Chapter 2

LITERATURE SURVEY

1. Software Risk Management. In: Institute of Electrical and Electronics Engineers (IEEE), 2011.

Modern software development faces many challenges nowadays. In order to avoiding, minimizing, and monitoring the risks and their impact, it is important to identify, estimate and evaluate the risks.

Risk management should begin at the earliest stages of program planning and continue throughout the total life-cycle of the program. Additionally, risk management is most effective if it is fully integrated with the program's systems engineering and program management processes—as a driver and a dependency on those processes for root cause and consequence management.

In software risk management, there is a wide range of supporting data and processes to help integrate and balance constraints against risk. The objective of a well-managed risk management program is to provide a repeatable process for balancing cost, schedule, and performance goals within program funding, especially on programs with designs that approach or exceed the state-of-the-art or have tightly constrained or optimistic cost, schedule, and performance goals.

An effective risk management process requires a commitment on the part of the project manager, the project team, and the contractor to be successful. When properly resourced and implemented, the risk management process supports setting and achieving realistic cost, schedule, and performance objectives and provides early identification of risks for special attention and mitigation.

The risk management process methodology involves five (5) basic steps:

1. Identify the risks - Understand the typical problems that might adversely affect the project.
2. Assess the risks - Rank the risks in order of importance based on probability of occurrence, impact of occurrence, and degree of risk certainty.
3. Plan the risk response – Analyze risk assessment alternatives and modify the project plan to adjust for the risk.
4. Monitor the risks – Throughout the project, continue to revisit the risk profile, re-evaluate major risks, and update the risk profile with action taken.
5. Document lessons learned – Learn from the risk identification, assessment, and management process.

Taking in the account the specifics of the IT area and the background in the related software fields the author suggests the following key principles in the software risk management:

1. To avoid large IT projects, that mean to divide large software development projects in to a certain stages with milestones and defined results and a person responsible for the success of the entire deliverable.
2. Attract and include IT specialists with background and experience for the IT project management instead of technical staff to focus on the quality of project management. Use helicopter view and strategic thinking of the independent subject matter experts that can look at the project from their own perspective and independently assess the state of the project health.
3. Consider all range of the risks, not only the risks that had occurred once in the previous project or available in the check lists, risks databases and other sources.

After the risks are identified and assessed they should be mitigated with one of the response actions based on the risk type and priority. The types of responses can vary depending on the chosen methodology, but the main four types of responses are:

1. Mitigate the Risk – incorporate specific plans into the project scope to deal with the occurrence of, or to minimize the likelihood of, the risk occurring.
2. Avoid the Risk – remove scope that includes risk from the project.
3. Share the Risk – transfer ownership of scope to another party so they now have risk.
4. Accept the Risk – do nothing, run the chance of the risk occurring, deal with it if it does.

2. An Approach using Agile Method for Software Development on 2016 1st International Conference on Innovation and Challenges in Cyber Security (ICICCS 2016).

The aim of this paper is to organize, analyse and make sense out of the dispersed field of agile software development methods. Based on the result of the analysis, practitioners are in a better position to understand the various properties of each method and make their judgment in a more informed way.

For the purposes of this paper, agile software development in general is characterized by the following attributes: incremental, cooperative, straightforward, and adaptive. Due to limited space, readers are referred to for further discussion on other possible characterizations. In the following, each method's objectives are briefly introduced.

Adaptive software development: Adaptive Software Development replaces the traditional waterfall cycle with a repeating series of *speculate*, *collaborate*, and *learn* cycles.
This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, time boxed, risk driven, and change tolerant. ASD claims to provide a framework with enough guidance to prevent projects from falling into chaos, but not too much, which could suppress emergence and creativity.

Agile modelling: Agile Modelling (AM) is a practice-based methodology for modelling and documentation of software based systems. It is intended to be a collection of values, principles, and practices for modelling software that can be applied on a software development project in a more flexible manner than traditional modelling methods. The aim is to keep the amount of models and documentation as low as possible. Cultural issues are addressed by depicting ways to encourage communication, and to organize team structures and ways of working.

Extreme programming: Extreme programming (XP) is a collection of well-known software engineering practices. XP aims at enabling successful software development despite vague or constantly changing software requirements. *Extreme Programming Explained* describes Extreme Programming as a software-development discipline that organizes people
to produce higher-quality software more productively. XP attempts to reduce the cost of changes in requirements by having multiple short development cycles, rather than a long one.

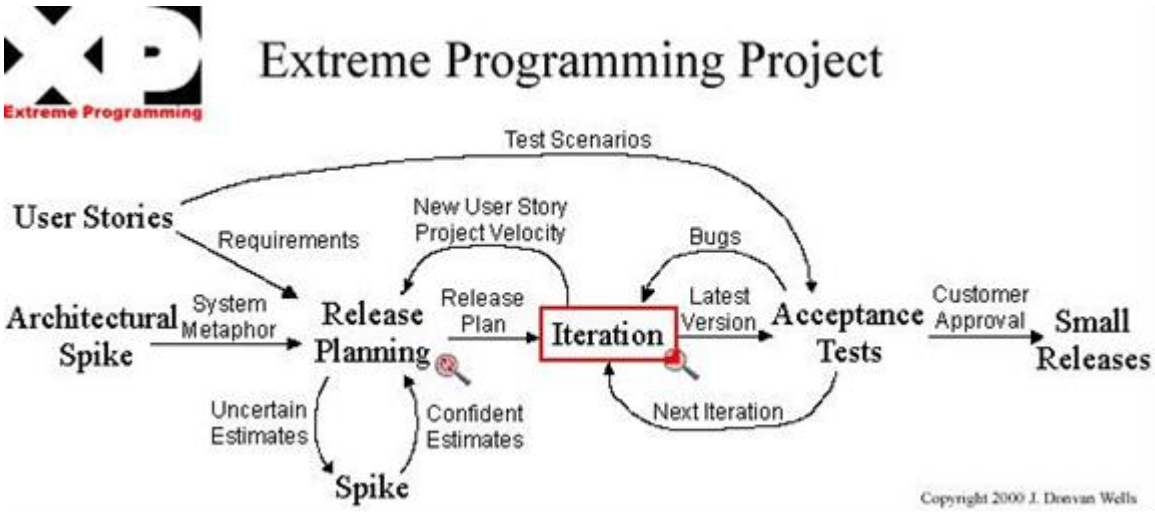

Figure 2.1: The flow in a project using XP

ASD METHODOLOGIES

There are several ASD methodologies available today. 125 out of 192 of this question's responses indicated they used the Scrum ASD methodology. Figure 1 shows the extent of adoption of different ASD methodologies.

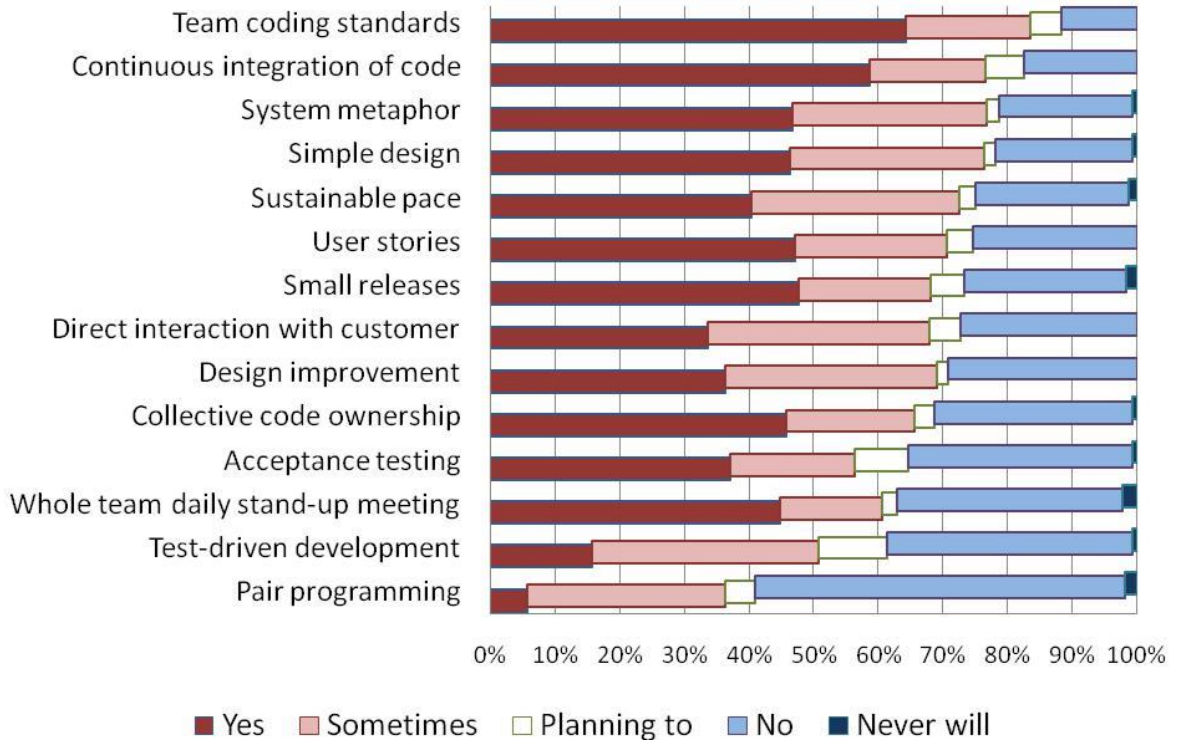

Figure 2.2: Percentage of Usage for Agile Practices

3. Automated Recharge of Prepaid Mobile Phones.

This paper presents a dynamic approach for online mobile phone recharge. Over the past several years, percentage of overall prepaid users has increased rapidly. However, the recharging process is still somehow manual. The system proposed in this paper shows an automated way to recharge the prepaid account. This approach will have a great business impact both in local and global aspects. The effectiveness has been proved with a SMSC server and a recharge SIM.

For mobile phone users, generally two types of services prepaid and postpaid are offered by the mobile phone operators. Prepaid service is much popular than postpaid service among mass people. In case of prepaid service mobile phone users have to have a certain balance of money to use the service. Here comes the concept of mobile phone recharge.

An automated system is implemented to recharge prepaid mobile in online. A recharge center shopkeeper as well as a normal customer can upload customer mobile number, recharge amount in web page by using his own valid username and password. Database server stored

the uploaded information. A file watcher always checks for a new request at the server. As soon as a new request is available for recharge, parser will parse the information to retrieve the desired customer number and valid amount. A message server (SMSC) also runs on the system for sending message from database to mobile phone operator's server. A MODEM with a recharge SIM is connected with the server to fulfill the desired communication. The message server (SMSC) will store the requested message and generate an xml file for the corresponding request.

Work Flow of Recharge Period

In recharge period message server is run on the PC. Two programs are simultaneously run also. A file watcher and another is parser. File watcher always watch the message inbox of the server for notifying if any new message come. When it got a message then sends it to the parser for parsing. After parsing there found a mobile number, recharge amount and customer name from the message which correspond to the customer. Then a query will generate to retrieve the whole information of the customer correspond to report from the customer database. Then he/she will get a SMS which belongs to an acknowledgement of receiving his/her request. Otherwise he/she will get an Acknowledgement of not satisfying the recharge requirement. The information of the customers who are able to recharge then stored in a table including the cell number, name and date of recharge.

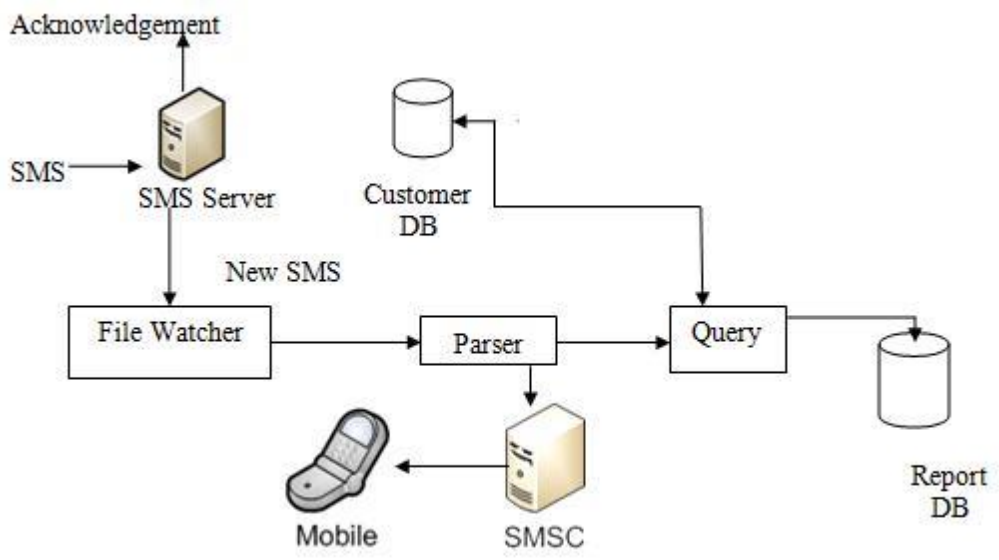

Figure 2.3: Flow diagram of the recharge period