SWEN90016 – Software Processes and Management, Semester 2, 2022

Assignment 1

Student Name: Chaojun Tang
Student ID: 1323782

*Note: to avoid confusion in this report, these are the definition of each term in this case:

- Client: Resilienza, or the executives in Resilienza who wish to develop 'ConquerIT'.
- development team: Since no additional information was given regarding whether the developing team will be internal employees or an outsourcing team, this report assumes it will be an internal developing team.
- Employee: general employees of Resilienza, excluding the developing team.
- Customers: general customers who are seeking services from Resilienza.
- Engineer: customer service engineer who directly communicates with customers.

Question 1: Business case (need) for the project

The original business model has limited the future business expansion of Resilienza due to its outdated IT system and time-consuming per-case processing. Resilienza wishs to develop an interactive IT platform in which time-consuming communication can be replaced with programs, and customers can receive faster yet premium service in the meantime. This project is set to facilitate its future business expansion, such that Resilienza could keep pace with the increasing amount of customers and keep its leading position in the market.

Question 2: Three constraints

1) New platform effectiveness & Customer acceptance

One of the major constraints of this project is how to persuade customers to shift from the using traditional service mode to the new service platform 'ConquerIT'. It is known that Resilienza has been providing its current business service model for the last 20 years. Its dedicated customer service, i.e., one-to-one discussion of business needs between customers and service engineers, is the main feature of this company. Some clients might prefer the personally tailored service (even the initial time spent on communication could be long), because less or no domain knowledge is required for the clients to get what they need.

Second, Resilienza changes the service mode to a customer semi-self-service mode requires customers to learn how to use the new platform. Indeed, this project does take into consideration customer acceptance in some ways, such as the Guided Buying and Selling engine simplifies the technical terms and knowledge to layman's language. However, it cannot eliminate the requirements that customers have to spend efforts to adapt to the new platform, which could be a deal-breaker for existing customers.

Moreover, during a one-to-one discussion over what they need, the clients feel important and are receiving premium service. In the new service mode, they will not feel the same way as they are required to do more work by themselves.

Hence, how to overcome this difficulty that let clients feel they're receiving better service is one of the constraints of this project. In other words, this project needs to be extremely user-friendly and effective to become successful, which is a major constraint for this project.

2) Developing core functions of 'ConquerIT' is difficult.

Resilienza is having its customer service engineers to providing services communicate with customers directly, understand their needs, and provide correspondent solutions. This model was built on the professionalism of each specialist. It can be predicted that converting specialists' knowledge into a systematically categorized database and providing appropriate solutions to customers is another constraint of this project.

First, the amount of data that needs to be collected is high, and collecting it is difficult. Although there is no mention of how many customer service engineers work for Resilienza, considering it has over 3000 employees worldwide and it provides one-to-one service to each client, we can safely assume there are hundreds of customer service engineers. Besides, they are working across 15 countries and each country is a different market. The amount of data that needs to be collected to enable 'ConquerIT' is high. Also, extracting knowledge from an engineer is difficult (as we have not reached Singularity). Plus, the database needs to be updated and expand frequently. Hence, collecting data is difficult.

Second, providing appropriate customer solutions is difficult. It seems that the fundamental logic of 'Guided Buying and Selling' engine is simple: customers answer non-technical questions and the system provides appropriate options. However, the designer of the system must consider the time effectiveness of each configuration, the availability of each hardware, and even the special needs of customers. Therefore, this engine is complex and will require a lot of testing to keep it at high accuracy.

Hence, developing the core functions of 'ConquerIT' is difficult and is a major constraint of this project.

3) Project development is on a tight schedule 🟥

Although there is no mention of what time frame is expected to develop the new IT system 'ConquerIT', we know for a fact that the Business Continuity Service market is expanding rapidly, and company expansion is already limited by its current business mode. A long developing time will lead to loss of customers and its market leading position. Hence, it is safe to assume that the time available for software development is relatively short.

Based on the provided document, Resilienza wishs not only to develop a new platform that can outperform the current system, but they are also planning to add new features such as data health check, small-business focused service, and cloud service to the new platform.

Therefore, with the shortened time frame and relatively large scope, the developing time is another major constraint for this project.

Question 3: Three risks

1) Business risk: ‘ConquerIT’ may not be welcomed by the market

The new system 'ConquerIT' changes the Resilienza business service mode fundamentally, from one-to-one customer service mode to semi-self-service mode. Priorly Resilienza has no experience of this new mode, nor can it predict the new system will become a success. It is possible that the new IT system is not welcomed by customers.

Customers might feel the new system is over-complex and prefer the existing servicing method. The existing system is a burden for Resilienza in terms of low efficiency and high cost, however it is the main selling point of its service where customers can have no technical knowledge and be trouble-free with their needs.

In this situation, Resilienza will either keep both old and new system running at the same time until the new system be improved which will significantly increase the running cost or lose part of potential customers that will decrease the income. Either way, if the new project outcome is not welcomed by clients, it will have a big impact on Resilienza’s business.

2) Product risk: ‘ConquerIT’ may experience unstableness.

Resilienza wishs to replace the current business mode which requires a lot of manual work from engineers with the new automated system where most of the work is being done by programs. However, there are chances that programs might fail or be unstable during operation [1]. Given the size of this company and the workload of the program, it is likely the dev team will develop a distributed system, which adds to the odds of system failure. Moreover, Resilienza has no prior experience maintaining such a program. If such an event happens, Resilienza might not be able to manage the potential damage.

In contrast, stability is an essential factor for this system as Resilienza provides Business Continuity Service. It would be quite sarcastic if Resilienza's own system fails when its clients are in need of continuity service, and possibly cause bad publicity within the market and customer loss of confidence in using Resilienza’s service. Hence, system failure or unstableness is one main risk for this project.

3) Project risk: ConquerIT’ may experience development delay

Although all projects might experience delay, the development of ‘ConquerIT’ may have a unique cause that leads to project delay. From the above analysis, we know that effectively converting specialists’ knowledge into a workable program is difficult by its nature, and we know one of the business needs is to replace large amount of manual works with programs. There is a possibility the current service engineers might fear the success of ‘ConquerIT’ will cause layoffs, unemployment or decrease their job security at least. This may drive engineers less cooperative than they should be, which could lead to project development even more difficult. Hence, the delay happens.

We have analyzed that ‘Time’ is an essential factor in the measurement of product success. If ‘ConquerIT’ get delayed by months, the result could be a budget overrun, losing a large number of customers and/or being caught up by competitors.

Risk ID	Risk	Probability (0-100%)	Impact (1-10)	Exposure (1-5)	Rank
1	End product not welcomed by the market	30%	8	2.4	1
2	Unstableness or failure during operation	5%	4	0.2	3
2	Development delay caused by feared employees	40%	5	2.0	2

Figure 1.1 Risk Analysis

Question 4: two possible Software Development Lifecycle models and their pros and cons

1) **Incremental - Formal**

One possible Software Development Lifecycle model for this project is the Incremental Model, which divides the whole project into several smaller projects, in which each of them has its own lifecycle, from planning to release [2]. In this case, Resilienza wants to develop a new IT platform 'ConquerIT' that customers can directly get service from it. It includes several key engines (i.e., 'Guided Buying and Selling engine') and several secondary goals ('Cloud Recovery service').

This model suits this project because the scope of this project is large; however, each goal is clearly distinguished from the other. Therefore, it is reasonable to divide this project into several smaller stages (i.e., each engine development is one stage of its own, and all secondary goals are one stage, with one more stage for final integration). In this way, each stage has its own waterfall lifecycle from analyzing & designing to deploying & maintaining and finally releasing a tested functional product.

Pros:

a) Developers will have a clear goal in their work for each sub-project. For example, instead of the final completion of the 'ConquerIT' project, developers now have smaller and clearer goals in a shorter time frame. Compare to agile mode, where the developing process can be complex, incremental model is relatively simple and easy to execute. This would be helpful to improve the work efficiency and keep the team morale high (staff can see 'tangible' outcome of their work). Similarly, project management & quality assurance would be more convenient for project managers as the sub-project scope is smaller.

b) Change of scope during the middle of the project is less costly than the waterfall model. Scope changing during the middle of a project could lead to bigger problems because the whole project was analyzed and designed at the initialization stage. Project managers might need to revisit their plan and modify the rest of the project plan [3]. The incremental lifecycle model can reduce the cost of change of scope, since each sub-project is a full project on its own. Changing of scope will impact a lot on its own, but less impactful to other sub-project.

c) Third, by using the incremental lifecycle model, Resilienza could put partial functionality of 'ConquerIT' in use once this part of the project is deployed (released). Partial functionality can be tested and can contribute to the current business. Moreover, Resilienza could analyze the performance of this functionality earlier and provide feedback or change of scope to project developing team. In this way, Resilienza can be benefited from its investment earlier, and get a better performance outcome when project finished.

Cons:

a) Since the whole project is divided into several smaller projects, and each smaller project is a full project by itself, the total resources allocated will be increased. For example, more planning is needed to decide how to divide the project and more administrative works are needed to support work processes.

b) It is likely that some sub-projects have dependencies over the others. Hence, project managers need to think ahead of the mitigation plan if the prior project doesn't meet the satisfaction in terms of quality or time frame. Moreover, since each project is on its own, but they are correlated in the roots, the communication between each project is critical. Project managers need to carefully manage the communication between related projects so fewer problem may happen in the final integration stage.

2) Scrum - Agile

Another Software Development Lifecycle model for this project is the Scrum model, in which the project is developed in incremental, rapid cycles. In each cycle, the developing team targets a specific goal and 'sprint' to it. Meanwhile, the developing team is open to customer feedback and scope changes. In this way, the clients are satisfied because the project outcome can be seen in the early stage (once one feature has been developed and tested), and changes are allowed to happen if clients have new or different ideas, and the developing team usually pays greater attention to good designing and technical excellence [4]. It suits this project because compared to other factors, Resilienza focuses more on the quality of the product as the major selling feature of its service is high quality.

Pros:

a) The most important advantage of Scrum model is that clients and the 'ConquerIT' developing team can constantly communicate and interact with each other. Developing team can ask for instant feedback from clients and modify its current approach on software developing. In this way, any changes to the original scope can be dealt with in the current release of project.

For example, one important feature of 'ConquerIT' is that customer engineers could pre-make continuity plan packages for customers to choose. Now Resilienza wish to change this idea in which engineers can add new packages instantly instead of periodically.

In a formal model, if this situation happens during the middle of the development, it can be problematic since the goal has already been sat and the developing team has no plan for changes. It can be quite risky if this change is not managed well.

However, in the scrum model since the developing team constantly communicate with clients and set the sprint goal frequently. If scope changes are needed, the product owner can modify or reset the required features of software and scrum master can ensure the team is working on the correct direction. This might add additional time cost during development; however, it can improve the possibility that clients get what they expected.

b) Another advantage of scrum model is that clients could have early access to operational and usable software. This feature is essential to a project like 'ConquerIT' as it changes the business mode of Resilienza largely. It would quite beneficial if Resilienza could have it tested among small group of customers and gain feedback from them. If the demonstrate of software is promising, Resilienza could gain direct benefits from it. Otherwise, Resilienza could analysis how to improve it in the early stages.

Cons:

a) In a large project like 'ConquerIT', it could be difficult to foresee the effort required to accomplish this project at the beginning. For example, the develop team soon finds out the work need to be carried out is much heavier than planned due to constant change of scopes and request more resources to finish the project. Hence some movement availability must be made in the contract. This could cause disagreement between client and develop team and lead to dispute.

b) Clients need to contribute much more during the development of this project, compared to a formal development model. Constant communication is a must in scrum model, otherwise development team cannot receive instant feedback from client, moving on to next phase. If clients provide feedback to the previous phase's work, development team need to come back to it and a lot efforts are being wasted in this process.

Bibliography

[1] A. Mishra and D. Dubey, "A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios," 5 October 2013. [Online]. Available: https://www.researchgate.net/profile/Apoorva-Mishra/publication/289526047_A_Comparative_Study_of_Different_Software_Development_Life_Cycle_Models_in_Different_Scenarios/links/59c00417458515e9cfd549a1/A-Comparative-Study-of-Different-Software-Development-L. [Accessed 26 August 2022].

[2] H. Hajjdiab, "Adopting Agile Software Development: Issues and Challenges," September 2011. [Online]. Available: https://www.researchgate.net/publication/276198893_Adopting_Agile_Software_Development_Issues_and_Challenges. [Accessed 26 Auguest 2022].

[3] K. Petersen and C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices," 10 July 2010. [Online]. Available: https://link.springer.com/article/10.1007/s10664-010-9136-6#citeas. [Accessed 26 August 2022].

[4] D. Reifer, "Ten deadly risks in Internet and intranet software development," 7 August 2002. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/991324. [Accessed 26 August 2022].