FarmX Project Case Study and Analysis

Background

This project is funded by Australian government and it got $180,000 grant. The purpose of the object is developing a platform act as both farming diary and social media so that farmers get unified which lead to more negotiation bargain and achieve better economic gains. The main functions of the software include creating and maintaining user profiles, multiple source of diary entries both manual entries and syncing from other devices and services, acting as social media where users can connect with each other, leave comments and organize events. Social media function is the main point of differentiating with other farming software and create appealing. The software will benefit from earning handling fee for the transaction it processed.

Challenging characteristics:

Firstly, it's challenging to syncing with other devices and services since different services have different protocol interface to adapt, which are challenging to design all of them out.

Secondly, it's challenging to design maintain a database to contain all user profiles, entries and social interaction. Also, it's challenging to build a capable server to stay workable when numerous number of users visit the database and social interaction pages at the time.

Thirdly, prior plan on further enhancement is challenging. Though it is no need to add in transaction supporter right now, the software need to possess the flexibility for further extension and it need to be designed to be compatible with financial processing facilities. This requires a challenging layout design in advance.

Finally, in the social interaction section, there are platform administrators required to process policy violation report and oversee foul play. This requires a challenging design of a central controller which have higher authority and can visit user data when necessary.

Risk:

Risk that other farm-related software develops similar functions before the *farmX* is completed. The impact of this risk is huge because the earlier-launched can take up the market first.

Risk that other farm services do not allow syncing. Because the farming platforms mentioned in the case study are also software, which are in a competing position with *farmX*, and they may hold the information to their own platform to maintain comparative advantage. The impact is medium, because the software is still usable even some data cannot be synced in. Manually entering the data also works even it is not as convenient as syncing.

Risk that being difficult to accumulate users. Wilma and Barnaby see their key differentiator is social media aspect. However, social media has a strong aggregated effect which means users tend to be drawn to those which already owns massive users. Farmers and farm enthusiasts can stick to existing and aggregated social medias, such as Facebook, Twitter and Meetup. For example, farmers from the same trading block can share a Facebook discussion group and use Meetup for gathering, instead of installing one more App on their phones. It's risky to get enough early adopter until the software is persuasive to act as a social media platform. To minimize this risk, *farmX* should pay effort to appeal and market to targeted farmers and farmer organizations to get its earlier adopters

and form a community at first so that it can attract people's attention gradually and naturally. The impact of this risk is big, because if not enough is using it, there is no way to draw in revenue.

Risk that many designed functions are not useful for targeted users. Many measurements and entries require devices and kits, for example wireless weather stations, soil test kits, and fitness tracking wrist bands. Many farmers may do not have them and decide not to use them. If they decide not measuring those data is the cost-efficient and time-efficient way to do the farming, they will think the diary entries idea is a waste of money and time. If few people are keeping diary, the *farmX* gets less user input and engagement. Successfully using the platform need certain level of expertise – 'intelligently monitor the national grain price fluctuation and analyse the market'. Some users may do not have the expertise to analyse national market or they already have stable distributors that they do not need to pay close eyes on national market. To minimize this risk, the *farmX* trial can start with industrialized farming areas where farming infrastructures are welly built and advance to less technologized area. This risk is medium, because if core functions are not regarded as useful, users may shift to other software.

Initialization

Initiation activities include analysing case study (business needs), analysing constraints (scope, time, cost), develop business case (cost versus benefit) and develop project charter (stakeholder analysis). (H.Drakos, 2018) More detailed initialization analysis is shown in *Figure1* below.

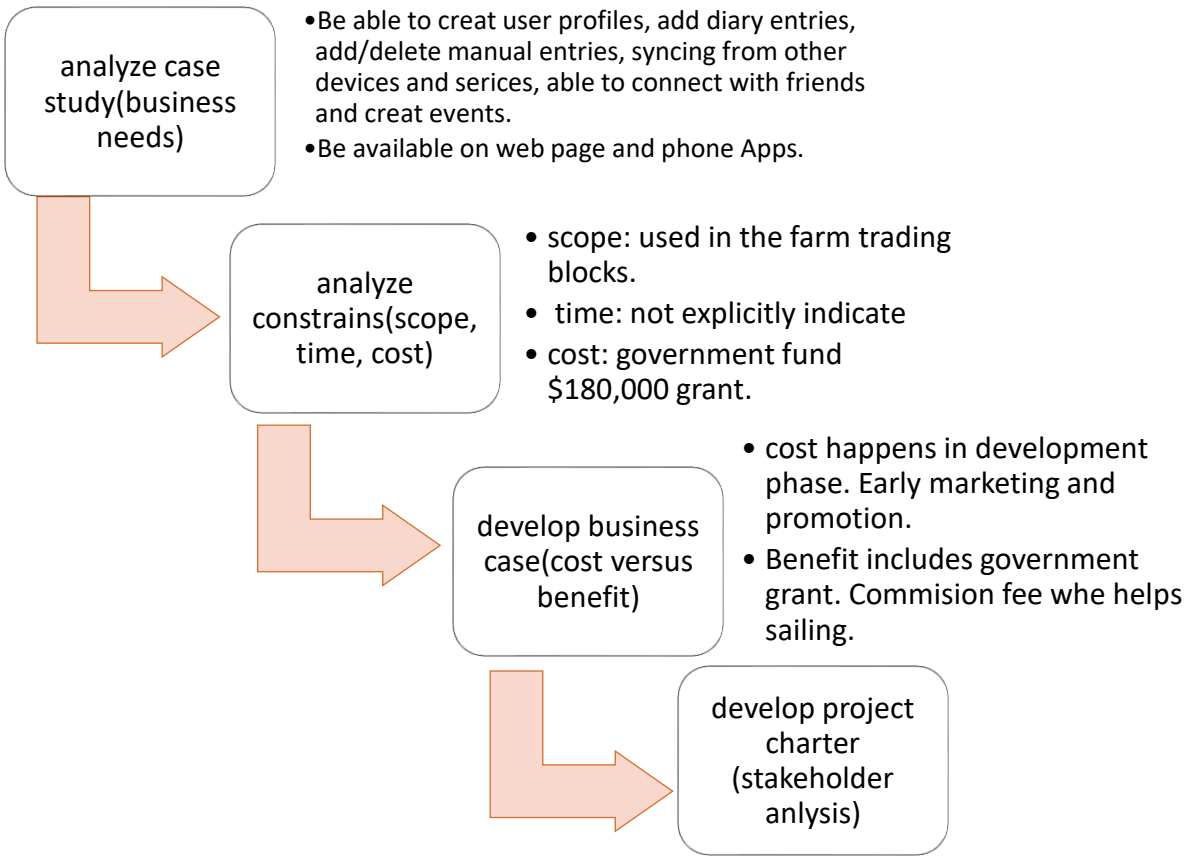

Figure 1 Initialization Activities

Project Charter

	Item	Quantity	Rate	Total
C O S T	Software developer	2	$2000	$4000
	Marketing	2	$1500	$3000
	Equipment	5	$1000	$5000
	total			

Table 1 Business Cost Analysis

	Item	Quantity	Rate	Total
B E N I F I T	Handling Fee	$20000	2%	$400
	Service Fee			
	……			

Table 2 Business Benefit Analysis

Stakeholder

The prioritized stakeholders are farmers and farming communities who potentially need a platform to record farming activities, weather, soil, market and other farming-related data and use it to connect with other farmers in the same agricultural blocks or community. Other stakeholders include sellers and distributors of farm products, consultants, researchers and policy-makers on agricultural market, and farming kits and devices company. They can gather user-end information and marketing farming tools on the platform.

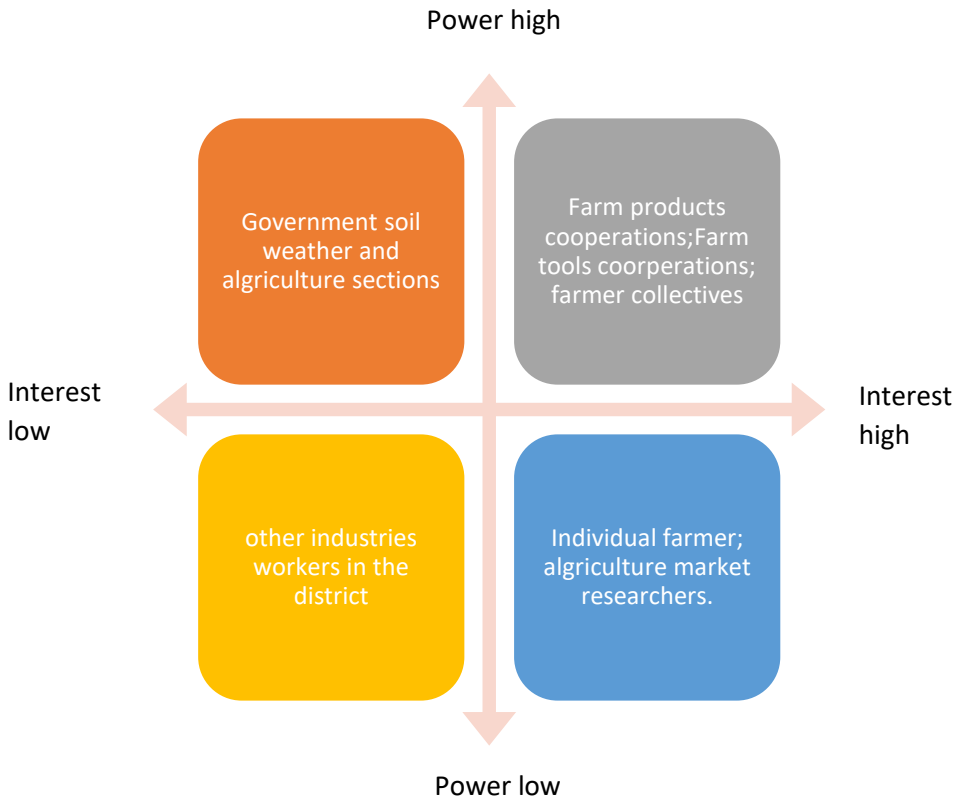

Figure 2 Stakeholder Analysis of farmX

SDLC models

Waterfall model

Waterfall model suits the projects where requirements are very well understood and stay stable along the development process. In the case study, the requirements are clearly defined and described, for example how members register and fill in profile, what kind of entries are recorded, what third party source sync from, and what social media function are applied (approve friend requests, send comments, …) so we can do it in waterfall model. Though the disadvantage is that if the requirements change during the development or any risk happens, it's difficult to manage the change. However, because it is targeted at farming industry which does not experience rapid and abrupt change. It is a stable and long-lasting industry which processes stable and clear business needs and requirements, so the disadvantage is minor and unlikely to experience huge changes during the development. Despite more functions will be introduced in "future enhancement", the requirement is still clearly laid one – 'eBay style' transaction and commission fee. It is still plannable and not counted as unpredictable change.

Secondly, the sequential process in the waterfall model also suits in this case because it does not have a large team to divide the tasks. If it is just a team of one or two, it is natural to complete the tasks sequentially, one after another.

Another advantage of waterfall model is that its extensive and thorough planning leads to more accurate timeline and budgets. I think it is important in this case because the $180,000 grant from government is a fixed number. If the costs exceed the budget, it is hard to see other source of revenue to cover the exceeding part and it is disastrous if there is not enough money to finish the software at all, so an accurate budget is a vital advantage.

On the other hand, waterfall model perform poorly on risk management since it does not has the flexibility to adjust when unanticipated risk happens. For example, as mentioned earlier in the risk section, if other App launches similar function earlier. Development under waterfall model does not revisit the plan phase (R.Betchtold, 1999).

Incremental Model

In incremental model, the whole requirement is divided into various releases (H.Drakos, 2018), customers can get important functionality early. In the case, the diary entry function can be handed over to customer first, and then develop the social media part, so that customer can run a trial on diary entry earlier. Also, because the requirement is divided into smaller modules, customer can respond to each build. If customer gets diary entry first, they can give feedback and new ideas on the development, such as adding new functions in social media section so that it relates closer to diary or syncing from more third parties' sources which help to keep the diary complete and convenient. If customers want the initial delivery faster, incremental model is also a good idea since the whole project is divided into modules. If customers want one part more urgent than the whole project, this is the way.

Incremental model also has its share of disadvantages. It need more resources and management attention, which can lead to increase in cost, for example, hiring more management personnel and developers. In the *farmX*, we do not have a large team and budget purely comes from government grant, so the budget is rigid and management attention is limited.

Another disadvantage is that partitioning the increments are difficult, because different sections and functions are inner-connected and closely related. For example, the entries of diary are the feed on social media page, the data syncing from other sources are the content of diary entries, user profiles are updated with their social media actions, and vice versa. All these connections make defining smaller modules difficult and confusing.

Besides, problems may occur at the time of final integration, which means that earlier divided modules have compatibility difficulty. In *farmX,* there is a lack of experienced developers to oversee the integration issue. If no one has an experience in integration stage, problem may occur.

Angile Scrum

Agile focuses on adapting changing situations (H.Drakos, 2018). For example, if Wilma and Barnaby come up with new idea on the requirements of the software, or similar software launched which makes part of the planned function pointless, Agile model provides a way to adjust plan on the updated feedbacks. In this way, Agile model always perform better in terms of risk management and change-handling.

Another advantage of Agile model is that it facilitates better communication between customers and developers. Customers get more updates during the process of development, and developers get more timely inform on customers' demand and need. Customers are focused over formalised sign-offs (H.Drakos, 2018), and they can update their requirements and needs along the development process. Besides, involvement and engagement across the team is also improved since

However, Agile model is difficult to be done without an experienced Scrum Master. From what the case study presents, the team does not have member with Scrum Master expertise. Recruiting Scrum Master will increase cost and risk, for example the Scrum Master quits in the middle of the development.

The choice of model

As advantages and disadvantages discussed above, I believe the waterfall model should be applied.

References

H.Drakos. (2018). *Software Processes and Management SWEN90016.* University of Melbourne.

R.Betchtold. (1999). *Essentials of Software Project Management.* Management Concepts Inc.