OULU UNIVERSITY OF
APPLIED SCIENCES

Atul Mahajan

RISKS AND BENEFITS OF USING SINGLE SUPPLIER IN SOFTWARE DEVELOPMENT

RISKS AND BENEFITS OF USING SINGLE SUPPLIER IN SOFTWARE DEVELOPMENT

Atul Mahajan
Thesis
Autumn 2018
Masters in Industrial Management
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
“Masters in Industrial Management”

Author(s): Atul Mahajan
Title of Master´s thesis: Risks and Benefits of Using Single Supplier in Software Development
Supervisor(s): Hannu Päätalo
Term and year of completion: Autumn 2018 Number of pages: 85 + 5

For many organizations involved in product development where software itself is not a main product, software development needs to be outsourced. Having software development with only internal resource utilization is not a viable option. A good balance of internal resourcing and outsourcing is needed to ensure that development continues as desired. At the same time, the company needs to be self-reliant in cases of crisis with one of the suppliers. In that context, a good mix of suppliers is also desirable.

The aim of this thesis is to study the benefits and risks an organization can face due to single supplier relationship in software development. The focus of the study is on recommending possible solutions to mitigate the identified risks. This is done by considering the importance of relationship with existing supplier, support needed for ongoing activities/development, and safeguard business interests for future.

The study has utilized the Research Methods Map detailed on Jyväskylä University’s web portal “Koppa” (Jyväskylä University. Research Methods Maps). The thesis utilizes multi-method research utilizing research methods such as Qualitative Research for understanding the topic and the background as well as collecting data. In addition, expert knowledge and Case Study approach are used to specifically dive into the details of the topic of this thesis.

The thesis has identified the risks of having a single supplier for software development with respect to various software development methodologies and models used by the customer organization. The impact of different contractual models on the supplier relationship has also been studied. The thesis has identified certain risks and possible solutions to mitigate the risks. The solutions identified will be utilized by the organization used in the case study of this thesis.

The new software development models such as Scaled Agile Framework (SAFe) or DevOps have not been studied into much details. How those development models affect the supplier relationship needs to be studied further.

Keywords:

software development, single supplier risk, supplier management

PREFACE

I decided to do this as a part of broader activity to study the current software development process and risks associated with it for a target organization. The organization develops the automation systems for the products of its parent organization. Based on the current scenario in the target organization, the focus of this thesis has been on supplier involvement in the development activities and to study the risks and benefits of using single supplier in software development.

I would like to thank the members of the target organization used as a case study here for their open and valuable inputs as well as constant support. The members of development team provided valuable insights into practical and technical challenges they face. The members of sourcing or procurement department helped me by providing material and reports which were used to understand the practices and challenges faced in the industry regarding outsourcing. Due to confidentiality reasons, the person names and reports are not mentioned in this thesis.

I also thank the members of the supplier organization who provided valuable insights into their challenges and thought process.

I would also like to express my deep appreciation and thanks to my instructor, Mr. Hannu Päätalo, Principal Lecturer, at Oulu University of Applied Sciences for supporting and guiding me constantly in getting this thesis done.

Special compliments go to my loving family and friends for helping, supporting and encouraging me in finalising my thesis.

Oulu 04.12.2018

Atul Mahajan

CONTENTS

TERMS AND ABBREVIATIONS

a.k.a.	also known as
API	Application Programming Interface
ART	Agile Release Train
BDFU	Big Design Up Front software development model
CALMR	Culture, Automation, Lean Flow, Measurement and Recovery, values of DevOps
CPAF	Cost-Plus-Award-Fee contract
CPFF	Cost-Plus-Fixed-Fee contract
CPIF	Cost-Plus-Incentive-Fee contract
CR	Change Request
DevOps	Delivery and Operations
DSDM	Dynamic Systems Development Method
ECU	Engine Control Unit
FDD	Feature Driven Development
FFP	Firm-Fixed-Price contract
FPAF	Fixed-Price-Award-Fee contract
FPIF	Fixed-Price-Incentive-Fee contract
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
HW	Hardware
IDE	Integrated Development Environment
ISO	International Standards Organization
MVP	Minimum Viable Product
OPM	Organizational Project Management Framework
OSI	ISO Open System Interconnection model
OSS	Open Source Software
PC	Personal Computer
PI	Program Increment
PMBOK	Project Management Body of Knowledge by PMI

PMI	Project Management Institute
PO	Project Order or Purchase Order
PSI	Potentially Shippable Increment
RAD	Rapid Application Development model
RFP	Request for Proposal
RFQ	Request for Quotation
RS-485	Serial Communication Protocol Standard
R&D	Research and Development
SaaS	Software-as-a-Service
SAFe	Scaled Agile Framework
SDLC	Software Development Lifecycle
SW	Software
TCP	Transport Layer Protocol
TDD	Test Driven Development
T&M	Time and Material contract
UI	User Interface
USB	Universal Serial Bus
XP	Extreme Programming

1 INTRODUCTION

Nowadays, most of industrial products are designed using high degree of automation. Quite often this automation involves the software in addition to electronic/electrical and mechanical components. While some of the product development is outsourced or done in collaboration with other industry partners, there is always an area of so-called “core competence”. The organizations always prefer to maintain core competence internally as much as possible to be in complete control of the product development. With rapid technological development, this becomes a challenge and slow and steady outsourcing could result in mid-long term situation where outsourcing becomes more significant over the internal competence.

This thesis studies such a case of a department which provides the complete electronic automation system for managing parent organization’s core product portfolio. The automation system includes hardware, embedded software that works with the hardware and the PC-based management software to configure, monitor and troubleshoot the embedded automation system. Embedded automation software development and maintenance done by the department is rather big and complex task and involves multiple teams.

The department uses the constant outsourcing of automation software development as having all needed resources internally is not a viable option. At present, there is only one supplier involved in development of so-called “platform software”, which serves a common basis for all automation components. That is a risk where development would slow down in case of challenging situation with the supplier. To make the situation even worse, the ratio of outsourcing as compared to utilization of internal resources is much higher. This will have disastrous impact in case supplier gets into challenging situation or acquired by another company (especially competitors) or cooperation starts to deteriorate for some reason.

The aim of this thesis is to study the benefits and risks the organization is having due to single supplier relationship. The focus of the study is to recommend possible solutions to mitigate the identified risks. This needs to be done by considering the importance of relationship with current supplier, support needed for ongoing activities/development, and safeguard business interests for future.

2 APPROACH

The study has utilized the Research Methods Map detailed on Jyväskylä University's web portal "Koppa" (Jyväskylä University. Research Methods Maps).

The thesis utilizes multi-method research using following research methods –

- Qualitative Research is used for understanding the topic and the background as well as collecting data for possible solutions evaluations etc.

- Case study approach is used to specifically dive into the very specific topic of this thesis. Note that the details of the case used to study of this thesis cannot be disclosed due to the sensitivity of some of the information and the agreement reached with the organization providing the research case.

The Qualitative and Network analysis techniques have been used for data collection and data analysis as part of the research to infer the conclusions and the recommendations for automation platform software development. Some of the techniques used are listed below:

- Analysis of needs and requirements for automation SW platform development with respect to the organizational culture, structure, processes and constraints
- Development methods and practices used or studied in past and the lessons learnt
- Wider study of best practices by researching the content of books and internet knowledge base
- Interviews with selected members of R&D unit and developer partners involved in the automation software platform development
- Using LinkedIn discussion groups
- Input collections from subject matter experts

The Author himself has a hands-on experience of several years in software development and dealing with the challenges of the topic discussed here. A lot of findings from present responsibilities at the time of developing this thesis are also taken into account.

The thesis provides an overview about an automation system and the architecture of embedded software to start with. It also provides an overview about the main component of software architecture called “platform software”. The challenges related to development of this component using outsourcing has been used as a main focus point of this thesis. The thesis then moves on to discuss multiple software development models as well as contracting models to outsource the development work. The intention has been to study the relationship between software development models and contractual models to find out the benefits or challenges of particular relationship. Such an analysis was made to evaluate whether the certain development model would work best for outsourcing development work when certain contractual model is used and vice-versa.

After evaluation of software model and outsourcing contractual models, the further attention has been given to the software architecture and the development environment/infrastructure as well as team composition. The intention with such an analysis is to find out whether certain software architecture and the development infrastructure would suit better for effective outsourcing.

The output from the above analysis has been used to formulate the possible solutions or risk mitigation for addressing reliance on single supplier for outsourcing development work.

3 CASE STUDY - BACKGROUND

In this case study, the organization under case study will be addressed hereafter as “customer”. As the customer is involved in automation system development for its heavy industry product, here we take a look at an overview of automation system components and the architecture of the embedded software to understand which embedded software part development uses outsourcing.

3.1 Typical Automation System in heavy industries

Most heavy industrial products such as cranes, lifts, turbines, power generators etc. include heavy automation. This automation typically consists of one or more control modules, displays, control panels and several types of sensors and actuators etc. as depicted below in FIGURE 1.

FIGURE 1. Engine Automation System Components

Most of the industrial products needing automation depend on the software (hereafter “SW”) in addition to simple electrical or electronic components or hardware (hereafter “HW”). Below the role of SW in each of the automation system components is described in brief.

3.1.1 Management System

The management system typically is a separate SW with Graphical User Interface (Graphical User Interface, Cited 28.04.2017) commonly termed as GUI. It provides Human Machine Interface (Human Machine Interface, Cited 28.04.2017) commonly termed as HMI towards the product. The management SW utilizes one or more of several communication interfaces towards the product’s embedded automation system (Embedded System, Cited 28.04.2017) and embedded SW (Embedded Software, Cited 28.04.2017) such as RS-485 serial interface, Modbus/TCP, USB and Ethernet based protocols etc. The typical Management System allows users of the product to monitor the behaviour of the product during its operational status, control its behaviour and put the product in operational mode or out of operational mode (start/stop functionality). Most Management Systems allow also the alarm/fault monitoring as well as diagnostics and troubleshooting functionality to address erroneous situations.

3.1.2 Product Validation System

Like any manufactured product, the heavy industry products also need to be tested. It is quite common to use so called “Test rigs” in the validation. The test rigs can use the automation system components against the emulated version of product. The benefits behind such test rigs are many-fold:

- Speed up the validation process
- Investment cost reduction as Test rigs are usually much cheaper compared to actual product
- Damages to actual product are restricted during validation process

The other testing systems can range from basic less complex testing equipment to very complex high-end testing equipment. Most validation systems for heavy industry products are high-tech systems with complex piece of SW including GUI as well as embedded SW.

3.1.3 Control Modules

Every control module in heavy industry products typically have stringent environmental requirements regarding tolerances for vibrations, operability for extreme temperatures, ruggedness etc. Each control module has embedded SW that is intended to serve the product in certain way. Example: Engine Control Unit (ECU) in automobiles carries out certain tasks regarding combustion control of the engine by utilizing inputs from various sensors and controlling the actuators etc.

Typically, these control modules also will have communication interfaces. They are either used for internal communication within the automation system needed for the product or for interfaces towards external systems or for both internal and external communication.

3.1.4 Industrial Display and Control Panels

Industrial Display contains GUI through which user can interact or work with the product. While some of the displays are used as monitors meant only for viewing the information, others can provide means to control the product behaviour. The industrial display can vary in designs. It can be a read-only monitor, or unit with combination of screen and push buttons to address specific needs of product, or even a full-blown touch interface. The SW complexity also varies based on functionalities supported by such displays.

Control panels usually provide visual indicators and means to carry out emergency or safety operations such as push button for Emergency Stop function etc. But advanced control panels can house also special control modules and displays to support much larger range of activities.

3.1.5 Sensors and Actuators

Sensors and actuators are typically simple components. Examples of these sensors could be temperature sensors, pressure sensors, piezo-electric sensors, proximity sensors, speed and position measurement sensors etc. Most sensors do not have any SW or very little SW. The sensors with SW typically fall under “Smart Sensors” category and usage of such sensors in heavy industry products is currently limited. With rapid development related to IoT however, there is a high probability that Smart Sensors would replace the basic sensors in long-term.

3.2 Software Architecture

The SW architecture and design can vary a lot. For most management systems, the standard hardware such as PC, connectivity interfaces such as Ethernet, Wi-Fi are used. The SW architecture and design choices depend upon approaches used whether the system is designed as simple native application running on PC, Web based system, client-server architecture etc. For embedded SW, however, the SW needs to be optimized for certain HW. Although different SW designs can be used in embedded SW, the high-level architecture typically remains the same.

FIGURE 2 below is an example of embedded systems architecture as depicted by Tammy Noregaard (T. Noregaard 2005, 57).

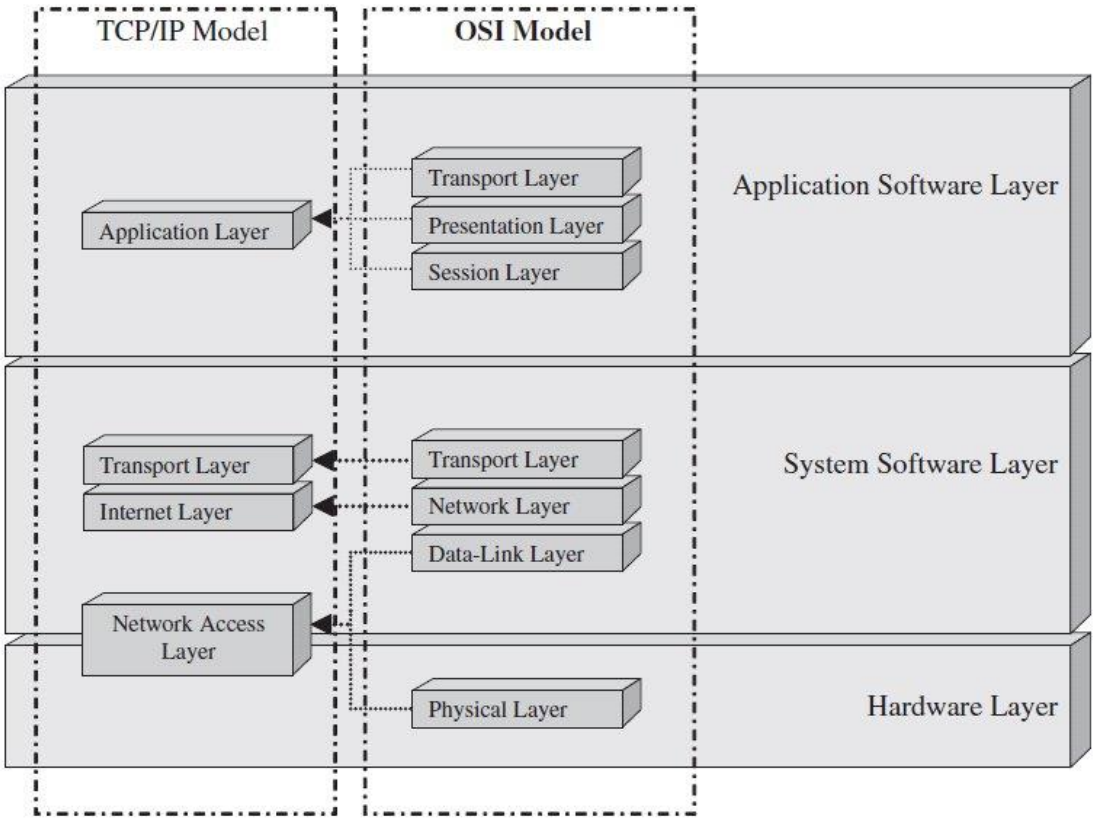

FIGURE 2. TCP/IP, OSI models and Embedded Systems Model block diagram (T. Noregaard 2005, 57)

The SW architecture in the customer case study is illustrated below in FIGURE 3.

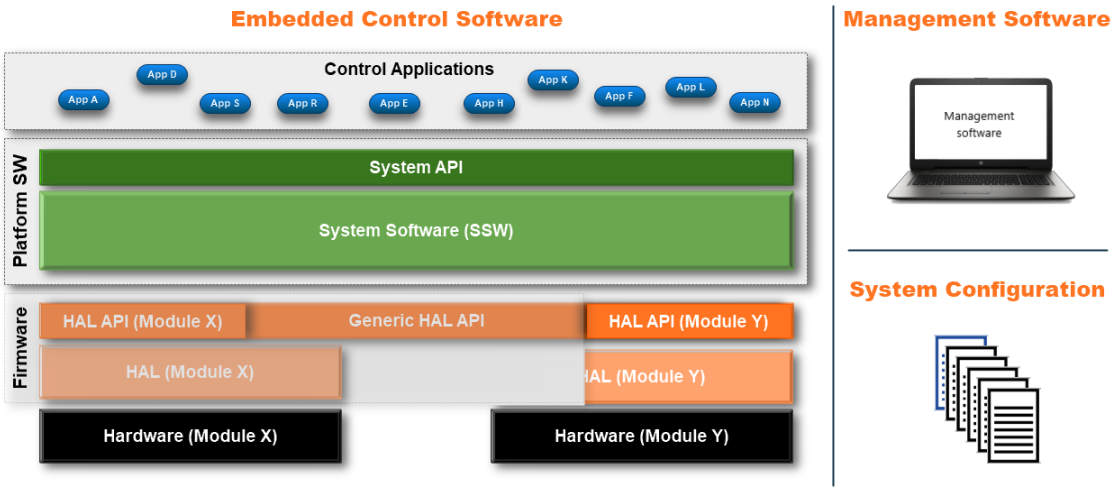

FIGURE 3. Customer Embedded Software Architecture

In this case, overall product SW can be further broken down into Firmware (Firmware, Cited 06.05.2017) including Hardware Abstraction Layer or HAL (Hardware Abstraction, Cited 06.05.2017), System SW, and Application SW. Firmware is low level SW and tightly coupled to the control module or display HW. System SW is sort of a middleware between low level SW and higher level Applications. Applications provide functionality and features for automation system whereas System SW acts as an enabler for those applications to work with the underlying HW. The Application Programming Interface or API (Application Programming Interface, Cited 05.05.2017), typically a set of APIs isolates HAL from middleware and is typically called as HAL API. Similarly, another set of APIs which isolates middleware from applications is called System SW API. The APIs not only ensure that integrity of lower levels in the SW architecture is protected, but also provide easier access and common structured approach to develop the higher-level SW layers. So, the development is typically bottom-up. With the matured system SW, unless HW is changed in a big way, the development process shifts slightly towards top-down approach as new functionality development or applications can require additional APIs or implementation in lower levels.

In this thesis, based on the case study, the system SW or middleware will be hereafter termed as “Platform SW”.

3.3 Platform SW Lifecycle

Platform SW shall follow the similar lifecycle model as the HW and the automation system. Typically one generation of automation system needs to be supported for 28-30 years. That is rather long duration. HW and automation system are managed with an obsolescence management plan based on the forecasting of components availability. However, the continuous development of SW makes this more challenging. The different generations of automation system are still expected to be supported in a common way, possibly using latest version of SW. So, backward compatibility and support is essential. For platform SW being the

middleware, the compatibility requirements are stringent. The platform SW needs to tackle support for underlying HW and at the same time provide interfaces to control applications on top. The architecture demands that control applications shall work without any modifications (except for new functionality) across different versions of platform SW. That compatibility needs to be taken care by platform SW. In practice that puts lots of constraint regarding how the SW is developed. Adapting new development models and development environments can be challenging. One of the major challenges for the customer is developing criteria regarding how long a SW release needs to be supported and what actions are to be taken when the support is discontinued.

3.4 Platform SW Development and Release Cycle

Platform SW development happens continuously. The different types of platform SW releases are used for different purposes.

3.4.1 Production ready maintenance releases

The existing products need to be supported by providing maintenance/bug-fix SW releases. Such SW releases can happen as both proactive or reactive efforts. The releases can be made proactively either for addressing certain defects found in internal testing or for providing important functional improvements identified as necessary updates for products in the field. In some situations, urgent SW releases need to be made to fix defects found in field installations making it reactive approach a.k.a. fire-fighting.

3.4.2 Production ready releases for new products

For new products and customer-specific or application-specific product deliveries, the new functionality needs to be implemented in platform SW. The production ready releases with such new functionality typically include also the

bug-fixes for existing defects found in platform SW. These releases are made after series of development releases which are tested to ensure that final production ready release meets the quality criteria.

3.4.3 Development releases

The platform SW releases are also needed to support the products under development or to provide new value-added functionalities on top of existing offering. This is needed to remain competitive in the target market. In some cases, it is crucial also to ensure that fixes implemented for certain defects have not caused any side effects. In order to achieve this, typically 2-3 intermediate development releases are made. Once the testing of such releases is successful and validation against the target product/s is success in adherence to defined quality criteria, the final production-ready release is made. These intermediate platform SW releases are termed or classified as proto or snapshot release, beta release, release candidate etc.

3.5 Development and Maintenance of Platform SW

The main parties involved in customer’s development process for its automation system are displayed below in FIGURE 4. Note that Customer here is the organization under this case study owning and developing the platform SW.

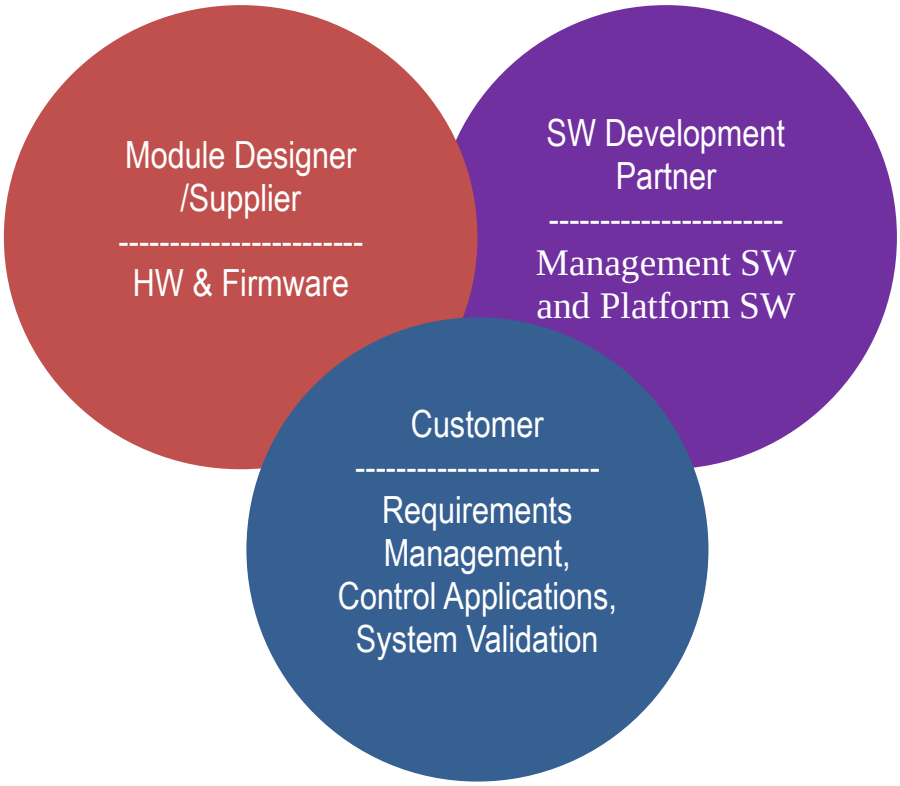

FIGURE 4. Development responsibilities distribution and dependencies in customer case study

The design of module HW and mechanics is responsibility of supplier responsible for module design or/and its manufacturing. The same is applicable also for module's SW development. The development and maintenance of firmware is responsibility of the same supplier. There are multiple suppliers providing different control modules.

The development of platform SW and management SW is outsourced to a SW development partner. This development partner needs to cooperate with suppliers responsible of module design. In addition, the close cooperation is needed with customer experts and developers working with control applications. Customer develops the control applications using own resources. Also, providing the use cases and requirements for development, evaluating architectural changes and prioritization of development activities is taken care by customer. The customer is managing the final validation of the product which includes also the testing of automation SW. So, while SW development partner is responsible for providing well-tested platform SW and module suppliers are responsible for

providing well-tested HW and firmware, the customer is taking care of testing the overall system including all these components and also the control applications.

The platform SW development is currently done using hybrid-agile development model. Customer is specifying the high level user stories and requirements to the development partner and the partner resources carry out actual design, implementation and testing for those. The development by partner resources is done using Scrum model. Refer to paragraph 4.2.5 for more details regarding Agile and Scrum.

The maintenance activities such as fixing the defects found in released platform SW etc.is done in cooperation between customer and development partner resources. The customer experts prioritise the backlog of defects according to business criticality and the development partner resources analyse and implement the fixes. The process used for this maintenance activity is Kanban which is also an Agile development model.

As the platform SW development is very crucial for customer and it is managed mainly by single partner/supplier, the risks involved are of very high impact for customer in case development partner gets into difficult financial situation or relationships get soar due to some issues etc. The risks if realised would certainly have a high impact to customer, especially considering that almost all of platform SW development is done by the partner with customer's own resources are mainly involved in defining requirements and prioritization of planned features and functionalities. This thesis focuses on those risks and how to address them proactively in order to enhance customer capabilities to safeguard vital SW lifecycle management for its products. Following chapters analyse different SW development models as well as SW procurement/ sourcing models to understand relationships between development models and contractual models used in procurement of SW development services. The analysis can provide the basis to formulate some of the solutions to be proposed to minimize the risks for the customer while taking into account also the supplier interests.

4 SOFTWARE DEVELOPMENT MODELS

The SW development methodologies and models have been changing over the time due to speed of overall technological advances. The necessity to bring the products as quickly as possible to markets has demanded faster ways to develop and deliver SW while maintaining the high standard of quality. Understanding of these models is necessary as some of the solutions proposed for the problem case study take certain development models into account. Per Stephens R., the development models can be largely categorised in two types: Predictive and Adaptive.

4.1 Predictive Development Model

The predictive development model assumes that the goals and means to achieve the goals are already known in advance and the execution is focused in getting things such as entire product design done based on the knowledge at the starting point. This is the reason why models under this category are also known as “Big Design Up Front (BDFU)” models (Stephens R. 2015, 270).

The examples of Predictive models are listed below:

- Waterfall Model
- V-Model
- Incremental Model

4.1.1 Waterfall Model

The Waterfall model is the first ever process model used for SW development. It was introduced by Royce (1970) as Software Development Lifecycle (SDLC) Model. It is simple to understand and use. The name waterfall originates from the fact that each of the development phase happens sequentially one after another

and none of the development phases are expected to overlap with each other. In real life, that is seldom the case.

FIGURE 5 below illustrates the Royce's original Waterfall model concept.

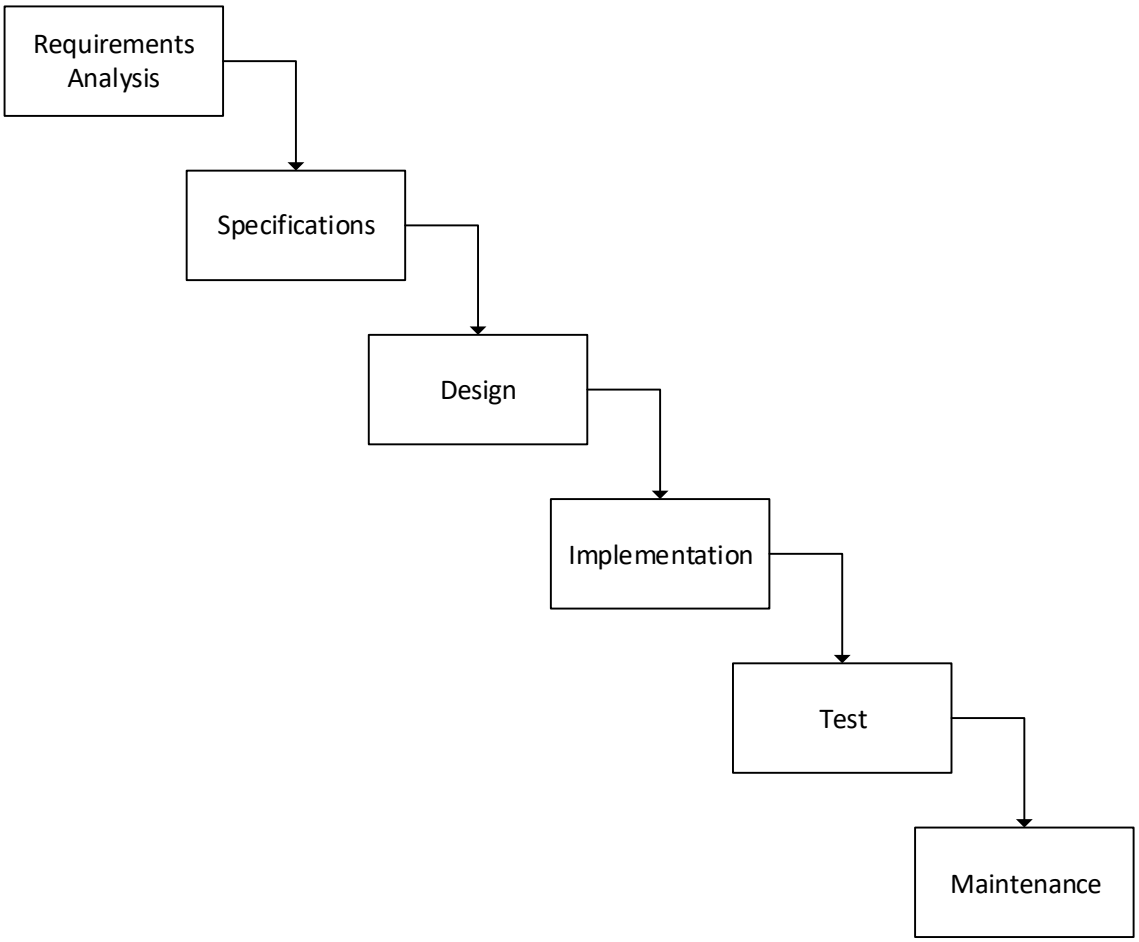

FIGURE 5. Initial Waterfall Model by Royce (Sage A. & Palmer J. 1990, 51)

The Waterfall model is good for SW development where the development is done with very clear product definition and set of requirements. Introducing change at later stage can have significant impact on schedule and costs as all previous phases need to be re-executed to take new change into account.

4.1.2 V-Model

The V-Model is based on Waterfall model. The main difference however is that during every development phase, a corresponding testing plan is also created. For example, when product definition is done based on business requirements, Acceptance Criteria or Acceptance Test Plan is developed which would later be used to test the developed SW product against the business requirements. The same goes for lower phases of the development. FIGURE 6 below depicts the V-Model.

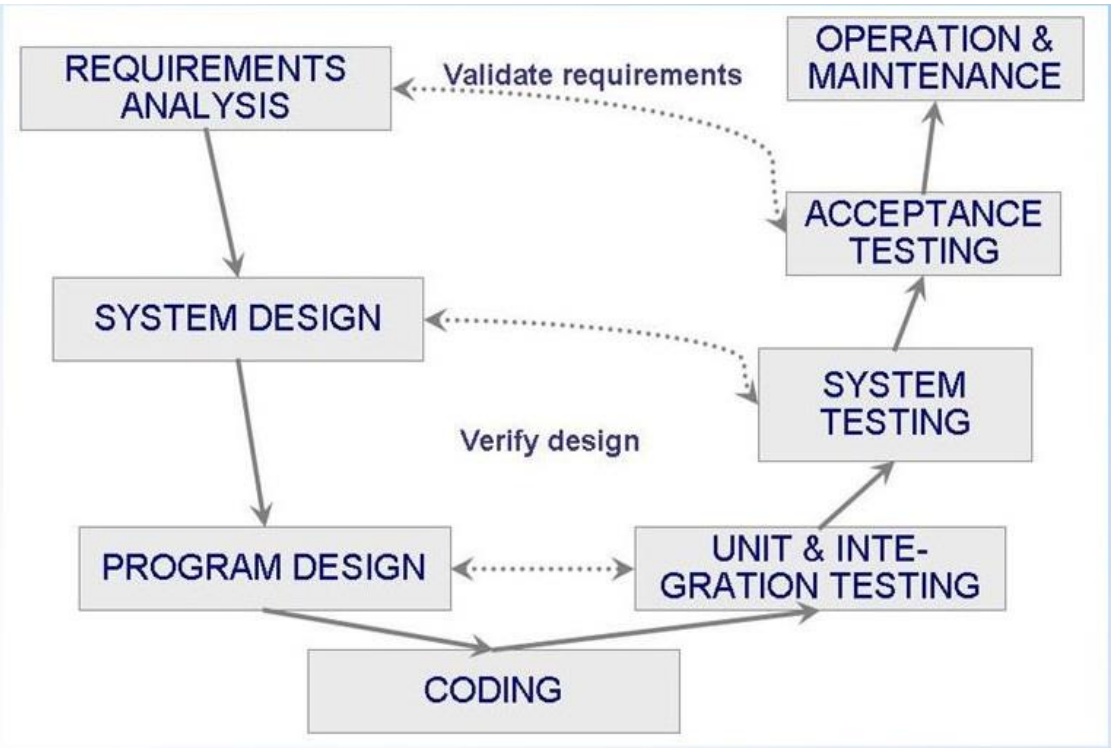

FIGURE 6. V-Model (Pfleeger, S. L. 2nd Edition, 53)

Similar to Waterfall model, V-Model is also good for development where product definition and requirements are well defined at start. The advantage compared to Waterfall model is that possible defects are verified at early phase instead of being detected only at later phase as in Waterfall model. Introducing late changes however has almost similar implications as in Waterfall model. Also, SW is

implemented almost as a last stage and hence it is not feasible to have any SW prototypes available to evaluate in earlier phases.

4.1.3 Incremental Model

Incremental model can be thought of set of multiple waterfall models. In Incremental model, an entire scope is split into multiple increments of development and each increment executes requirements gathering and analysis, design, implementation and testing in sequential manner as in Waterfall model. Typically, the first increment produces the basic version of SW product. The further increments keep adding more features and functionalities until the full scope of the SW product is realised. FIGURE 7 below illustrates the Incremental Model.

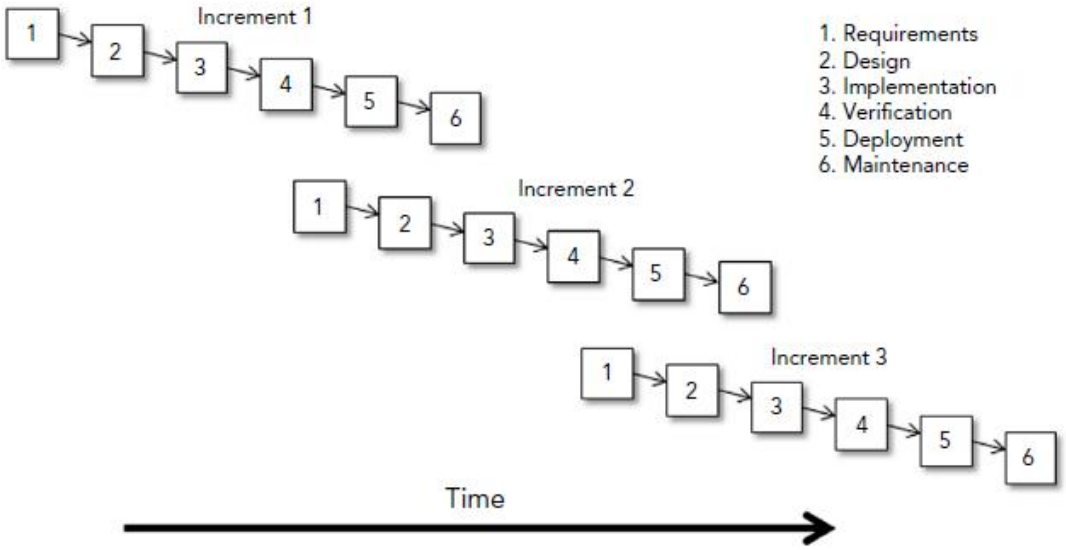

FIGURE 7. Incremental Model (Stephens R., 2015, 274)

Incremental model is highly suitable where the full scope of the intended SW product is well defined at the start. It can produce basic working SW at an early stage as compared to Waterfall or V-Model. Also, the requirements can be modified to some extent and be implemented in one of the following increments.

The negative side of this model is that it requires higher degree of planning to divide the scope properly in increments.

4.2 Adaptive Development Model

Unlike Predictive models, Adaptive model allows to change the end goals during the development process. The key principle of Adaptive model is to evaluate the end goals periodically against the outcome of development to plan and execute needed changes to reach the intended new goals. Often it is a case that customer requirements for the end-product are different or changed by the time the end-product is ready. Adaptive models cater to these changed requirements during end-end development process.

The examples of Adaptive models are listed below:

- Iterative model
- Prototype model
- Spiral model
- RAD model
- Agile development
- Continuous Integration, Continuous Delivery and Continuous Deployment
- DevOps
- SAFe

4.2.1 Iterative Model

In Iterative development model, the initial increment/s focus on developing the basic product with minimal implementation. The subsequent increments then focus on adding more and more improvements to complete or enhance the product. The concept has striking similarity with Incremental model. However, there is a subtle difference. In Iterative development for example, whole product is developed in initial iterations with very basic offering to build the foundation.

Subsequent iterations change and enhance that functionality. On the other hand, Incremental model would develop planned features to the complete extent in every increment. The last increment then produces the final product as intended. Stephens R. explains this well using concept of fidelity initially used by Karl Scotland. (Stephens R. 2015, 286).

Iterative model allows reorientation of project before each new iteration as the development is based on evaluation of features developed on basic level before they are finalized or shaped fully. This makes also management of changed requirements easier than any Predictive development model.

Many organizations utilize combination of both Iterative and Incremental development models. The Unified process (Jacobson, I., Booch, G. & Rumbaugh, J. 1999, 7) for example describes the combined use of Iterative and Incremental models.

4.2.2 Prototype Model

The prototype is basically a mimic or replica of product or set of product features. For example, in construction industry, typically a model or prototype is developed initially to visualize how the construction would look like when finished. Similarly, in SW development, proof-of-concepts or prototype SW programs are created to understand the behaviour of the intended product features. It allows to grasp the shortfalls and improvement needs during initial development stage to create the final product. User Interface mock ups are good examples of SW prototyping.

The prototypes can be classified as Throwaway Prototypes, Evolutionary Prototypes and Incremental Prototypes (Stephens R. 2015, 289).

- Throwaway prototype is created mainly as means of quickly understanding target product behaviour and requirements. Once that is achieved, the prototype is scrapped and the product development is done from scratch.

- Evolutionary prototype is built in such a way that the prototype itself can be modified and developed further gradually to create the final product.

- In Incremental prototyping, multiple prototypes are created with each prototype trying to address certain features/functionalities of the product. These prototypes are then refined as and when they are evaluated. At later stage, these prototypes are merged or integrated together to create the final product.

The Evolutionary and Incremental prototyping requires good discipline such as conforming to coding guidelines, creating some level of documentation and test cases etc. So, they are typically more time consuming and demanding as compared to Throwaway prototypes. Incremental prototyping allows multiple teams to work on different feature sets of product and can speed up development as compared to simple Evolutionary prototyping.

4.2.3 Spiral Model

Spiral model was introduced by Barry Bohem in 1986. See FIGURE 8 below taken from IEEE journal.

Spiral model introduced by Bohem proposes risk-driven approach. Each cycle of the spiral model requires the use of prototyping. The risk assessment is carried out in each cycle against the constraints such as costs, resources and business objectives to make the decision whether to continue the development further. The number of cycles are executed to achieve specific objectives such as finalizing requirements, specifying design or implementing set of features etc. until the final product is ready.

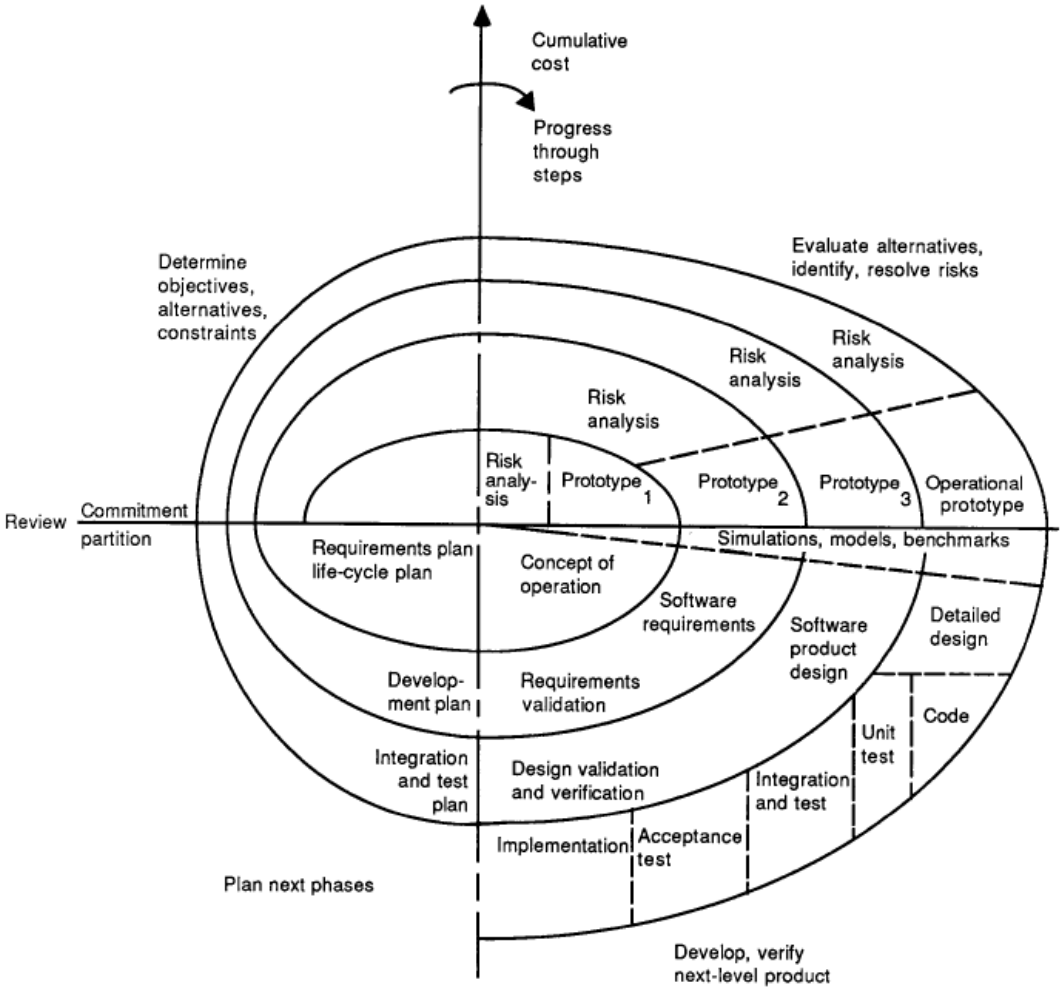

FIGURE 8. Spiral model (Bohem B., IEEE journal 1988, 64)

Spiral model describes 7 steps as part of each cycle. These steps are described below:

1. Identify needs and constraints
2. Determine objectives for the target product
3. Identify various alternatives of achieving objectives
4. Evaluate alternatives against objectives and constraints. Here the risks are identified.
5. Develop risk resolution strategies
6. Determine the risk for completion of the development
7. Plan the action based on step 6 to complete the development

After completion of these steps, assessment is made to decide the continuation of the development. The next cycle commences in case decision is in favour of continuation. In case of negative outcome, the development is stopped.

The Spiral model allows adaption of any type of development model as part of a cycle. Each cycle can adapt different model if needed so. Also, extensive prototyping and risk assessment allows meeting the real objectives of the target product. It helps avoiding penalties due to late changes to requirements or defects identification at later stage during development.

4.2.4 Rapid Application Development Model

Rapid Application Development model is also widely known was RAD model. While most of the development models described above focus on getting the product development done to meet the objectives towards the target products, minimise the defects, risks etc., RAD model is focusing on accelerating the development. It also tries to address the changing needs and requirements in an efficient manner. Short iterations and increments are used for everything such as requirements gathering, design development or validation etc.

Some of the key techniques used in RAD are summarised below (Stephans R. 2015, 305).

- Small teams and short iterations
- Requirements gathering via focus groups, prototyping, brainstorming etc.
- Continuous customer testing of designs via prototyping rounds
- Continuous integration of new code
- Informal reviews and communication among team members
- Timeboxing (scope can change, but completion date of iteration cannot change)

RAD model is more resource centric. Typically, highly skilled resources are needed. In addition, some other resources such as various modelling and

automation tools need to be utilised as well. In RAD model, separate teams work on development of components in parallel. These components are then integrated together in timely manner to produce the prototypes for early evaluation.

RAD model is suitable for quickly testing out intended functionality by prototyping rather than developing the SW product itself. While continuous customer testing typically helps, it can also create problems in some cases (Not correct customer representatives/experts, frustration of frequent evaluation etc.). Also, the large and complex product development might be tricky because such development projects involve large teams and communication overhead can easily become an hindrance.

4.2.5 Agile Models

Agile concept (rather than model) was first introduced by group of developers in 2001. The group published *Manifesto of Agile Software Development*. The manifesto lists their priorities as below:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

In addition to the manifesto, there are 12 guiding principles listed on manifesto's website www.agilemanifesto.org/principles.html. The author's interpretation of those is summarized below (In some places, original text is preserved due to its simplicity).

1. Customer satisfaction through early and continuous software delivery is of highest priority.
2. Always welcome changing requirements

3. Deliver working software frequently with short time periods (2-6 weeks cycles)
4. Business people and developers must work together daily during entire project
5. Build projects around motivated individuals, provide them needed environment and support and trust them to get job done
6. Use face-to-face conversation as preferred communication method
7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. All involved stakeholders shall always maintain a constant pace.
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity is essential
11. The best architecture, requirements and designs emerge from self-organizing teams
12. Team evaluates and adjusts the behaviour periodically to become more effective

While all above principles are relevant in agile development, the principle 3, 5, 6 and 11 form the backbone for agile processes. Typical agile development expects team to be in same physical location for constant and close cooperation. In addition, the team needs to be trusted and given freedom to get the job done as per agreed scope and schedule.

It is worth noting that Agile development mentioned above cannot be treated as a development model. Instead, there are development practices which follow the Agile manifesto and principles explained above and those practices can be treated as Agile development models. SCRUM, XP (Extreme Programming), FDD (Feature-Driven Development) and Lean are some of the popular examples. There are several other models such as Crystal, DSDM etc. Although not elaborated in details, those are also studied to achieve objectives of this thesis.

1. SCRUM

Scrum model introduces two important roles for the team:

- **Product Owner**, representing customer/user responsible for defining the user stories and prioritizing project goals. The prioritized list of user stories and features is called “Product Backlog”.
- **Scrum Master**, responsible for resolving conflicts within the team and make sure that team follows right Scrum practices. He/she is also responsible to motivate team to improve practices and execution efficiency.

Scrum model uses concept of Sprints. Sprint is timeboxed incremental iteration (Stephens R. 2015, 328) typically about 1-4 weeks in duration. Each sprint is expected to produce full tested and approved SW deliverable a.k.a. Potentially Shippable Increment (PSI).

Before each sprint begins, a sprint planning meeting is held where Product Owner introduces user stories to be implemented during coming sprint. After brainstorming, user stories selected for sprint are removed from product backlog and transferred to sprint backlog.

During sprint, team holds 10-15 min daily scrum a.k.a. “standup” meeting to discuss and evaluate the progress between the scrums.

When the sprint ends, a sprint review meeting is held together with Product Owner to make sure PSI produced during sprint meets the expectations. If Product Owner feels that user stories are not fully fulfilled or incorrectly implemented, then such user stories are again planned for next sprint.

After the sprint review meeting, Scrum Master chairs special review meeting to assess how sprint was executed, what would be improvement ideas for coming sprints etc.

Scrum process also introduces concept of “Burndown Chart” to measure the progress. Typically, the user stories in product backlog are mapped against sprint numbers to see how the overall progress is being made to meet project goals. In addition, a term “Velocity” is used which represents amount of work done during the sprint.

2. XP (Extreme Programming)

Extreme Programming commonly known as XP is about taking normal programming practices to extreme level (Stephans R. 2015, 349). Pair programming and Test Driven Development (TDD) are examples of extreme programming.

Traditionally, development process involves a routine task of code review. So, different developer reviews the piece of code written by another developer. Quite often the piece of code to be reviewed is selected either randomly or based on criticality. However, it is almost always only limited to portion of the code rather than whole set of code written by the developer in between the code review sessions. In pair programming, two developers work together and alternate their roles periodically. The developer in a role of “Pilot” or “Driver” writes the code and continues with a monologue which explains the logic behind the code. The second developer assumes the role of “Observer” or “Navigator” and ensures that the code being written really makes sense and suggest possible corrections/improvements on the go. This enables continuous code review as well as code improvement.

In Test Driven Development (TDD), the test cases verifying expected result of the implementation are written before the actual code is written. When the code is written, it is executed to test those existing test cases. FIGURE 9 below illustrates the simplified process or logic of TDD.

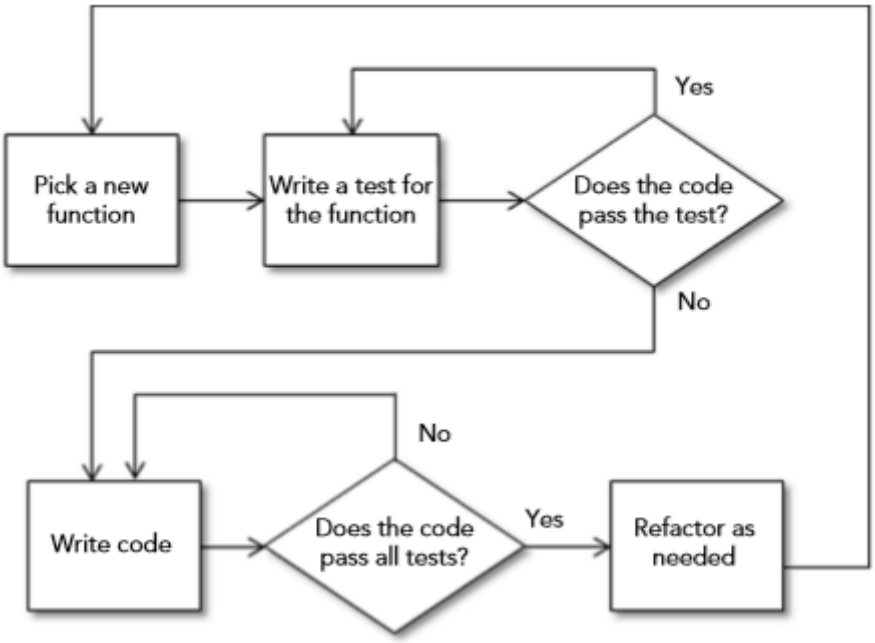

FIGURE 9. Test Driven Development Flowchart (Stephans R. 2015, 358)

TDD enables better conformance to meet the intended objective or goals of the implementation because tests are written in advance with expected end result in mind, instead of focusing on testing the written code. So, it is a reverse, but useful process as compared to traditional development.

There are many other Agile models such as Feature Driven Development (FDD), Crystal, Dynamic Systems Development Method (DSDM) etc. Those are studied, but not discussed here in depth.

4.2.6 Continuous Integration, Continuous Delivery and Continuous Deployment

Continuous Integration (CI) as defined by Martin Fowler states “a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily – leading to multiple integrations per day.” (Ståhl D. & Mårtensson T. 2018, 12).

Continuous Integration by nature is a practice developers need to adapt. The idea behind it is to avoid integrating work of developers with long gaps because the integration gets much more complex as time passes by. The chances to introduce and detect defects in software increase with late integration, especially for large projects with several developers working on it. Frequent integration addresses the issue of such late findings and increased rework. While Continuous Integration inherently doesn't really demand any particular toolset or automation, the right tools and processes make it more feasible to apply in practice.

Continuous Delivery can be interpreted as a next step in software development and deployment process after Continuous Integration. Continuous Delivery refers to releasing the software as often as possible. The expectation is that the changes are integrated, tested and software is released quite frequently. While Continuous Integration does not mandate automation, Continuous Delivery more or less requires that. The tools such as build robots, automatic testing framework etc. needs to be deployed so that frequent releases can be made.

Continuous Delivery contrasts the traditional delivery methods which work on the principle of "identify release candidates at the end" (Ståhl D. & Mårtensson T. 2018, 15). Frequent releases help developers in evaluating the software features and functionality quite often with stakeholders and customers. So, in Continuous Delivery, the feedback loop can be rather short and it helps in identifying gaps between understanding of developers about requirements and the expectations of stakeholders quickly as compared to traditional approach. The developers can then implement the needed changes to produce the next release and get new feedback from the stakeholders. This is very efficient as a lot of rework at later stage can be avoided. It also helps in prioritising implementation of more valuable or urgent features first.

Continuous Deployment as a term is quite often used for Continuous Delivery. In reality, the Continuous Deployment refers to not only releasing software, but also making it available for its users. So, it is like productizing the released software with almost 0 downtime. Continuous Delivery and Continuous Deployment can be differentiated better while dealing with website updates or products which are

part of Software-as-a-Service (Saas) offering. While Continuous Delivery focuses on producing a software release, Continuous Deployment requires such a release been automatically tested or validated against the target system and taken into use. And for Continuous Deployment to happen, Continuous Delivery is a pre-requisite.

FIGURE 10 below illustrates the differences between Continuous Integration, Continuous Delivery and Continuous Deployment.

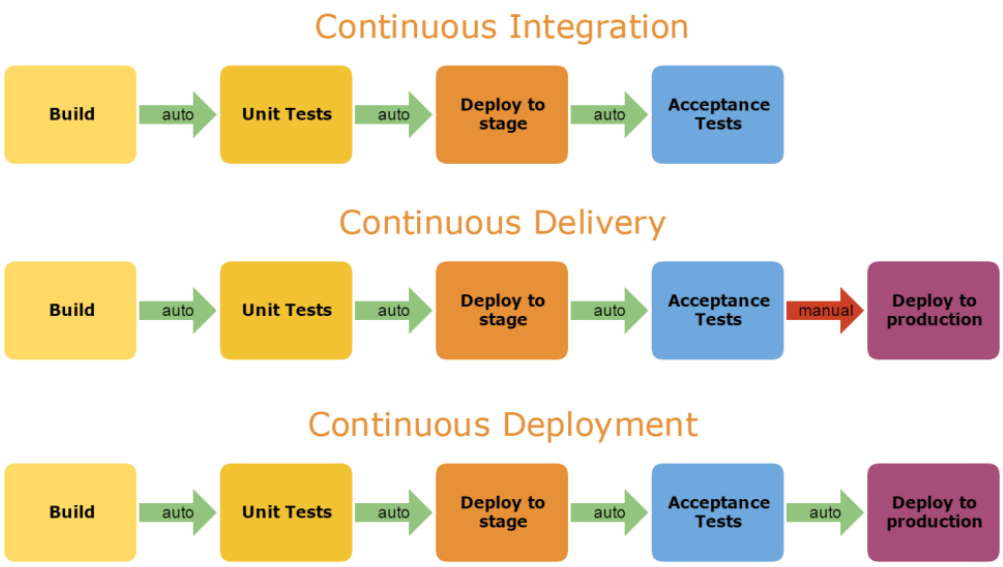

FIGURE 10. Difference between CI, Continuous Delivery and Continuous Deployment (Nastel, Cited 21.11.2018)

4.2.7 DevOps

DevOps comes from the words Development and Operations. It encourages to use automation and monitoring throughout SW development until the deployment. While many people treat DevOps as Continuous Delivery, there are certain differences. The major difference is that Continuous Development is process oriented while DevOps is culture oriented. It focuses on building the mindset and organizational culture to be structured for quick and extensive collaboration. For some, DevOps can be seen as a superset including Continuous

Integration, Continuous Delivery, Continuous Deployment and Operations. FIGURE 11 below illustrates that mindset. It must be noted that there is no common understanding or clear differentiation available in the development community regarding the differences between the two.

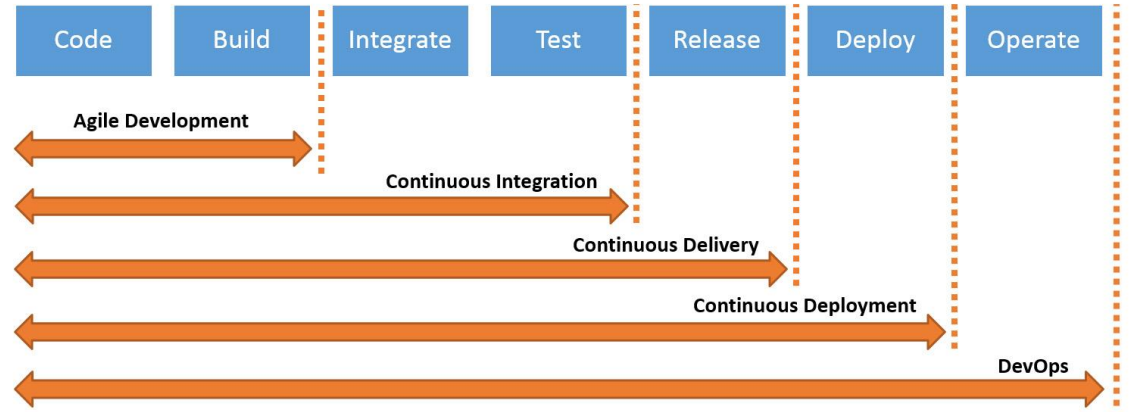

FIGURE 11. Comparison - Agile Development, CI, Continuous Delivery, Continuous Deployment and DevOps (Watts, S. 2017, Cited 22.11.2018)

FIGURE 12 below provides an overview of most common tools used for DevOps. Most of these are used also for Continuous Integration, Continuous Delivery and Continuous Deployment.

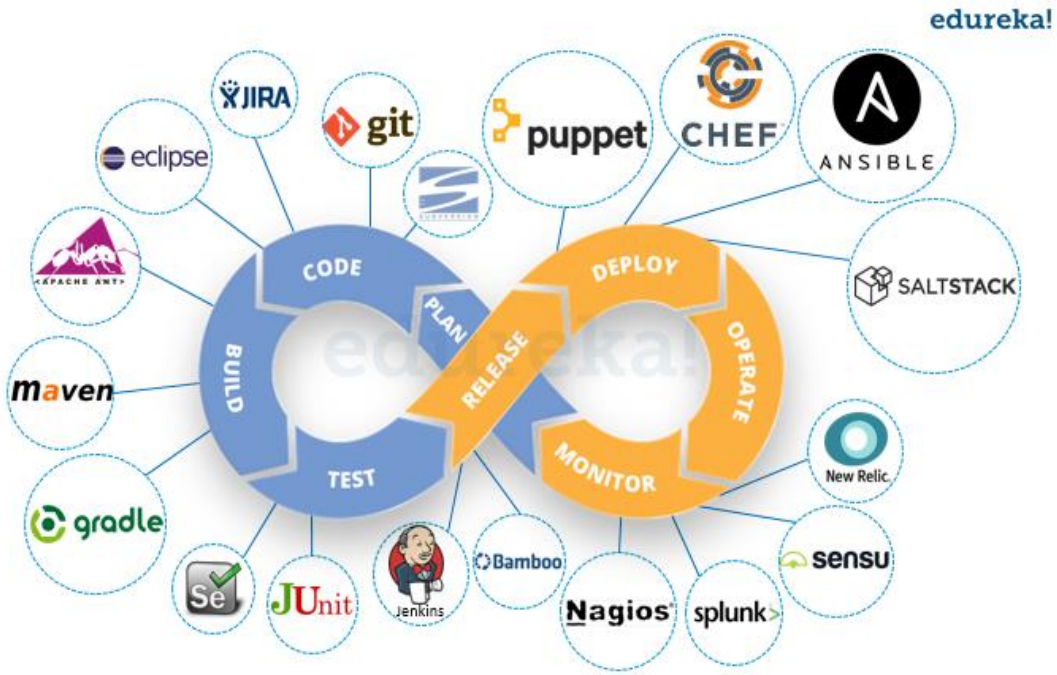

FIGURE 12. DevOps Model and Tools (Zha P. & Khan R. 2018)

DevOps has its values listed as CALMR which stands for

- Culture
- Automation
- Lean Flow
- Measurement
- Recovery

The Scaled Agile Framework, most commonly known as SAFe implements DevOps.

5 DEVELOPMENT BY OUTSOURCING

In this chapter, we focus on the practices used for SW development projects considering the topic of this thesis. The organizations use collaboration or outsourcing for most large-scale development projects. Outsourcing means procuring the services from supplier organization instead of using own resources. There are several reasons to utilize collaboration or outsourcing. Below some common reasons are listed from perspective of outsourcing in SW development. One or more of those can be applicable depending upon the organization.

- **Lack of internal resources**
  Typically, the expert organization exists with core competences, but the overall number of resources are much lower than needed to execute the project. For example – SW programmers or test engineers etc.

- **Lack of needed competences**
  The development project might require new technology knowledge or skills which are missing in the present organization. In such cases, using collaboration or consultancy services for entire or part duration of project is cost-effective and efficient as compared to having new resources recruited.

- **Cost and Schedule Constraints**
  Quite often, outsourcing provides opportunities to reduce the costs as compared to using own resources for development. The countries such as India, China and countries in East Europe have been able to provide the resources at much lower costs to organizations based in Central and Western Europe and North America. Also, in some countries, the regulations make it difficult and expensive for organizations to hire own resources.

- **Internal focus on core competencies**

  Most organizations prefer to utilise the internal resources a.k.a. core competences towards business-critical activities. The other activities requiring so called noncore competences are outsourced to organizations which can perform them better (Greaver M. 1999, 70). This is good management practice to keep focus on organization's core business objectives.

5.1 Common Terms used in Outsourcing

Before going into further details about outsourcing practices etc., let's familiarize with some of the outsourcing related terms used in the organization used as a case study here.

- **Supplier/Vendor**:

  Another organization providing services or products.

- **Subcontractor:**

  Hired organization providing resources to execute certain part of contract or project. The process of using hired resourcing is called "subcontracting" and it is older term as compared to "outsourcing".

- **Hired-in:**

  The supplier resources are located at customer premise. The customer is giving out tasks to the supplier resources as if they are own resources/employees. In generic outsourcing terms, this would be called "onsite assignment".

- **Frame Agreement:**

  A broad contractual agreement with supplier for overall cooperation defining generic contractual terms and conditions, resource pricing, warranties etc.

This agreement provides the established frame to issue individual project/purchase orders to request specific work or deliveries etc.

- **Purchase/Project Order (PO)**:

  A contractual document issued to supplier to order specific work or deliverables from that supplier. Typically, PO is issued against the offer received from supplier for certain work or deliverables. Once issued, PO is always valid for certain period, often 3-6 months' duration. PO can also describe any agreed deviations to Frame Agreement and payment terms etc.

- **Request for Proposal (RFP)**:

  Based on needs and requirements, RFP is issued to supplier. RFP defines the requirements and expectations for deliverables expected from supplier. In response, supplier provides the proposal or offer detailing how the requirements would be fulfilled etc. Quite often the cost or pricing for the work is also included in the proposal.

- **Request for Quote (RFQ)**:

  RFQ is usually issued for getting the offer from supplier primarily to understand the final cost of the work to be requested from supplier. Typically, RFQ shall be issued after reviewing the proposal received from supplier in response to RFP and then agreeing on final scope and schedule etc. to receive the final quote. In response, supplier provides the final offer with the pricing details. If there are any deviations w.r.t. Frame Agreement for payment or contract terms, then those are also detailed in the final offer. This final offer if accepted, serves as basis to issue a PO to the supplier.

- **Invoice:**

  The document used by supplier for billing the cost to the customer according to terms agreed in PO.

5.2 Contracts in Outsourcing

The agreement between a customer and supplier basically forms the contract. While a verbal agreement can also be a contract, it is difficult to utilise it legally when it comes to resolving issues when problems arise. So, contract always assumes a form of a formal document or set of documents. Frame Agreement and Purchase Order are typical examples of contract documents.

The different forms of contracts used in SW outsourcing are explained below:

5.2.1 Fixed-Price Contracts:

In Fixed-Price contracts, the cost to be reimbursed by the customer for the work done by supplier or subcontractor is fixed. The cost is directly associated with the fixed scope or deliverables. The risk of unexpected cost overruns etc. remains fully with the supplier or subcontractor when the agreed scope is to be fulfilled. Fixed-Price contracts are good when customer requirements are fully clear and well defined. But if the requirements are poorly defined or the scope agreed in contract starts to change a lot, then the customer would suffer from considerable cost overrun. The changes to agreed scope is termed as “scope creep”. Every change to agreed scope will be evaluated by supplier or subcontractor as a change request (commonly referred as CR) and the cost for each change request would be calculated with additional risk buffer etc. So, the effective increase in cost is exponential with introduction of every change request. Below are different variants of Fixed-Price contracts.

- **Firm-Fixed-Price (FFP):**
  FFP contract is made so that supplier or subcontractor assumes the full responsibility of delivering agreed deliverables in the contract according to committed schedule for agreed cost. FFP contract is good for customer as there is no risk for customer towards any cost overruns by supplier or subcontractor.

- **Fixed-Price-Incentive-Fee (FPIF):**
  In FPIF contracts, a higher ceiling or cost roof is agreed between parties taking into account most probable risks. Supplier or subcontractor is expected to provide timely deliverables with the total cost not exceeding the ceiling. If the deliverables are provided with lower cost than the ceiling, then supplier or subcontractor gets an incentive based on agreed profit sharing formula. FPIF contracts can benefit both the parties if planned and executed well. The cost risk for supplier is lower in FPIF as compared to FFP contract.

- **Fixed-Price-Award-Fee (FPAF):**
  FPAF contract is similar to FPIF contract. FPIF contract is based on specific goals to be reached by supplier or subcontractor and the incentive is determined with pre-agreed profit sharing formula. On the other hand, FPAF contract is split into award periods. Supplier or subcontractor performance during each period is entitled to certain award or incentives from customer. The decision of giving out award is solely at customer's discretion.

5.2.2 Cost-Plus Contracts:

In Cost-Plus contract, customer reimburses the actual costs supplier or subcontractor bears for getting the agreed work done and provides additional fees as a profit for the supplier or subcontractor. The risk of cost overrun lies with customer in Cost-Plus contracts. Below are some variants of Cost-Plus contracts.

- **Cost-Plus-Fixed-Fee (CPFF):**
  In CPFF contract, customer pays all the cost to supplier or subcontractor for the work done and pays the fixed sum or fee agreed in the contract (typically based on percentage of originally estimated costs)

- **Cost-Plus-Incentive-Fee (CPIF):**
  In CPIF contract, customer pays for the certain costs and also incentives to the supplier or subcontractor based on achievement of agreed performance

targets. In CPIF contract, any difference between the actual costs and the pre-agreed baseline costs are shared between customer and supplier according to formula agreed in the contract. The cost risk is bit lower for customer in CPIF contract as compared to CPFF contract.

- **Cost-Plus-Award-Fee (CPAF):**
  In CPAF contract, customer pays the actual costs incurred by the supplier or subcontractor and based on certain agreed performance objectives provides the additional fees to the supplier or subcontractor. The assessment of performance targets and the determination of award is solely at discretion of the customer.

5.2.3 Time and Material (T&M) Contract:

In T&M contract, customer typically pays for the subcontractor resource costs (cost per person) and the actual cost of materials and if agreed, an additional fees. In most SW T&M contracts, the resource rates are agreed based on the competence levels, experience and resource roles etc. Quite often, these rates are defined in the Frame Agreement between the two parties. However, purchase orders with mutual consent can override the Frame Agreement rates.

T&M contract is quite similar to Cost-Plus contract where the cost risk remains largely with the customer. However T&M contract allows customer to hire specific resources to address internal resource or competence gap faced during certain time period.

T&M contract is good when the requirements are not clearly defined and progressive development is needed where new requirements are defined or original requirements are refined during the development. As compared to fixed-price contracts, customer needs to pay more attention to avoid so-called “budget creep” or “cost creep” instead of scope creep. And that is typically done by closely following the budgeted costs against the use of T&M resources.

5.2.4 Agile Contracts:

There have been discussions ongoing in development community regarding the contractual models used traditionally and what changes would be needed to contracts to complement agile development practices. The SAFe community has been proposing a collaborative approach to agile contracts. One contractual model proposed is called SAFe Managed-Investment Contract. (SAFe Agile Contracts, Cited 23.11.2018)

SAFe Managed-Investment Contract proposes two phases to the contract. First one being the Precommitment phase and second one as Execution phase. The concept essentially proposes that instead of making the contracts the traditional way with full scope, budget and schedule, the contracts shall be based on progressive development. FIGURE 13 below illustrates the Precommitment phase.

The Precommitment Phase process stresses on collaborative approach whereby customer and supplier would agree on Vision and Roadmap. In addition, the Minimum Viable Product (MVP) and potential content for one or more Program Increments (PI) are identified. The contractual framework is then agreed to enlist the terms and conditions, commitment from customer for funding the planned PIs and criteria for continuing with the work etc.

The customer can initiate precommitment phase with multiple suppliers and based on the outcomes from different suppliers, can decide with whom the further development can continue.

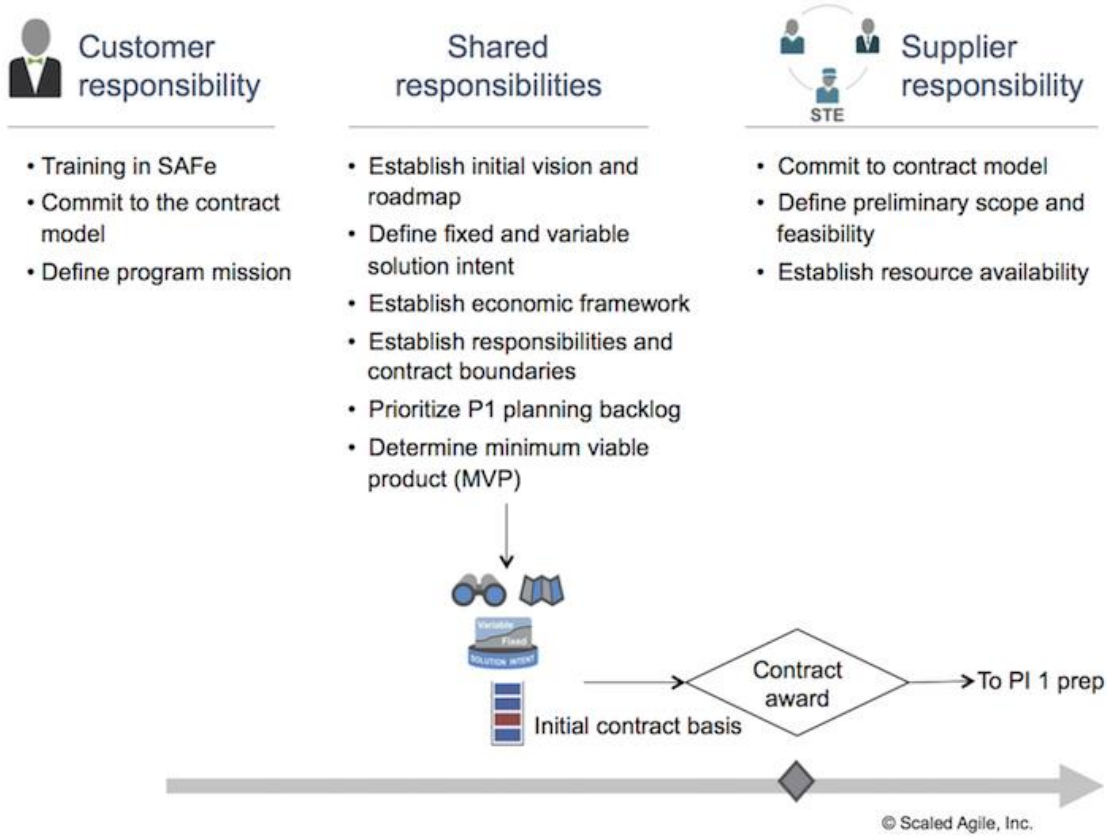

FIGURE 13. SAFe Managed-Investment Contract Precommitment Phase (SAFe Agile Contracts, Cited 23.11.208)

The SAFe Managed-Investment Contract model describes the second phase as Execution phase having 4 key activities:

1. **PI Preparation:** Both customer and supplier work together to plan content and prepare for the 1st PI planning session

2. **PI Planning:** Focuses on entire program and planning the 1st PI on iteration level details

3. **PI Execution:** Supplier is mostly involved in PI execution. However, customer involvement happens from time to time, especially for evaluating system demos

4. **PI Evaluation:** Completion of each PI is treated as a key milestone. The solution demo is held and based on evaluation of the demo, the decision

is made whether to continue with next PI or stop the development or change the course in any way if needed.

The SAFe Managed-Investment Contract Execution phase is illustrated below in FIGURE 14

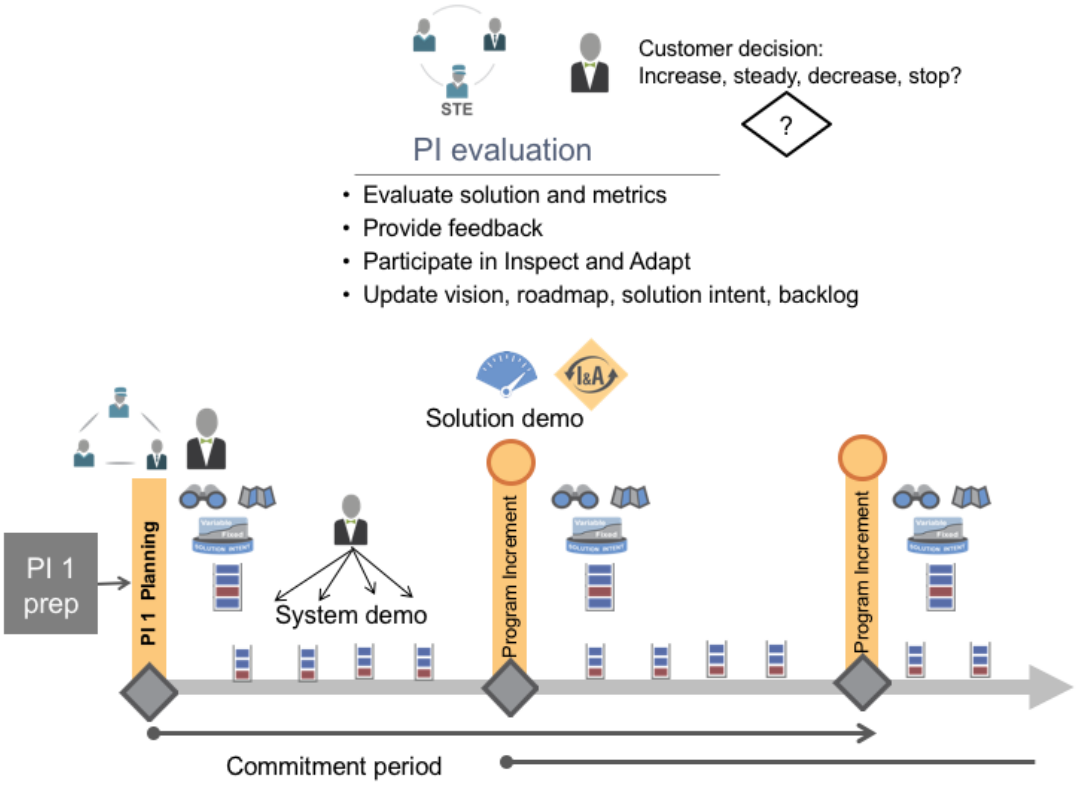

FIGURE 14. SAFe Managed-Investment Contract Execution Phase (SAFe Agile Contracts, Cited 23.11.2018)

5.3 Outsourcing Models

The outsourcing of SW development can utilise different models or approaches. A few common ones are explained below:

- **Offshore outsourcing:**
  In offshore outsourcing, the supplier or subcontracting company is typically located in another country. The main driver behind offshore outsourcing is

cost savings due to cheap labour. India and China for example are considered major destinations for outsourcing by European or US companies. In some cases, offshore outsourcing is used to address the needs of the consumers in that particular country or region. It is mostly applicable for SW as Service offerings.

- **Nearshore outsourcing:**
  In nearshore outsourcing, the supplier or subcontractor company is located in the neighbouring countries. Example – Quite many Finnish companies utilise subcontractors from Estonia. Nearshore outsourcing may not be as cost effective as offshore outsourcing. But it enables the close cooperation and frequent visits between customer and subcontractor.

- **Onshore outsourcing:**
  Onshore outsourcing is quite similar to Nearshore outsourcing. The main difference is that the supplier or subcontracting company is typically located in the same country or region. While the direct cost advantages with this model are not as good as offshore or nearshore outsourcing, the model is effective in development requiring very close and continuous cooperation between two parties. The culture, language and travel barriers are almost negligible in this model and can be effective for fast paced development.

- **Onsite outsourcing:**
  In onsite outsourcing, the resources of supplier or subcontractor are located on the customer premise. One example of onsite resource is referred by customer as “hired-in”. Hired-ins are utilised specifically to fill the resourcing gap the customer team has. This resourcing gap can be either due to lack of enough internal resources to meet the development targets or the lack of competence in customer resource pool which needs to be addressed. In some offshoring models, onsite hired-ins are utilised to manage the offshore development being done by the same supplier or subcontractor. In onsite outsourcing, the customer and hired-ins work together, managed by customer as if they both are customer’s internal resources. Note that not all the onsite resources necessarily be managed by customer. They can also be co-located

in the customer premise and still function own their own without direct customer supervision.

In onsite outsourcing, the main challenge for customer is to safeguard confidential and sensitive internal information from being accessed by supplier or subcontractor resources. So, proper management of access rights to premises and tools (especially document management system) must be in place. At the same time, it is important that the onsite resources do not feel alienated from rest of the team. So, the team building and cooperative culture becomes very important.

5.4 Outsourcing models vs Development models

The outsourcing models and contracts used, have clear benefits and drawbacks when compared against SW development models. The customer measures or assesses the success of development or collaboration based on fulfilment of the scope or deliverables to be provided by supplier or subcontractor within the agreed schedule and budget frame.

Predictive models such as Waterfall, V-Model or Incremental development model work well when requirements are well-defined and well-understood. These development models allow the customer to effectively utilise most contractual models and also outsourcing models. The fixed-price contacts can be allotted to get black-box development done for an entire product or to develop part of the product where requirements are well defined. Having fixed-price contract when requirements are unclear or poorly defined, can result in cost overrun for supplier or subcontractor. As a short term worst scenario, supplier may not be able to execute the contract and that can harm the customer business plans. If short term impact is somehow absorbed by the supplier, then it can trigger in higher pricing in future contracts to mitigate the risk, even when requirements are well defined by customer. The communication between both the parties in such scenarios bears the great importance. Messages can be easily misinterpreted in the context

of things already going wrong and it can put either or both parties in denial of cooperation mode. The teams can lose focus and motivation to complete the deliverables successfully and that would hurt both the parties' interests.

If there is certain level of ambiguity or risk, the cost-plus contracts can be used. T&M contracts can also be used in case the degree of uncertainty about requirements is higher and customer wants to have tight control on the development done by the supplier or subcontractor. However, the customer would likely to have lower cost benefits in utilising such contracts. In some T&M contracts, the final cost can deviate heavily from the budgeted cost or estimates if customer fails to closely monitor the usage of hired T&M resources.

The challenges for contractual and outsourcing models start to get prominence when the development is done using one or more Adaptive models.

The benefits and challenges for each of the outsourcing models are described as a comparison table (appendix 1).

5.5 Influencing factors for outsourced SW development

Let's take a look at what would be the key influencing factors when outsourcing is used for SW development.

5.5.1 Customer organization and strategy

Choosing the right outsourcing model requires that right SW development model has been taken into account to effectively use outsourcing to bring benefits to customer organization. A couple of important factors in deciding right development model are customer's own resource pool and strategy to maintain and develop the competences. Customer must evaluate number of available resources, competencies of the available resources and possibility to acquire more resources or develop competencies of existing resources in order to plan

on a SW development process. Most organizations want to maintain so-called “core knowledge” and “core competencies” internally. After evaluating the internal resource availability and with clear resourcing strategy, customer can define the right development process. This development process in turn has great influence on which outsourcing strategy to be used. For example – customer might be interested in developing only a solid framework and plug-in architecture by using internal resources. The rest of the development focuses on value-added plug-ins. In such cases, customer can internally use adaptive development processes for framework development and go for fixed price projects with supplier for individual plug-in development.

Without having right resourcing strategies, opting a new development model and then going for outsourcing as a forced alternative can lead to dramatic failures for organizations with long term impact.

5.5.2 Customer Organization – Culture and Processes

The customer organizational culture has considerable impact on the outsourcing strategy. While some organizations can easily adapt the outsourcing strategy, some might struggle to consider that option. It is quite often related to resistance of change, but also fear of losing jobs or control. The way of working and level of mutual trust within organization are also the key factors. The organizational fabric and soft values such as empathy, cultural sensitivities and respect also affect the outsourcing. These values not only can swing the decision about opting outsourcing, but can also decide which suppliers to be selected and which contractual models would likely be picked etc.

In some organization, the procurement management can be so rigid that organization cannot not find a way to go for any other contractual models but FFP and T&M. It is also possible, that supplier management processes can force the selection of only onsite or onshore outsourcing. In some organizations, the use

of hired-in is preferred while in another, it is totally prohibited. All these originate from the organizational culture and its flexibility to adapt to changes.

In addition to culture, the organizational processes influence the customer's decision making regarding outsourcing. For example, organizational processes might demand adherence to specific processes or compliance requirements for suppliers. The customer processes might be conflicting with suppliers own processes or can cause so much overhead for supplier, that supplier can decide to opt out of the cooperation. There are many examples where processes guide the decision-making regarding outsourcing.

5.5.3 Development Team composition

Customer needs to plan the development team composition properly. While defining the teams for the development, most important factor is to ensure that team members can complement each other with the right skills and knowledge and can work together to produce intended deliverable. The human nature, cultural diversity and chemistry within the team play far greater roles than one would imagine. In addition to the competencies, the location of the team members plays a great role in deciding the development model and process. Adaptive development is challenging if team members are located in multiple locations. Also, it is challenging to manage when separate teams are geographically spread while members within each team are co-located together. Adding to these challenges, involving one or more suppliers to this equation requires great deal of planning. Customer needs to take into account both social and professional cultures of supplier, supplier's financial viability, supplier's credibility in the market and most importantly the capability of meeting the objectives and needs of the customer by having right resourcing in context of both competencies and pricing.

5.5.4 Product Management

In order to translate the organization's strategy and objectives in terms of product lifecycle management, some sort of product management organization is needed. Most organizations have Product Manager role towards certain product or product portfolio. The similar role is also needed for SW products and services. In SW development, 2 key roles can exist for product management – Product Manager and Product Owner. Note that the different roles do not mean different resources. Product Manager is typically looking after a whole SW product and interacts with the end users of the product. He/she is also responsible for overall product lifecycle, managing long term roadmap and proposing development needs and strategies. Product Owner on the other hand has similar role on smaller scale. Product Owner focuses on maintaining product requirements and user stories collectively called as "product backlog". He/she is actively involved in planning the SW development with the development team. The Product Owner role is essential in Agile development models. Although it can also exist in predictive development models, it's not that common. For SW product of smaller size and less complexity, Product Manager and Product Owner can be a same resource. However, for large and complex SW product, multiple Product Owners manage smaller components of the product and work with a Product Manager having ultimate responsibility of the entire product.

Product Owner in particular has a very important role in agile software development. The ability of Product Owner to effectively manage the backlog by prioritising and scheduling the backlog items for development has direct impact on the development team, including the supplier resources. In paragraph 5.5.6, it can be seen how Product Owner is involved when different outsourcing strategies are used.

5.5.5 Project Management

Organizations quite often utilise so-called OPM framework where OPM stands for Organizational Project Management to translate its vision and business strategies in concrete development activities resulting in products or services. PMI PMBOK Guide states that “OPM is a strategy execution framework utilizing project, program, and portfolio management as well as organizational enabling practices to consistently and predictably deliver organizational strategy producing better performance, better results, and a sustainable competitive advantage” (PMBOK Guide 5th Edition, 34).

Portfolio management caters to organizational strategies by having carefully selected mix of programs and projects. Each program tries to achieve specific objectives or benefits by managing projects and other related activities. The projects within a program are either related to each other to complete the offering or they are interdependent. That is in contrast to portfolio where the program and projects are quite often not directly related to each other. PMBOK Guide states that “A project is a temporary endeavour undertaken to create a unique product, service or result.” (PMBOK Guide 5th Edition, 31).

FIGURE 15 below illustrates the relationships between portfolio, program and projects.

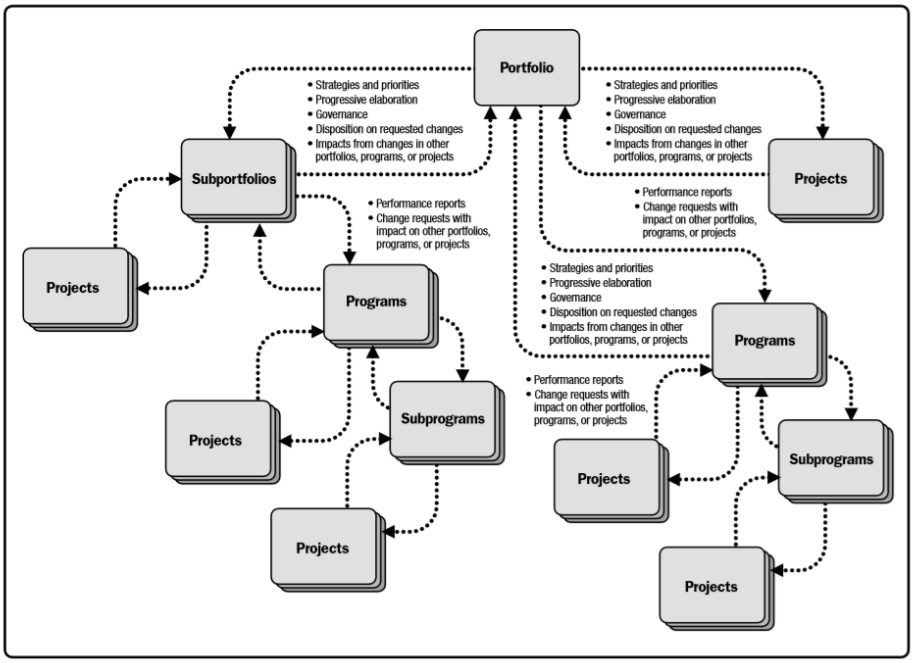

FIGURE 15. Portfolio, Program and Projects (PMBOK Guide 5th Edition, 33)

Portfolio is managed by Portfolio Manager. Program is managed by a Program Manager and Project is managed by a Project Manager. Project Manager is accountable for the project's success and responsible towards meeting the project objectives or fulfilling scope within constraints of given time, budget and resources.

A Project Manager needs to carefully orchestrate the development and ensure smooth progress of development activities within the project. To achieve that, the Project Manager needs to be aware of cultural diversities and sensitivities of the team members and need to be a good communicator and empathetic leader. At the same time, Project Manager needs to take difficult and shroud decisions from time to time in order to ensure that project's obligations towards allocated budget and schedule are honoured. In that context, a Project Manager can influence quite often in selections of outsourcing strategies and selection of development partners/suppliers. Project Manager's attitude and skills often impact how the project team works together and that also impacts the relationship with suppliers involved.

5.5.6 Choice of outsourcing model/s

Outsourcing model chosen for SW development has considerable impact on the successful development. For example, how the customer team would do the agile development for an offshore, nearshore or even onshore outsourced project? Provided that there is a good communication and development infrastructure, it would be possible that customer team and the supplier team in totally different locations indulge in a common development. They can participate in meetings using tools such as Skype, WebEx, teleconferencing etc. and utilise development environment such as opensource GIT. A few ways this can be achieved, but requires greater degree of planning and execution efforts as explained below:

- Customer Product Owner either remotely or at supplier premise, participates scrum and sprint meetings regularly with supplier development team. The supplier team develops one or more components of the product according to scope and actions agreed with Product Owner. The developed components could then be integrated into the master development branch (where all source code resides and used for constant development), managed and maintained by the customer team. This model works best when the SW design is modular and scalable and it is possible to develop different components separately and integrate together.

- Part of the supplier team is located onsite at customer premise. Customer Product Owner can interact with onsite supplier resources. In this model, supplier Project Manager or Coordinator is also located onsite. Although this would allow onsite resources to work closely with customer resources, this cannot solve the problem fully for adaptive development models. The fact still remains that team located offshore, nearshore or onshore supplier premise still would have same challenges as if there were no onsite resources. This model too works best when the SW design is modular and scalable and it is possible to develop different components separately and integrate together.

- Most of the adaptive development models are based on development that relies on culture of trust and commitment and quite often with co-located teams. So, one option is that use an outsourcing model where supplier is not treated as an outside entity but as a partner. In this case, customer would treat the supplier team as if it is one of its own team and both customer resources and development partner resources work together on common development environment. Although this seems to be the way to go, the real life challenges make it very difficult to implement and sustain. This model indeed requires a great bond between organizations and all individuals working together on the development.

5.5.7 Development Environment

Development environment refers to the resources (other than human labour) and tools used to enable the SW development. The development environment collectively refers to the HW (servers, computers used by developers), internet connectivity and policies, operating systems, SW languages, coding guidelines, configuration management system, build and release system, testing and validation HW and SW, tools utilised for coding such as Integrated Development Environment (IDE) like Eclipse or MS Visual Studio etc. Open source development which is community-based, shared SW development utilises tools such as GIT etc. Development environment has major impact on deciding outsourcing strategy. For predictive development, the development environment need not be fully replicated at supplier premise. But for most adaptive development, development environment needs to be shared by customer with supplier or replicated in the supplier premise. This brings in additional concerns related to data security, connectivity and access rights, costs and flexibility in upgrading the development environments.

6 STAKEHOLDERS

Stakeholders here are listed from the perspective of the customer and that means from the organization under study of this thesis, developing the automation system. Although the own resources and project teams etc. are also stakeholders, those are excluded from the list below.

6.1 Internal Customers

Automation system is part of the parent organization's product portfolio and not a separate sellable product on its own. The customers for the parent organization purchasing products and services are called End Customers. For the platform SW of the automation system, the key customers are so-called internal stakeholders responsible for automation system deliveries towards the end customers. These internal customers are resources from parent organization and dealing with products provided to end customers. Typically the colleagues from Sales and Marketing, Customer Support and Delivery Project Management organizations are internal customers for platform SW.

Internal customers are the main stakeholders for platform SW development as they define the key functionalities or requirements towards the automation system. Those requirements directly affect the platform SW. These customers also play an important role in prioritizing development activities and getting funding arranged for the needed development.

6.2 Sponsors

The sponsor is the person responsible for committing the funding for development activities, including development projects. In some cases, there can be multiple sponsors funding development activity jointly.

The development of platform SW is a continuous process and depending upon the target SW release, the sponsors can vary. Although there is a demand to implement or release some functionality, arranging the needed funding can be challenging due to the complex processes and constraints of each sponsoring department within the parent organization. Major sponsors are listed below:

- **Programs/Projects:**
  This funding is quite often secured for priority development activities with a well-defined process framework to initiate and execute the projects.

- **Funding for continuous maintenance of the platform SW**:
  This is based on known backlog of defects/improvements and typical known inflow and turn-around times to deal with those. This funding is usually part of customer organization's own annual budget.

- **Individual sponsorship by another entity within parent organization**:
  This is typically demand based and in most cases, funding is secured easily due to the demand.

- **Collective sponsorship for development**:
  This is most tedious, as quite often negotiations amongst sponsor organizations are required regarding share percentage of sponsorship etc.

6.3 Project Owner

Project Owner is responsible for developing goals and objectives for the development project. Project owner is involved with project since very start, from conception of the project until the project is completed or terminated. For project manager, project owner is the most critical stakeholder. The change request approvals, escalation related to project risks etc. are managed via project owner. Project owner shall support the project manager and project team in executing the project smoothly and removing obstacles as needed.

Project owner is typically an head of a department or organization doing the development. Also, Program manager can be project owner for projects under the program.

6.4 Regulatory authorities

The automation development has to follow stringent regulatory requirements. All the business segments typically need to conform to industry standards related to processes, safety and security aspects of the products and services etc.. In addition, regulatory bodies demand conformance to their requirements. There can be country-specific requirements, which need to be taken into account as well. For example, different countries have different emission requirements for automotive, energy or marine industries.

6.5 Partners

Partners here are typically organizations outside parent organization and institutions such as universities, schools, NGOs etc. Quite often, the partners collaborate together on research projects brining innovative or ground-breaking products or services. But, in some cases, even standard development projects can also be done by partners together. The partnerships provide various angles and thoughts to approach the problem or target. The young and fresh talent from universities can provide creative ideas while experts from industries complement by evaluating and developing those ideas further according to practical reality. These joint activities provide also the opportunities to university students to learn practical challenges, professional etiquettes and develop people network. That helps them transform themselves in future professionals and also create opportunities for employment.

7 CHALLENGES

In this chapter, we will try to understand different challenges customer (used as for case study) is facing when it comes to platform SW development.

7.1 Expectations of sponsors and customers

Typical sponsors for the development of automation system are non-automation personnel. It makes it difficult to sell the ideas of needed development activities in order to secure the funding.

Sponsors expect that their requirements are fulfilled when their request is accepted. However, the current process does not guarantee that the accepted request would see the successful resolution within certain timeframe. Quite often, the request goes to the development backlog and gets buried there, unless it is highly demanded and considered as an urgent need.

Customers (internal) expect that the well tested and validated automation system and functionalities are available when the SW releases are provided. At the same time, the expectations are so that all available functionalities in old automation system are also available in newer system. Although that is a valid expectation, sometimes HW and SW architecture of newer systems can impose constraints. In such cases, managing customer expectations becomes a difficult negotiation game.

Another expectation is usability, clear instructions and guidelines. There is a lot of improvement possibility in this area. User centric design and usability are just started to be used in customer organizations since last couple years and no dedicated expertise in this area exists. Understanding real stakeholder needs by creating well-written user stories and requirements would benefit a lot for developing the automation system effectively and that in turns would benefit also the platform SW development.

Sponsors/customers quite often demand low cost but yet rugged and high quality system. It is challenging to find the balance. But this is usually the case everywhere in industry.

7.2 Resourcing and Competence

The entire automation SW platform implementation has evolved as a complex SW architecture and codebase. Thus developing the platform further requires heavy learning and considerable duration before new resources can learn the existing design and codebase in order to contribute. Some of the most complex parts would take more than a year before an expert resource can really well-understand the implementation details and can effectively troubleshoot the problems.

The continuous development over the years have seen compromises to SW architecture and modular design. The abstraction between the SW layers has not been maintained well and that poses the challenges in involving new resources and suppliers to contribute towards the platform SW development.

7.3 Development Environment and Process

The development process has evolved over the years when it comes to platform SW development. From being traditional waterfall model, to hybrid agile and moving towards agile during the years. However, the development has mostly done by supplier resources while customer resources have mainly been involved in backlog prioritization, system level requirements definition etc., in the process of moving towards agile development. The development environment is largely maintained by supplier. On the surface, this doesn't seem to pose any challenges to customer. But when the holistic view is taken to look at the entire release chain, there are dependent activities within customer organization such as development of applications and making releases for automation system etc. This release

chain cannot be easily enhanced or improved unless the adaptive practices such as continuous integration and continuous delivery are implemented within the customer organization. It more or less forces the customer to take control of the development environment and utilize more internal resources to implement continuous integration and continuous delivery. It is a time consuming and tedious task.

7.4 Dependencies

As explained in paragraph 3.2, platform SW is a middle layer of embedded SW. However, it utilizes information in system configuration files and also interacts with management SW. That is in addition to the key implementation that manages its southbound interfacing towards HAL and northbound interfacing towards applications. The communications with management SW in particular has created a tight dependency between platform SW and management SW implementation and changes made to functionalities need to be evaluated and implemented in both the places. The intended modular architecture of management SW has not been realized in practice. The end result is that any changes in user interface can affect the communication towards the platform SW and implementation of communication methods in management SW can impact the platform SW in terms of performance etc. This cannot be overlooked while dealing the challenges for platform SW development.

7.5 Cooperation with partners and suppliers

The development of platform SW as well as management SW is done in coordination with a single supplier for many years. In reality, the ratio of outsourcing to internal resource utilization is very high. In that context, one can conclude that the development is mainly done by the supplier. At present, only backlog prioritization is done by customer resources.

The above situation has imposed the following challenges:

1. Customer resources and in turn, customer organization as such has become highly dependent on one supplier.

2. Lack of participation in actual development activities such as coding, code reviews or testing by customer team has resulted in a situation that customer resources are no longer in a position to understand the challenges faced by developer team and not able to effectively challenge the design proposals or effort estimations provided by the supplier team. In summary, the internal core competence level has dropped significantly due to high reliance on the supplier.

3. Due to level of complexity of the platform and management SW and the knowledge available with current supplier resources, it is difficult to involve another supplier in the equation. It is also challenging to recruit many developers and SW experts to ramp up the core competence. Although it is not impossible, it is a daunting and time consuming task, taken into account the management approval process for recruiting, finding the suitable candidates and building up their competence without jeopardising the relationship with current supplier.

7.6 Security and Conformance

The customer organization would like to ensure the security regarding the following:

1. The source code for both management SW and platform SW needs to be well protected by using methods such as obfuscation and encryption in order to safeguard its competitive edge regarding automation system superiority from its competitors and rug elements.

2. Use of open source software (OSS) components or any other licensed SW components needs to be properly managed in order to avoid causing infringements and facing heavy financial penalties

3. Special attention needs to be paid for cyber security requirements. It needs to be done from the perspective of safeguarding own interests as well as safeguarding its products against the cyber security threats. Also, the conformance to regulatory requirements regarding security is mandatory.

4. Any information collected and used on individual level needs to be managed in secured manner for conformance with GDPR rules.

All these pose own challenges already even using own resources. The situation becomes more challenging when supplier resources are involved.

7.7 Conflicts of Interest

Customer organization faces conflicts of interests on multiple fronts. As the automation system developed by the organization is used across entire product portfolio owned by different business entities or business lines within the parent organization, the prioritization of development backlog becomes challenging. These are the internal conflicts due to different needs of different business lines. A well-defined order intake and backlog management system is needed to manage these conflicts in a systematic manner without affecting the sentiments of stakeholders from different business lines. In short, the organization needs its own portfolio management in place.

In addition to internal conflicts, there are conflict of interests when dealing with suppliers. A few of these are listed below with respect to current situation with platform SW as well as management SW development having a single key supplier:

- Customer would like to ramp up internal core competence whereas supplier would like to have more business and in turn increase in its resourcing towards the customer.

- Customer would like to optimize the processes and expenses related to SW development. On the other hand, any initiatives taken to achieve that goal is likely going to affect the supplier negatively. It is mainly due to the fact that the supplier has almost entire share of the SW development.

- Customer would like to consider an option of utilising more than one supplier in outsourcing to safeguard the future SW development. Supplier on the other hand would feel threatened with the knowledge of such actions by customer. Even if supplier management would realise the customer viewpoint, it will be hard for supplier resources working for many years with customer to anticipate such a development. The emotions involved in long term relations can cause anxiety and demoralise the supplier team, at least in short term. It needs to be noted that involving already new resources of its own in the development activity would cause some nervousness with the supplier resources. So, adding one or more suppliers to the development activities can be even more stressful considering that those will be their competitors. It would require great management skills from customer team to ensure that supplier team understands the customer viewpoint, remains motivated and will cooperate well with other suppliers.

8 RISKS AND BENEFITS

Let’s take a look at the both the benefits of having a strategic single supplier for outsourced development and on the contrary, the risks involved with such an approach based on the topics discussed above. These are in addition to the reasoning explained in chapter 5 for why customers opt for outsourcing. Although the benefits and risks listed below are valid for typical customer-supplier relationship, those are relevant and applicable in particular for the customer of the case study involving platform SW development.

8.1 Benefits

8.1.1 Partnership of mutual benefits

Using a single strategic supplier for outsourced development is like embracing a happy married life. For such a marriage to last, it requires trust and mutual understanding about each other’s’ needs and interests. And the most important virtue needed from both parties is long term commitment that would overcome the periodical challenges and misunderstandings.

Once a long term strategic partnership is developed, it becomes a productive endeavour. Cooperation between customer and supplier team becomes more coherent. As it can be seen from many industry examples, the supplier resources working for many years with same customer develop such a closeness with customer that they feel a sense of accountability of their actions towards customer’s business value propositions. It is a definitive benefit arising from a long term partnership. Similarly, the bonding created between supplier and customer resources helps in generating new work for supplier resources.

8.1.2 Flexibility

A long term partnership typically encourages supplier to invest in ramping up resourcing and competence that would benefit customer in near and long term. This helps customer quite often to utilise supplier resources for evaluation of certain technologies or new development projects without investing heavily on the internal competence ramp-up. Once the feasibility is evaluated and concrete development plans are developed, customer can focus on investing on internal resources. Supplier on the other hand, benefits from investing in such a competence ramp-up especially if that is done towards the new technology or process development activities. That knowledge and expertise can be utilised by supplier to develop business with new customers.

8.1.3 Cosy Cooperation

With the long term strategic partnership, the foundation of contractual obligations are more or less done. The example would be Frame Agreement between the two organizations and well-defined process for purchase order issuance, warranty claims handling and payment terms etc. The cooperation works well between customer and supplier even when exceptions need to be made to agreed processes. Both the parties focus on getting things done rather than waiting for formal process to take its course. Let's say that a purchase order needs to be issued by customer before supplier resources can start work on certain development activity. When customer-supplier relationship is new or based on short term agreements, supplier will be hesitant to start any work before purchase order is issued by customer irrespective of urgency of the deliverables by customer. But in long-standing relationship, the mutual trust takes the driver's seat when such incidents arise. Supplier resources can start working on the planned activity even with a verbal commitment from customer while purchase order issuance is in progress.

8.2 Risks

The risks here are listed from the perspective of customer unlike the benefits which were analysed above from both customer and supplier perspective.

8.2.1 Business Risk

The biggest risk for customer in relying with the sole strategic supplier for SW development. The customer becomes fully reliant on the supplier. The financial crisis for the supplier or possible takeover of the supplier by another organization can create serious challenges for customer. If the takeover is done by one of the customer competitors, then the impact would be even more severe.

8.2.2 Complacency

The long term cooperation can result in too cosy of a relationship for supplier resources towards the customer. This can result in “taken for granted” situations, where supplier resources fail in keeping the commitments, yet assume that the things are fine due to close relationships. To make things worse, some supplier resources can start prioritising the development activities or backlog items which would be in conflict with customer’s urgent business needs. This quite often can be seen with long lasting T&M contracts.

8.2.3 Vendor-lock

The total reliance of customer on single supplier can create a so-called “vendor-lock” situation for customer. In this case, the typical nature of give-take relationship between customer and supplier quite often gets reversed. Supplier can take advantage of customer’s situation to inflate the contract prices, negotiate the new terms for contracts favouring the supplier etc. If the supplier starts to take advantage too often, it likely would force customer to consider different long term

options. But going for alternative options can be painful journey, especially if the customer resources are lacking the core competence.

9 PROPOSED SOLUTIONS

So far, we looked at the customer situation regarding platform SW development and use of single supplier for the development of platform SW and management SW. We also looked at different SW development models, outsourcing models, challenges faced in the development of platform SW and risks and benefits of using single supplier in SW development. Based on those topics, below are some of the reflections from the author regarding use of outsourcing of platform SW development.

The long term strategic relationship is always beneficial for customer as well as supplier. The current relationship between customer and supplier towards platform development has a strong foundation, mutual trust and committed and competent resources working with platform SW development for very long time. Such a relationship shall always be cherished and the cooperation shall continue. However customer shall also plan the mitigation for risks mentioned above regarding use of single supplier. To achieve that, following actions are proposed:

9.1 Build up the core competence

At present, the core competence related to platform SW and management SW is with supplier. Customer needs to define a clear resourcing strategy for coming years to acquire enough resources of its own and build the competence by ensuring needed training and involvement in hands-on development work with the supplier resources.

9.2 Define SW Development Model

Once the resourcing strategy is clearly defined, customer needs to focus on choosing the right suitable SW development model. Considering the holistic view of entire release chain as discussed earlier in paragraph 7.3, customer shall decide which development model would suit best. At present, the continuous

integration model has been used by supplier for development. If entire release chain needs to be streamlined, then the continuous integration environment needs to be moved under customer control and needs to be complemented by continuous delivery of the SW releases. Once this is done, the newer development models such as DevOps or SAFe can be taken into use as well.

9.3 Emphasize Relationship Management

Special attention needs to be given to attain the emotional aspects of supplier resources. Injecting own resources in development carried out solely by supplier resources for several years can create an anxiety and sense of insecurity within the supplier resources. In order to maintain the effective cooperation, it is utmost important to take special confidence building measures with supplier. Customer shall initially appoint skilled resources in coaching in roles which would constantly interface with the supplier resources. These roles could be project manager, product owner or architect etc. who would smoothly make the supplier resources understand the customer viewpoints, remove the false assumptions and anxiety and build stronger relationships. Nowadays, many organizations spend fortune on so-called “Digital Transformation” to induce a change in organization culture and address the possible resistance to change regarding digitalisation initiatives. Similarly, there is a need to plan proper onboarding of supplier resources to make them aware of customer’s business needs and the reasons for changes being made regarding outsourcing policies or SW development model.

9.4 Improve SW Architecture and Design

It is essential that the SW architecture uses modular design with abstraction across different layers of SW. A well-defined and well-implemented modular architecture creates several possibilities to involve different teams or suppliers to work on individual modules of the SW. Combining that with the right development environment and SW development models can create very efficient process of

SW development with optimum throughput. An example of modular architecture for platform SW could be a communication layer which abstracts the communication towards the HW or the physical layer and provides interfaces for the upper layers. It can then enable integration of different types of communication protocol stacks in the platform SW. Different suppliers can work on integrating different stacks. Similarly, for management SW, a proper modular architecture would enable different suppliers to work on different user interface functionalities and develop those independent to each other. The plug-in architecture can enable developing different services without affecting the existing SW integrity. Without ensuring proper SW architecture and design, one can invite only chaos if development is done with the multiple suppliers.

9.5 Define Multi-sourcing Strategy

To mitigate the possible risks of getting fully reliant on single supplier, customer shall evaluate what are the possible options to involve other suppliers in the equation. Note that the intention here is to reduce the risk of being dependent for non-core development activities. Because building core competence internally shall remain the top priority. Distributing core competence amongst supplier base is not a good strategy in order to mitigate the risk of single supplier.

Customer shall define the right outsourcing strategy by taking into account the cost benefits, cultural diversity and the quality related risks while choosing the new suppliers. It is also recommended to utilise different outsourcing models and contractual models for certain activities to be outsourced. Below are some of the ideas/suggestions which can be considered:

- Outsource the maintenance of CI development environment to new supplier while ensuring that one or more internal resources remain involved and oversee the maintenance activities. Here supplier resources need to be onsite and FPIF or FPAF contract can be used based on performance KPIs and based on the number of resources etc.

- Involve a new supplier in testing activities such as writing ATF test cases, manual executing of test cases or maintaining of test environment etc. This strategy quite often helps in building the competence of new supplier resources gradually, as they get familiar with the customer products and development processes. Different outsourcing and contractual models can be used based on where the resources need to be located, connectivity and access to infrastructure etc.

- Let the new supplier work on creating proof-of-concept or MVP for certain feature of functionality. Some suppliers provide resources which are specialised in developing creative concepts with focus on specific area such as usability or performance. This model can work very well to ramp up the knowledge of the new supplier resources towards customer products and processes. In addition, the resistance from existing supplier resources towards accepting new competitor in arena is low, because the concepting activities do not directly interfere with the mainstream development. This paves the path to create the gradual mutual trust and cooperative development environment for the future. Typically T&M type contract is suitable for such activities.

- Utilise the new supplier to create the plug-in services initially to get familiar with the domain and the functionalities to start with. Then outsource some smaller development which needs to be integrated to mainstream SW development. A FFP contract would be suitable for such activities based on the proposal received from supplier in response to RFQ issued by the customer. Any of the three - offshore, nearshore or onshore outsourcing can be utilised for this based on the quotations received from supplier/s. Onsite resourcing can be used also for such activities. But in most cases, it has little benefit for customer compared to other outsourcing models.

9.6 Ensure Accountability

Quite often, it happens that the contractual documents such as frame agreement or purchase order terms define the accountability and responsibilities of customer and supplier clearly. Let's take a warranty clause as an example where supplier is made responsible to provide a fix for defect found within certain timeframe after implementation is done by supplier resources. This can work out when single supplier is involved in development since start as a sole responsible for development of particular SW feature or functionality. The challenges start to appear if the development is done jointly using supplier resources and customer resources or having multiple suppliers working together on common development. Customer needs to ensure that right tools and processes are put in place so that each supplier can fulfil the promises towards the warranty clause.

9.7 Create a Competitive Environment of Trust

When using multi-sourcing, it is quite important to ensure that all involved parties feel the sense of equality in all terms – equality in terms of how they are treated as compared to other supplier's resources, equality in terms of opportunities presented to them and transparency by customer in making the decision making for awarding the contracts.

The utilisation of RFP and RFQ shall not be viewed merely as a way to follow sourcing process. These are the most important instruments to evaluate the competences of suppliers, cost benefits and make a rational decision of awarding a contract to certain supplier where the overall value is seen the highest. This value is measured not only in terms of the technical solution proposed in RFP or cost advantage provided in the RFQ, but also taken into account the resource availability, various risk factors, assumptions and constraints listed by supplier etc. Utilising RFP and RFQ effectively for awarding development contracts and sharing the knowledge regarding factors which influenced the decision, can help build the trust for all suppliers towards the customer and that also helps building

more cooperative environment. Note that sharing the knowledge of influencing factors in decision making does not mean sharing the sensitive data such as pricing etc. Customer needs to pay attention to what information to be shared and how that shall be presented.

10 CONCLUSION

This thesis was done in order to understand the risks and benefits of having a single supplier involved in SW development, especially where the ratio of outsourcing to internal resourcing is very high. This was done as a part of wider initiative to understand the current SW development process, identify risks and best practices and suggest the possible improvements for the organization used here as a case study.

The thesis was structured with planned sequence of analytical activities, mainly using qualitative research. The intention was to study the different SW development models, outsourcing models and contractual models to try to understand if there is any relationship between them and would certain SW development model derives benefits or suffers due to specific outsourcing or contractual model.

Only a small degree of relevance was found towards the relationships between SW development models and outsourcing models. Author's opinion from this study is that while selecting the right SW development model, it is good to have some degree of understanding of what outsourcing model would suit the best. However, not too much of focus shall be put on this relationship. With the proper infrastructure, development environment and processes in place, most of the outsourcing models can be used.

During the thesis work, it was noticed that the contractual models used for outsourcing work would have greater degree of relevance towards the development models taken into use as compared to the outsourcing models. With right contractual model applied to certain type of development activity, the cost-to-benefits ratio is higher. This however requires a greater degree of planning by customer organization.

It must be noted that the tools and processes used with different development models were studied during the thesis. But those are not elaborated in the this text.

The next focus in the thesis was to study the risks and benefits of single supplier in outsourcing of the SW development. The intention was not to simply analyse risks and consider the option of multi-sourcing i.e. utilising more than one suppliers. Instead, the main intention was to study the risks and find out the possible ways to mitigate those risks. During the development of this thesis, it was evident that it is not worth to jeopardise a well-working and long-term relationship with a supplier by introducing additional suppliers to the SW development, simply because there are possible risks associated with having single supplier. The well -working relationship indicated that supplier resources were highly committed towards doing their best to provide high quality deliverables to customer and there is a strong bonding between the resources on both sides. Considering this, a holistic approach needs to be taken to understand the level of dependability on the existing supplier and the level of so-called “core competence” within the customer organization. It is observed that the core competence needs to be retained within the customer organization and concentrated efforts were needed to plan the concrete resourcing strategy to ramp up the core competence within the organization.

In order to mitigate the single supplier risk in long run, a well-defined strategy is required whereby a foundation and environment needs to be created before injecting a new supplier to the equation. The attention needs to be given to keep existing supplier resources committed and motivated. A transparent sourcing process needs to be deployed to create trust amongst all the parties.

The objectives of this thesis have been successfully achieved as explained above. However it must be noted that due to time constraints, it was not possible to study the newer adaptive SW development models such as DevOps or SAFe thoroughly from all angles. The same was the case with newer contractual models such as Agile contracts. It is recommended to study those further.

REFERENCES

Beck, K. Extreme Programming Explained, 2nd Edition. 2004. Addison-Wesley.

Bosch, J. Speed, Data, and Ecosystems Excelling in a Software-Driven World. 2017. Boca Raton: CRC Press.

Brown, D. & Wilson, S. The Black Book of Outsourcing. 2005. New Jersey: John Wiley & sons.

Cohen, L. & Young, A. Multisourcing: Moving beyond outsourcing to achieve growth and agility. 2006. Gartner Inc.

Constructing Excellence Frame Agreement. http://constructingexcellence.org.uk/tools/frameworkingtoolkit/what-is-a-framework/

Greaver, M. Strategic Outsourcing – A Structured Approach to Outsourcing Decisions and Initiatives. 1999. New York: AMACOM.

Gruver, G., Young, M. and Fulghum, P. A practical Approach to Large-Scale Agile Development. 2012. Addison-Wesley.

Humble, J. & Farley, D. Continuous Delivery Reliable Software Releases Through Build, Test, and Deployment Automation. 2010. Pearson Education.

Hurskainen, M. Analyzing IT Project Success – An Empirical Approach to Critical Success Factors. 2014. Lappeenranta University of Technology.

IEEE Computer Society. Guide to software engineering body of knowledge. Version 3.0.

Jacobson, I., Booch, G. & Rumbaugh, J. The Unified Software Development Process. 1999. Addison Wesley.

Jyväskylä University. Research Methods Maps. https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/en/methodmap

Kim, G., Behr, K. & Spafford, G. The Phoenix Project. 2013. Portland: IT Revolution Press.

Kim, G., Humble, J., Debois, P. & Willis, J. DevOps Handbook How to create world-class Agility, Reliability, & Security in Technology Organizations. 2016. Portland: IT Revolution Press.

Mezak, S. Software without borders. 2006. Los Altos: Earthrise Press.

Nastel. Devops: Continuous Integration vs Continuous Delivery vs Continuous Deployment. Cited 21.11.2018, https://www.nastel.com/blog/devops-continuous-integration-vs-continuous-delivery-vs-continuous-deployment/

Noregaard, T. 2005. Embedded Systems Architecture. 2005. Burlington: Newnes.

Pfleeger, S.L. Software Engineering Theory and Practice. 2nd Edition. New Jersey: Prentice Hall.

PMBOK Guide, A Guide to the Project Management Body of Knowledge. 5th Edition. Pennsylvania: Project Management Institute Inc.

Royce, W. Software Project Management A Unified Framework. 1998. Addison Wesley.

SAFe Agile Contract. Scaled Agile Inc. Cited 23.11.2018, https://www.scaledagileframework.com/agile-contracts/

Sage, A. & Palmer, J. Software Systems Engineering. 1990. John Wiley & sons.

Schwaber, K. & Beedle, M. Agile Software Development with Scrum. 2002. Prentice Hall.

Stephens, R. Beginning Software Engineering. 2015. Indianapolis: John Wiley & sons.

Ståhl, D. & Mårtensson, T. Continuous Practices A Strategic Approach to Accelerating the Software Production System. 2018.

Watts, S. 2017. BMC Blogs. Cited 22.11.2018, http://www.bmc.com/blogs/continuous-delivery-continuous-deployment-continuous-integration-whats-difference/

Agile SW development. Cited 21.04.2017, https://en.wikipedia.org/wiki/Agile_software_development

Application Programming Interface. Cited 05.05.2017, https://en.wikipedia.org/wiki/Application_programming_interface

Embedded Software. Cited 28.04.2017, https://en.wikipedia.org/wiki/Embedded_software

Embedded System. Cited 28.04.2017,
https://en.wikipedia.org/wiki/Embedded_system
Firmware. Cited 06.05.2017, Available at
https://en.wikipedia.org/wiki/Firmware

Graphical User Interface. Cited 28.04.2017,
https://en.wikipedia.org/wiki/Graphical_user_interface

Hardware Abstraction. Cited 06.05.2017,
https://en.wikipedia.org/wiki/Hardware_abstraction

Human Machine Interface. Cited 28.04.2017,
https://en.wikipedia.org/wiki/User_interface

Iterative and Incremental development. Cited 31.03.2017,
https://en.wikipedia.org/wiki/Iterative_and_incremental_development

Waterfall development model. Cited 31.03.2017,
https://en.wikipedia.org/wiki/Waterfall_model

Wisegeek What is Ambient Temperature? Cited 21.04.2017,
http://www.wisegeek.org/what-is-ambient-temperature.htm

Zha, P. & Khan, R. 2018. A Review Paper on DevOps: Beginning and More to Know. International Journal of Computer Applications 180(48):16-20

APPENDICES

Appendix 1 Benefits and Challenges of different outsourcing models

Benefits and Challenges of different outsourcing models Appendix 1

Outsourcing Model	Benefits	Challenges
Offshore	• Most cost effective when requirements are well defined and for predictive development models • Works well with Fixed-price projects transferring cost risk from customer to supplier • Allows customer to let supplier handle non-core activities such as SW maintenance, support services etc. while utilising own resources for core activities and new critical development • Low customer worries regarding competence and resource management • Enables component or plug-in development of larger SW product with modular architecture • Data security and access management is	• Not suitable for adaptive development model because geographical separation of supplier and customer teams • The infrastructure investments related to setting up development and test environment according to customer needs are usually higher due to travel, material logistics, custom duties and support needed from customer side etc. • Customer focus and investments are needed also to facilitate proper communication channels (communication links between sites, tools and licenses etc.) • Language and cultural barriers prominent between supplier and customer teams • Challenges for customer to ensure the quality control

	not much of an issue for customer as most supplier resources are located in supplier's own premises	
Nearshore	• Most cost effective when requirements are well defined and for predictive development models • Works well with Fixed-price projects transferring cost risk from customer to supplier • Allows customer to let supplier handle non-core activities such as SW maintenance, support services etc. while utilising own resources for core activities and new critical development • Low customer worries regarding competence and resource management • Enables component or plug-in development of larger SW product with modular architecture	• Typically lower cost savings for customer as compared to offshore development • Not suitable for adaptive development model because geographical separation of supplier and customer teams • The infrastructure investments related to setting up development and test environment according to customer needs are usually higher due to travel, material logistics, custom duties and support needed from customer side etc. The savings done in traveling costs as compared to offshore development can easily be offset by higher labour and material costs compared to offshore investment. • Challenges for customer to ensure the quality

	• Data security and access management is not much of an issue for customer as most supplier resources are located in supplier's own premises • As compared to offshore - frequent customer-supplier collaboration possible due to shorter distances involved in traveling and thinner cultural boundaries	control remains although less prominent compared to offshore development
Onshore	• Most cost effective when requirements are well defined and for predictive development models. • Works well with Fixed-price projects transferring cost risk from customer to supplier • Allows customer to let supplier handle non-core activities such as SW maintenance, support services etc. while utilising own resources for core activities and new critical development	• Typically lower cost savings for customer as compared to offshore and nearshore development. Cost is often of lower priority for customer when utilising onshore development. • Less suitable for adaptive development model because separation of supplier and customer teams. In some cases, adaptive development model can be implemented with proper infrastructure investments and mutually agreed processes.

	• Low customer worries regarding competence and resource management • Enables component or plug-in development of larger SW product with modular architecture • Data security and access management is not much of an issue for customer as most supplier resources are located in supplier's own premises • Close customer-supplier collaboration possible due to vicinity of teams, less distances involved in traveling and almost non-existing cultural boundaries	
Onsite	• Suitable for both predictive and adaptive SW development. However more beneficial for adaptive development. • Suitable for consulting activities and short-term expert support from supplier	• Least cost-effective for typical development projects and especially large development projects with well-defined scope • Customer needs to ensure proper access rights to safeguard confidentiality of information and at the

	• Utilises mainly T&M contracts • Quality is enforced and followed up by customer	same time ensure that onsite resources are well-integrated with/within own teams