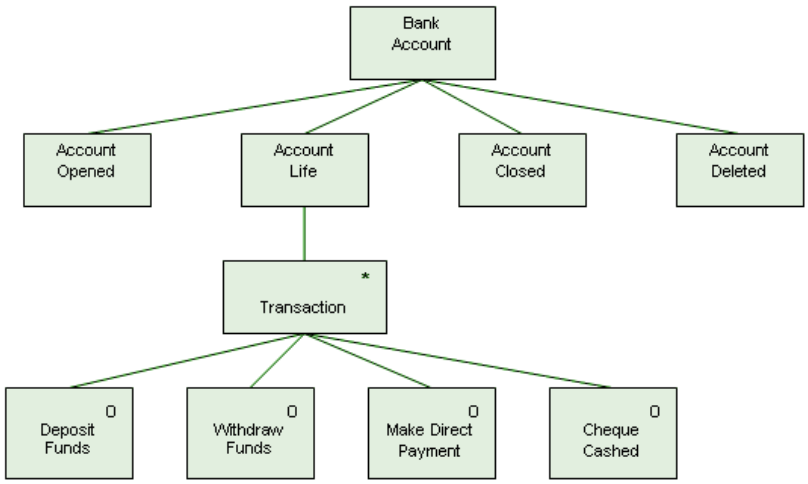
1. **Key: - (Technique Number)**

1) BACKGROUND READING
2) INTERVIEWING
3) QUESTIONNAIRES
4) TASK OBSERVATION
5) ETHNOGRAPHY
6) REQUIREMENTS STRIPPING
7) JOINT APPLICATION DESIGN
8) LADDERING
9) BRAINSTORMING
10) REQUIREMENTS WORKSHOP

Week 3

Analysis Technique	**Description**
Data Flow Models (DFDs)	Data Flow Models are one of the most common techniques that fall under structured analysis. A Data Flow Model is "a diagrammatic representation of the flow and exchange of information within a system. It is used to graphically represent the flow of data." (Techopedia.com, 2019). It accounts the processes and activities of different parties in order to come up with a diagram that represents data flow. These DFDs can be used to decompose system into smaller subsystems. **Strengths:** • It can be used while describing the boundaries of a system. • It is easy to read and can be useful to provide system's knowledge to the stakeholders as it is comparatively readable by non-technical users. **Weaknesses:** • It takes a long time to produce the diagram. • As it cannot be directly implemented as objects, the programmers could have a tougher time.  *Figure: Example Data Flow Diagram (Conceptdraw.com, 2019)*

Entity Relationship Models (ERDs)	Entity Relationship Models are another type of diagrams that represent business entities and their links to each other. An entity can be defined as "A piece of data or an object or concept about which data is stored." (Beal, 2019). The ERDs generate an abstract diagram which can be used to design a system's database. There are three basic terminologies that are of interest in an ERD: the entities, relationships and attributes. The relationships are the links between entities. The attributes are the properties of an entity. **Strengths:** • Allows straightforward representation of the database model. • Easy conversion to other models such as relational models or data models. • The clear representation of entities, relationships and attributes allow clear understanding of the data and helps in minimizing redundancy. **Weaknesses:** • Not all information is seen in an ERD. • It is specifically used for High Level Designs. • Data manipulation is not represented by an ERD.  *Figure: Example ERD (Drivingwinter.ga, 2019)*

Entity Life Histories (ELHs)	Entity life histories tell about what happens to an entity over time. These could be useful as they help in database design. TechnologyUK provides a more complete definition of Entity Life History as "a diagrammatic method of recording how information may change over time and models the complete catalogue of events that can affect a data entity from its creation to its deletion, the context in which each event might occur, and the order in which events may occur" (Wells, 2019). While it is a rather difficult task to observe and model all the entities at the same time, the modelling is done one at a time. Entity life histories can have different elements such as sequence, iteration, selection, etc.  *Figure: Example ELH (Technologyuk.net, 2019)*

REFERENCES

Techopedia.com. (2019). *What is a Data Flow Model? - Definition from Techopedia*. [online] Available at: https://www.techopedia.com/definition/28523/data-flow-model [Accessed 30 Oct. 2019].

Conceptdraw.com. (2019). *Data Flow Diagram Examples*. [online] Available at: https://www.conceptdraw.com/How-To-Guide/data-flow-diagram-examples [Accessed 30 Oct. 2019].

Beal, V. (2019). *What is Entity Relationship Diagram? Webopedia Definition*. [online] Webopedia.com. Available at: https://www.webopedia.com/TERM/E/entity_relationship_diagram.html [Accessed 30 Oct. 2019].

Drivingwinter.ga. (2019). *ERD Diagram Examples*. [online] Available at: https://www.drivingwinter.ga/erd-diagram-examples/ [Accessed 30 Oct. 2019].

Wells, C. (2019). *Entity Life Histories*. [online] Technologyuk.net. Available at: http://www.technologyuk.net/computing/software-development/systems-analysis/entity-life-history.shtml [Accessed 30 Oct. 2019].

End of Week 3