LUT University
School of Engineering Sciences
Degree Program in Computer Science

Rohan Durugkar

A DESIGN SCIENCE APPROACH FOR ENHANCING EFFICIENCY AT AN INDUSTRIAL AUTOMATION FIRM

Examiner and Supervisor: Associate Professor Jussi Kasurinen

Second Examiner: Professor Ajantha Dahanayake

ABSTRACT

LUT University
School of Engineering Sciences
Degree Program in Computer Science

Rohan Durugkar

A DESIGN SCIENCE APPROACH FOR ENHANCING EFFICIENCY AT AN INDUSTRIAL AUTOMATION FIRM

Master's Thesis

66 pages, 7 figures, 2 tables

Examiner and Supervisor: Associate Professor Jussi Kasurinen
Second Examiner: Professor Ajantha Dahanayake

Keywords: industrial automation, design science, case study, code generation

The thesis work described in this document was carried out over a period of six months at a leading industrial automation company located in Denmark. The company provides their customers with software and hardware needed to enable automation. The thesis work was carried out in order to discover, analyse and solve known but undocumented and unknown problems in the product lifecycle. It was evident that there existed various problems along the entire product lifecycle in the company. Interviews, meetings and workshops conducted revealed the various problems. Once the problems were documented and analysed, a solution called 'Lifecycle Tool' was designed and documented. 'Lifecycle Tool' looks at the entire production lifecycle of an industrial automation solution and acts as an additional layer. This layer facilitates structured communication, visualization and code generation. A high-level architecture of the 'Lifecycle Tool' designed in the thesis work acts as a proposed solution to certain issues faced by the firm. Also, certain lower level details like database design and software requirements which would enable the starting of the development process of the 'Lifecycle Tool' are written in this thesis.

ACKNOWLEDGEMENTS

The author would like to thank the manager at the firm where this research was carried out for all the necessary resources and the motivation which were promptly provided and the employees who were very cooperative during the interviews. The author would also like to express gratitude towards his family and close friends for the patience and support throughout.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

BDC	Business Development Consultant
CG	Code Generation
CO	Company Offerings
CP	Communications Platform
CRM	Customer Relationship Management
ECI	Employee-Customer Interaction
HMI	Human-Machine Interface
IoT	Internet of Things
IT	Information Technology
I/O	Input/Output
MVI	Machine Visualization Interface
OEM	Original Equipment Manufacturer
PLC	Programmable Logic Controller
SCR	Standard Communication Rules
XML	eXtensible Markup Language

1 INTRODUCTION

1.1 Background

The thesis work described in this document was carried out over a period of six months at a leading industrial automation company located in Denmark. The company provides their customers with software and hardware needed to enable automation on their industrial machines. The thesis work was carried out in order to discover, analyse and solve known but undocumented and unknown problems in the product lifecycle. The initial problems were presented in the inception meeting and possible solutions were discussed with an application manager and a business development consultant from the company.

As revealed in the inception meeting, the product lifecycle of developing a mechatronic deliverable to a customer had various issues. The major issue identified by the manager as well as certain employees was the inefficiency in the communication between the various individuals involved in the process of development of a mechatronic device. The process of development takes place in four stages. In the first stage, the requirements are collected from the customer. According to one of the employees, the customers do not usually have a product specification.

Once, the product requirements are understood, the company prepares an estimate of the price of the product. According to one manager, the estimated effort *is always less than* the actual work required to deliver a finished product. The estimation is made only on the basis of the amount work that will be required to implement the features. The manager noted that about 50% of extra, unaccounted overhead, comes from the time taken by the employees to understand the requirements. More overhead comes from customers who add or change requirements at later stages of the project.

Another observation was that the product requirements are never *officially* documented. An engineer who is working on one specific feature of the final product is usually unaware of the ‘bigger picture’ or the visual representation of the entire project. Also, the core requirements of the product are not represented across the various stages of product development. Certain information is lost as the project progresses through the various stages.

The development goes through various stages which initially involves mechanical engineers (customers), followed by electrical engineers and then software engineers and developers. As the terminologies vary across these domains, there is an ambiguity in the communication between engineers. These cross-domain communication problems, according to the manager and the employees result in misunderstanding of specifications and inefficiency in the development process.

1.2 Goals and delimitations

It is evident that there exist various problems along the entire product lifecycle in the company. It is beyond the scope of this thesis to produce a functioning solution to all of these problems in the time available. Hence, this thesis focusses on designing a modular solution which can be implemented in individual, functional components. A possible solution suggested by one of the managers was a code generation software. This software would take the product architecture as an input from the mechanical engineers and output a suitable software architecture and documentation. It may also output actual code.

This thesis work is aimed at investigating the actual problem in the communication by using case study as a research method. Interviews will be carried out with the most important stakeholders.

1.3 Research Methods

The underlying framework of this thesis is based on the 'Design Science in Information Systems Research' approach as described by Hevner et al. (2004). According to Silver et al. (1995), the motive of implementing information systems in an organization is to enhance the organization's efficacy and productivity. Also, the degree to which the organization's motive is ascertained is directly proportional to the competencies of the information system, features of the organization, people, systems and methodologies. (Silver et al., cited in Hevner et al. 2004). It is binding upon the people involved in information science research field to "further knowledge that aids in the productive application of information technology to human organizations and their management" (ISR 2002, inside front cover, as cited in Hevner et al. 2004).

The design science approach facilitates the primary goal of this thesis, i.e. to solve the problem of inefficient communication in an organization. This approach was chosen it aids in the development of an IT artefact which would increase the organization's efficiency and productivity. This approach presents seven adaptive guidelines to conduct, evaluate and present research which involves the designing of an IT artefact. According to Simon (1996), the origins of design-science lie in engineering and 'the science of the artificial'. Basically, it is a model for solving problems. (cited in Hevner et al. 2004).

Hevner et al. (2004) argue that the IT artefacts which are products of design-science facilitate the problem solving and organizational capabilities of humans as they provide computational and intellectual tools. Design science, as argued by Hevner et al. (2004) gives rise to and assesses IT artefacts which are created to resolve problems recognized in an organization. These IT artefacts can be in the form of software, natural language descriptions which are not formal, logic etc. The IT artefact as a result of this thesis work will be a detailed software architecture which will be prepared after the requirements have been specified. In order to specify the requirements, interviews will be taken with the various stakeholders in the company to discover in detail the problem which leads to inefficiency.

1.3.1 The Seven Guidelines for Design Science in IS research

The seven guide lines provided by Hevner et al. (2004) are as follows,

1) Design as an Artefact
2) Problem Relevance
3) Design Evaluation
4) Research Contributions
5) Research Rigor
6) Design as a Search Process
7) Communication of Research

These seven guidelines are discussed in detail in the following section. Hevner et al. (2004) do not prescribe to follow these guidelines strictly, but recommend the practitioners to use their own discretion, skills and judgement to decide how, when and where to use them. But,

they also assert that for any design science research to be whole, all the guidelines should be addressed in some or the other manner.

1.3.1.1 First Guideline - Design as an artefact

The first guideline is, ‘design as an artefact’. Hevner et al. (2004) define the term ‘IT artefact’ from two perspectives, broad and narrow. In the broader perspective, they include in their definition, the models and methods used in the development as well. In the narrower perspective, they do not contain stakeholders and the evolution of the artefact as the time passes. According to Denning (1997) and Tsichritzis (1998), artefacts which are a result of design-science based research are seldom complete IT systems, but are improvements that specify ‘ideas, practices, technical capabilities and products’ past which the development of information systems can be achieved (cited in Hevner et al. 2004).

The instantiation of an artefact serves various purposes, it shows whether the design process of the product and the product itself are viable. According to Nunamaker (1991), it shows a ‘proof by construction’ (cited in Hevner et al. 2004). Hevner et al. (2004) argue that the development of such an artefact in an organization is the initial step and a necessary step towards its commissioning.

Hence, the development of a software architecture as a result of this thesis work will provide a ‘proof of construction’ for the system to be developed and will show that it is viable and feasible to develop such a system to solve the problems which the organization currently faces.

1.3.1.2 Second Guideline – Problem Relevance

The second guideline is, ‘problem relevance’. According to Hevner et al. (2004), development of artefacts which affect the phenomena that occur is the goal of design science. The authors assert that in order to address issues in organizations, it is important to construct technology-based artefacts, organization-based artefacts and people-based artefacts.

The solution which will be designed in this thesis would consist of the three artefacts mentioned above, where, the technology-based artefact would be the software architecture of the solution, the organization-based artefact would be a set of rules on how communication is to be done and the people-based artefact would consist of making use of the human resource to update the information etc.

Hevner et al. (2004) formally define a problem as "the differences between a goal state and the current state of the system". The relevance of a problem arises from the actions taken to remove the differences between the goal state and the current state of a system.

1.3.1.3 Third Guideline – Design Evaluation

It is essential to showcase the quality and efficiency of a design artefact and this can be achieved through evaluation methods. Evaluation is a very important part of the entire research process. The evaluation of an artefact is based upon the requirements set by the business environment. In design science, the artefact designed incrementally affects the business environment. As the design process progresses, more artefacts come into play and evaluation is based on how these artefacts affect the environment (Hevner et al, 2004).

According to Hevner et al. (2004), the evaluation of an IT artefact can be done in terms of attributes like usability, completeness, consistency, performance, accuracy, reliability etc. Design is inherently iterative and this implies that the evaluation phase provides crucial feedback to the construction phase and a design artefact is said to be complete as soon as it address the problem it was meant to solve. Design evaluation methods consist of observational (case study, field study), analytical (static analysis, architecture analysis, optimization and dynamic analysis), experimental (controlled experiment, simulation), testing (functional, structural) and descriptive (informed argument, scenarios) methods.

The evaluation process is out of scope of this thesis work as a functional software artefact will not be programmed because of time constraints. But it is certain that as the solution will be developed, evaluation will pay a large role in the formation of the system (Hevner et al. 2004).

1.3.1.4 Fourth Guideline – Research Contributions

The obvious outcome of a design science research work is an explicit contribution of one or more than three types, namely,

1) The design artefact: The main responsibility of the artefact is to solve the problem it was meant to. The artefact may also append more knowledge to the existing knowledge base.
2) Foundations: Extension and improvement of the existing foundation by innovative construction of new methods, models and constructs can also be a major contribution of design research.
3) Methodologies: This involves the innovative development in the design and usage of evaluation methods.

As mentioned earlier, the outcome of this thesis work is the software architecture artefact. In the future, other research contribution of this project will be new foundations and methodologies (Hevner et al. 2004).

1.3.1.5 Fifth Guideline – Research Rigor

Rigor deals with how the research is performed. In both, construction and evaluation of a designed artefact, rigor is of utmost importance. But it is important to remember that focussing too much on rigor may lead to hindered relevance. Efficient use of knowledge bases is the cornerstone of enforcing rigor. Knowledge bases consist of research methodologies and theoretical foundations. (Hevner et al. 2004). The rigor in this thesis work is derived from the guidelines provided by Runeson and Höst (2009) to conduct a case study.

1.3.1.6 Sixth Guideline – Design as a Search Process

Hevner et al. (2004) assert that design science is an iterative process. It is seldom that a search leads to actually finding optimum results. Hence, a heuristic strategy for searching can be used. The authors define ‘design’ as a process of searching an efficient solution to a problem. Some of the most important components of design-science research are abstraction and representation of the various ways to solve a problem. Hence, design science involves

simplifying complex problems into subsets of problems, followed by approaching these 'smaller' problems. This process of decomposition tends to be detached from reality but acts as starting point. As the process iterates over various cycles, the abstractions get more and more refined and the relevance increases.

1.3.1.7 Seventh Guideline – Communication of Research

According to Hevner et al. (2004), the outcome of a design science research should be showcased to both management-based stakeholders and technology-based stakeholders. As a requirement, the technology-based stakeholders need enough information so that they can actually implement the designed artefact. They also need to know how the design prepared was and the reasons behind it. For example, this thesis work describes in detail how a database should be designed – its entries, interface and connections.

For management-based stakeholders, it is not necessary to focus on the technical details of the artefact but on how would the artefact be applied in the organizational environment. Such information should be presented in a summarized way and an easy to understand manner. This thesis work explains how the artefact would be applied in the organization and the future work will include training manuals etc.

1.4 Structure of the Thesis

Section 2 contains related research, which takes a brief look on similar design science (subsection 2.1) and case study research (subsection 2.2), section 3 contains a description of case studies. It describes how the case study was conducted, its objectives and outcomes. Subsection 3.2 contains data from all the interviews and workshops conducted. Subsection 3.3 contains the qualitative data analysis of the interviews and the workshops. Information is coded and arranged according to the various codes in the subsections.

Section 4 is named 'Designing the Solution' and shows the design process of the software architecture. Subsection 4.1 summarizes the problems and proposed solutions from the interviews and subsection 4.2 describes in detail, the actual solution.

Section 5 contains the results, section 6 contains the discussion and conclusion and section 7 contains the summary of the thesis.

2 RELATED RESEARCH

2.1 Design Science

2.1.1 'A Representational Scheme for Analysing Information Technology and Organizational Dependency'

The paper demonstrates a new methodology for representation of important elements which govern relationships within an organization and how these relations can be captured, communicated and evaluated under dynamic situations. The new methodology is called as 'dependency network diagrams'. The research was presented and applied in the Canadian vehicle repair industry (Tillquist et al, 2002).

The authors demonstrated 'problem relevance' by asserting that it is of utmost importance to take into consideration the organizational interaction and process coordination so that operative information systems can be built. Their research was primarily focused on including dependence relations within the organization and outside the organization in the conceptual model. 'Research rigor' was derived from the well-known 'Organizational Behaviour' theory by Stern, Pfeffer and Salancik (1979). The authors did not compare their proposed 'dependency network diagrams' with other techniques used to model information systems. If design is to be used as a 'search process', it is important to compare proposed techniques with already existing techniques (Hevner et al, 2004).

As mentioned before, a model for conceptual representation of interactions was designed by the authors. This was the artefact generated in this research and it was named 'dependence network diagrams'. The design was evaluated in a case study conducted in the Canadian vehicle repair industry. It was not clarified as to why did the study take place in this particular industry and whether or not, similar kind of insights would have been generated if other techniques were used.

2.1.2 'Controlling Prototype Development through Risk Analysis'

The authors in this research demonstrate a fresh approach on the managing of evolutionary prototyping. According to the authors, the major, unaddressed problem was the trouble in governing projects which involve prototyping (Baskerville & Stage, 1996).

The problem relevance was derived from the fact that the management of projects was a major problem for managers in IT, developers and customers. According to the authors, novel methods were required to efficiently check resources and expenditures allotted to projects. The authors did not demonstrate research rigor as they did not present a formal approach for conducting the design. Different artefacts were generated as a result of the research, the first being a risk mitigation model and the second being, a method for management. Evaluation was not explicitly conducted by comparing the proposed methods to existing methods.

2.1.3 'Anonymous Mechanisms in Group Decision Support Systems Communication'

The authors in this research look at the problems related how anonymity can be achieved in group decision support systems (Gavish & Gerdes, 1998). Extensive research has been conducted on group decision support systems and this is one of the testimonies to the problem relevance. Anonymity is increasingly anticipated in such systems. The research rigor in this research work was derived from existing research in cryptography and secure network communication protocols based on the work done by Chaum (1981) and Schneier (1996).

This research was conducted through a thorough analysis of existing encryption and secure transmission protocols. This contributed to design being a search process. The authors designed five mechanisms to confirm anonymity and used them to implement procedural artefacts and communication protocols. For design evaluation, the authors used formal proof methods and they also conducted a comprehensive cost-benefit analysis. As mentioned earlier, the research contributions of this work were methods to deliver anonymity in group decision support systems.

2.2 Case Study

2.2.1 Introduction of eXtreme Programming in a New Context

Runeson (2012) mentions the application of case study in three application domains – automation, defence and telecom companies, namely, ABB, Ericsson and Vodafone. eXtreme programming (Beck, 1999) was applied to project management in these case studies.

The rationale for this case study was the questions which arose at an industrial seminar on eXtreme programming and Agile development methods. Stakeholders were interested in knowing if or how would Agile development techniques co-occur with their present development techniques of stage-gate project management (Cooper, 1993). According to Cooper (1993), stage-gate project management techniques have a waterfall model. The objective of this study was to extend the knowledge of the use of eXtreme programming in software development done on a large scale. The aim of this study was be quoted as, "The study aims to investigate the effects of introducing an agile software development process within a traditional stage gate model style project management system. The study specifically aims to identify the main problems with this combined approach as well as key success factors".

Interviews were the main source of obtaining information and data collection. These interviews were designed to be semi-structured, which took the form of discussions. A total of twenty interviews were conducted. For the data analysis, transcribed quotes from the interviews were coded and grouped. Six categories of analysis were identified as a result of the data analysis.

2.2.2 Identification of Test Processes and Improving Them

Runeson (2012) reports a case study on the management of quality in a development environment using iterative methods. According to Runeson (2012) this study had a flexible design and chiefly made use of quantitative data. The objective of the study was identification of the test process in a firm and enhance it. Also, the other objective was to

mimic two other studies which were conducted in a similar context. The research questions were derived from the two objectives mentioned.

Data collection was primarily done by making use of the databases provided by the firm and interviews were used as a secondary source of information. An approach similar to the Boehm's spiral model (Boehm, 1988) was used in the case study and hence, data analysis was carried out in every cycle of the case study. As a result, discoveries from the case study were reported to the stakeholders continuously.

3 CASE STUDY

3.1 Case Studies

This case study was conducted on the basis of the guidelines provided by Runeson and Höst (2009). According to the Runeson and Höst, case study is a fitting approach for research in software engineering because it *"studies contemporary phenomena in its natural context".* As this thesis work has the primary goal of studying the problem of inefficient communication between stakeholders at an industrial automation company, and the thesis assumes that a software needs to be developed to address the problem, the case study methodology is apt. The case study would reveal the intricacies of the problem and the stakeholder expectations for the software solution which has to be developed. Software architecture will be generated as a result of the case study.

Based on the classification made by Robson in 2002, Runeson and Höst mention four types of intentions for research, namely: exploratory, descriptive, explanatory and improving. Exploratory research is aimed at discovering the occurrence of a phenomenon, knowledge on that occurrence and making possible hypotheses for research. Descriptive research depicts an occurrence whereas explanatory research aims at discovering the explanation for an occurrence. Improving research aims at making better a part of certain phenomena. (Robson, cited in Runeson & Höst, 2009)

This thesis work is primarily a descriptive study as it tries to portray the problem of inefficient communication in detail. The study is also an explanatory study as the aim is to explain the problem before proposing possible solutions.

3.2 Interviews & Workshops

Five planned interviews, one open interview and one employee-customer meeting (workshop) were conducted and are documented in this thesis work. The set of questions asked in the planned interviews were as follows,

1) “We were told that the current development process is divided into three parts, namely, meeting with the client, meeting with engineers and implementation. Where do you fit in the product development process?”
2) “What does a (employee position, for example, engineer) do?”
3) “What does the product development process look like?”
4) “What are some recurring issues that you have encountered in the development process?”
5) “Can you see any solution to these issues?”

Question 1 aims at discovering the position of the interviewee in development process. This would help in understanding what issues lie in which stage of the development process. Question 2 aims at understanding the job title of the interviewee and the various roles she plays in the company. Question 3 aims at thoroughly understanding the development process of an industrial automation solution – from meeting with the customer to the commissioning on the machine. Question 4 collects the various issues and that occur at the different stages of the development and Question 5 collects the opinions of the employees, as to what would solve those problems.

The interviews were conducted with two other master’s degree students from Denmark. One asked the question to the employees, and two took notes. The notes were later compared for inconsistencies in interpretation, biases and missing details. No audio or video recording was done during the interviews. The names of the interviewees have not been included in the thesis for confidentiality and privacy.

3.2.1 First Interview

The goal of this interview was to discover the current product development process within the organization and the various problems that occur. The interviewee was a ‘Business Development Consultant’. He was an industrial electrician before joining the organization and is doing a master’s in business administration. He was a marine engineer and a technical manager with various companies.

He has worked for 21 months in the organization and has been a part of three major projects, including multiple minor projects. He was also involved with various IoT (Internet of Things) projects and is currently working on data collection projects. Within the organization, he is responsible for strategy, organization design, human resource work, and marketing.

These were the set of questions asked followed by his answers:

1) Question: "We were told that the current development process is divided into three parts, namely, meeting with the client, meeting with engineers and implementation. Where do you fit in the product development process?"
   Answer: "I handle customer relations and look for potential customers."

2) Question: "What does a Business Development Consultant (BDC) do?"
   Answer: "A BDC looks for new technologies for the organization and new technologies for the customers. He can also act as a technical sales backup if needed. A BDC helps to optimize internal processes and assists the customers with new updates and technologies. They are also involved in the development of scripts and smart tools, software robots for administrative work etc."

3) Question: "What does the product development process look like?"
   Answer: "The process depends on the type of the customer; whether they are an existing or a new customer. The first meeting involves brainstorming and idea generation. We break down the information gained in the meeting into hardware and software requirements. Five to six meetings happen before the concept is approved."

   "In the initial phase of meetings/workshops with the customer, pain points and challenges are shared. We tell the customer what is possible with the current set of technologies we have."

   "A proof of concept is made and approved in the second phase. The solution development starts in the final phase."

4) Question: “What are some recurring issues that you have encountered in the development process?”
   Answer: “There is no common language when different engineers communicate with each other. For example, a mechanical engineer (customer) speaks a language which is difficult to understand by a software (application) engineer. Also, different mind-sets throughout the process makes it difficult to be consistent. We have to adapt to the customers’ different domains and expertise. Each employee within the organization has their own unique language.”

   “Another issue is that the customers need greater clarification as to what the company has to offer. The company is not a product supplier, but it is a service provider which also provides products.”

   “A CRM system is used to keep the information updated but the use of this is unknown. It is mostly used for new customers and not current customers.”

   “Communication between stakeholders happens through email and it lacks culture and facial expressions. It is possible that the written word is misunderstood. This type is communication is not prioritised. Also, information exchanged through phone calls and face to face meetings is difficult to ‘save’.”

   “There are not enough resources in the company. We need more people.”

   “The culmination of the above-mentioned issues contributes to a harder kick-off briefing to the colleagues who are suddenly joining the product development process which might have been ongoing from the past six to twelve months. This means that they have to absorb a colossal amount of information and filter through everything.”

5) Question: “Can you see any solution to these issues?”
   Answer: “One solution can be to bring the A-Team when interviewing the customer in order to cover all the bases. Another solution can be to have a unified data collection system.”

3.2.2 Second Interview

This interviewee was also a 'Business Development Consultant'. He started as an information technology engineer before joining the organization learning Java, C#. He was involved in generating network topologies and automation solutions. He worked as a machine builder, writing programs and creating the IT infrastructure for industrial laundry machines.

These were the set of questions asked followed by his answers:

1) Question: "We were told that the current development process is divided into three parts, namely, meeting with the client, meeting with engineers and implementation. Where do you fit in the product development process?"
   Answer: "My work is mainly focussed on software, I also design the software architecture."

2) Question: "What does the product development process look like?"
   Answer: "Customers meet at fairs, through hearsay or networking. The company sees a potential customer and then they have a meeting. Customers want to see how their processes can be improved."

   "Drawings are made on whiteboards and specifications are discussed at workshops. The goal of such workshops is that everyone is 'synchronized'. Customers are also a part of the workshops."

   "The usual product development process looks like this, 'research and development followed by production and then maintenance.'"

3) Question: "What are some recurring issues that you have encountered in the development process?"
   Answer: "Other engineers (mechanical engineers who are customers) have no clue what software is. They can design the machine, know the flow and sequence of the machine. An ideal person is a mechanical engineer who can program."

"A few years ago, it was easier for mechanical engineers in the automation industry. Then, PLCs (programmable logic controllers) were introduced and were initially used for electronic purposes. It was visual programming in the 80s and the 90s and was done using Ladder. Ladder does not support complex programs and programming should be done by programmers and software engineers and not electricians or mechanical engineers."

"In most of the cases, software engineers get a mechanical design and are expected to make it work."

"The real game changer in the machine industry is software and that is where all the difference lies. This is not so obviously visible to the customers."

"Programmers are often only used for bug fixing."

4) Question: "Can you see any solution to these issues?"
   Answer: "Two programmers, one programs on the simulator and the other puts it on the machine and then they bring in the mechanical engineer. The pros of this method are that it works very well, and the design decisions are quickly discussed. The con is that everyone needs to be near the machine physically and this is not always possible."

   "Having a common interface for everyone where we can see how the machine works, physically."

   "Finding out whether the mechanical engineers have modularized their systems or not. It is helpful if the machine has already been represented as modules."
   "Receiving sequence diagrams of the machine from the customer in order to know the sequences while programming."

   "It would be interesting to know how mechanical engineers see how a machine works. Do they think in states?"

"We need a structured way of communicating with mechanical engineers."

3.2.3 Third Interview

The interviewee was a 'Sales Manager'. He is a trained electrician and has a bachelor's degree in power engineering and business administration. He was a marine engineer and a technical manager with various companies. He has a working experience of 13 years in the industry.

These were the set of questions asked followed by his answers:

1) Question: "We were told that the current development process is divided into three parts, namely, meeting with the client, meeting with engineers and implementation. Where do you fit in the product development process?"
   Answer: "I'm responsible for the sales as my team acquires the customers."

2) Question: "What are some recurring issues that you have encountered in the development process?"
   Answer: "Today's mix of employees looks like that of 1975 but the value of software in machines is increasing. There are certain issues when we speak with customers. The software developers come in very late and the sales people make promises to the customers which are too big."

   "The challenge to our company is to convince the requirement of automation to the customers, focussing on the fact that software and technology are taking over."

   "Our strategy is to convince the top management in OEM companies."

   "Minimizing the total cost of ownership is the key driving factor for us. But what drives a mechanical engineer (customer)?"

3.2.4 Fourth Interview

This interview was carried out with an application engineer. Application engineers are responsible for the actual development of a project. They are responsible for writing the code amongst other tasks. The interviewee was an experienced application engineer. The goals of this interview were to understand,

1) What information do application engineers need in the development phase; for which they need to go back to the stakeholders in the initial phases of the project?
2) Can this information can be captured and reused?
3) What causes delays in the implementation process?

These were the set of questions asked followed by his answers:

1) Question: "We found out from the previous interviews that 'programming cannot start at the initiation of the implementation of a project'. In your opinion, why do you think this occurs?"

   Answer: "The proof of concept made in the initial stages is usually a 'quick' version of the project. I need to understand the machine first. The actual movements of the machine need to be translated into software. I also need to understand the sub-parts of the machine and then make the architecture. I need to see if there are any reusable software components. I must understand the machine completely and see how the customer wants it. Amongst the stakeholders from the initial stages, it is the customer who knows more about the machine."

   "Before implementing the specific details, I need to get the complete, big picture of the machine. It is required to visualize often. The best way to start is to actually observe the machine. Sometimes, there is no machine available."

   "Customers sometimes bug fix the old code and this fact is not mentioned in the specifications."

"It is important to obtain as many cases and specifications in the beginning because often, various use cases appear after commissioning, and then the engineers have to go back in."

"Customers start specifying details about the machine by discussing the faults in the machine, they believe that it shouldn't be hard for engineers to fix these issues."

2) Question: "The programming phase takes 64 to 128 hours, whereas, ideally, it should take 4 to 8 hours. In your opinion, why is it so?"
   Answer: "We spend 40 to 50% of our time on understanding how and what to program. It varies from project to project but the logic can be difficult and complex sometimes. It also depends on the number of I/Os."

   "If there are multiple machines, then there can be multiple use cases and that results in more code. If there are multiple customers within a project, the instances of errors are increased."

   "Sometimes it is required to 're-specify the specifications' which consumes a lot of time. Sometimes, state machines need to be drawn."

   "We do not even get written specifications sometimes. We receive a user manual of the machine which doesn't have any specifications at all."

   "Coming up with algorithms or calculation logic, reverse engineering and trial and error is quite time consuming as these are customer and machine specific."

   "Changing requirements from customers happens very often, new requirements arise in the later stages of development. Similarly, end customer requirements also arise. Sometimes, customers' competitors may make something new and then, new requirements arise for us."

3) Question: "What so you think can be done so that the time to implement is reduced?"

Answer: “Obtaining pseudocode or state machines before starting the implementation would be helpful in reducing the time to implement.”

4) Question: “Can you list some questions and doubts which arise when you go back to the stakeholders from the initial phases of the project?”
   Answer: “Question about the functionality of the machine, gear ratios, mechanical engineering related questions etc.”

3.2.5 Fifth Interview

This interview was also carried out with an application engineer. The interviewee has been working on developing new concepts and standards. The goals of this interview were also to understand,

1) What information do application engineers need in the development phase; for which they need to go back to the stakeholders in the initial phases of the project?
2) Can this information can be captured and reused?
3) What causes delays in the implementation process?

These were the set of questions asked followed by his answers:

1) Question: “We found out from the previous interviews that ‘programming cannot start at the initiation of the implementation of a project’. In your opinion, why do you think this occurs?”
   Answer: “What the customer says, and the contents of the specification document are completely different sometimes. In such cases we need to re-specify the specifications and that takes a lot of time as the customer needs to be contacted again.”

   “The specifications are misinterpreted sometimes and then the decisions are made by me based on my experience. Such decisions may turn out to be wrong sometimes.”

   “Once the project is understood, the ‘shell’ of the project needs to be created.”

2) Question: “The programming phase takes 64 to 128 hours, whereas, ideally, it should take 4 to 8 hours. In your opinion, why is it so?”
   Answer: “Specifications change constantly, we need to redo many things as the customer gets new ideas or there have been misunderstandings, internally or externally.”

   “I try to be modular while writing code, but the machine has been coded in a ‘spaghetti code’ manner and then all coding needs to start from scratch. Also, coding standards are not followed, and this delays the process.”

3) Question: “What so you think can be done so that the time to implement is reduced?”
   Answer: “Auto generated templates for coding which contain a shell of the project, libraries and coding standards would be helpful.”

   “Better communication tools to obtain specifications form the customer might also be helpful.”

   “A visual tool could be helpful for the engineers to understand the project and for the customers to specify the requirements.”

4) Question: “Can you list some questions and doubts which arise when you go back to the stakeholders from the initial phases of the project?”
   Answer: “Sometimes, I need to go back to the Business Development Consultant to ask him which way of implementation he prefers. This needs to be done because he was the one who started the conversation with the customer.”

   “50% of the functionality is usually not specified by the customer. 25% can be guessed and I need to go back to the customer to ask the rest.”

   “I would prefer to read specifications and not write them.”

3.2.6 Sixth Interview

This interview was also carried out with an application engineer. He is also an experienced application engineer and also has experience as a business development consultant. The goals of this interview were also to understand,

1) What information do application engineers need in the development phase; for which they need to go back to the stakeholders in the initial phases of the project?
2) Can this information can be captured and reused?
3) What causes delays in the implementation process?

These were the set of questions asked followed by his answers:

1) Question: "We found out from the previous interviews that 'programming cannot start at the initiation of the implementation of a project'. In your opinion, why do you think this occurs?"
   Answer: "Sometimes, the information we receive is not specific enough and sometimes, it is extremely specific. Information is lost as it passes through the hierarchy in the company."

   "In my opinion, it is impossible to specify all the requirements in the phases before development starts."

2) Question: "The programming phase takes 64 to 128 hours, whereas, ideally, it should take 4 to 8 hours. In your opinion, why is it so?"
   Answer: "The customers often don't know what they exactly want."

   "The tasks which are listed in the Kanban board are just about a third of the actual requirements."

   "The information transfer between the business development consultant and the application engineers is lossy."

3) Question: "What so you think can be done so that the time to implement is reduced?"

Answer: “Starting up of the coding correctly with proper inputs from stakeholders in the previous stages would save time, for example, having libraries, communication standards, methodology, architecture etc. Also, we need to be able to generate code from packML diagrams.”

“State diagrams would be amazing to have before starting to write the code.”

4) Question: “Can you list some questions and doubts which arise when you go back to the stakeholders from the initial phases of the project?”
   Answer: “We need to go back to the stakeholders from the company to get opinions or ideas on how the project should be implemented and also to ask about the design or hardware choices.”

   “We also need to go back to the customer to obtain the possible states of the machine and other specifications.”

3.2.7 Seventh Interview – Open Interview

This open interview was carried out with a Business Development Consultant and an Application Manager. A specific set of questions was not used for this interview as the interviewees were from a higher level in the management and had an overview of the entire process and the underlying issues within it. Nevertheless, the theme of the interview was similar to the other interviews – looking at the development process followed by discussing the issues in the development process and then discussing how these issues can be solved. The informal discussion is documented in the following paragraphs. It is grouped in a way such that the context remains uniform throughout the paragraph.

“We lose much of the information until we reach the development phase in the project. This leads to a delay in the initiation of the implementation of the project.”

“Maybe, data could be formally transferred from the initial phases to the development phase. How much information do we gain in the initial stages that can be used in the development

stage? What does it take to implement a solution? We do not have a proper project implementation start-up meeting. We need to capture this data in a nicer way."

"As a solution, a tool would be used in the stage before development where it would facilitate the winning of a customer and save as many project specifications at the same time. Maybe, there is only one chance to win a customer. The more detailed the customer demo is, the more trust we can gain from them. We need to reduce the time taken for the implementation of software from 64-128 hours to 4-8 hours. Software needs to come in the earlier phases of the product lifecycle. It would help to implement a 'project start-up' package or a system and then give it to the developers."

"Right now, we have a tool which converts an XML proof of concept into a folder hierarchy. No modules are generated."

3.2.8 Customer Meeting - An Example

This customer meeting was held at the customer's company. There were two representatives from the host company (customer) and four representatives from the visiting company (the company on which this thesis is based on). The goal of this meeting was to 'gather enough information to make an initial estimate on the work needed to convert the machine hardware and software to that of the visiting company, also, to identify the major risks.' The agenda of the meeting comprised of the following topics,

1) Background and history of the host company
2) The mechanical design of the machine to worked upon
3) Major functions of the machine
4) Complex processes
5) Current hardware (I/O, motors, drives)

The following observations were made,

1) A manager from the visiting company wanted to understand the flow of material in the machine. He was trying to set a common language comprising of terms and

phrases to describe the machine and the processes. He would present them and verify from the host company if they were correct. By doing so, he was trying to unify and create a set of common words, so that everyone understood each other.

2) The visiting company spoke in the terminology of mechanical engineers as they were experienced in conducting such meetings.
3) The visiting company tried to explain the advantages of software to the host company, but they were hesitant or unwilling to adopt new technologies.
4) An engineer from the visiting company made a list of motors on the white board by using inputs from the host company. The gear ratios of each of the motors were listed. (see figure 1). The engineers then tried to understand what the function of each of the motors was. The motors were given names ‘on-the-fly’.
5) Information from this meeting was not being recorded exhaustively. Employees from the visiting company were taking notes on their computers. The notes from one of the employees can be seen in figure 2 and 3. (personally identifiable information and company names have been redacted for confidentiality purposes)
6) Employees from the visiting company said that the mechanical units would be converted in packML. The hierarchy, function blocks and gear ratios will be converted to code. The interacting modules will be converted to software architecture.

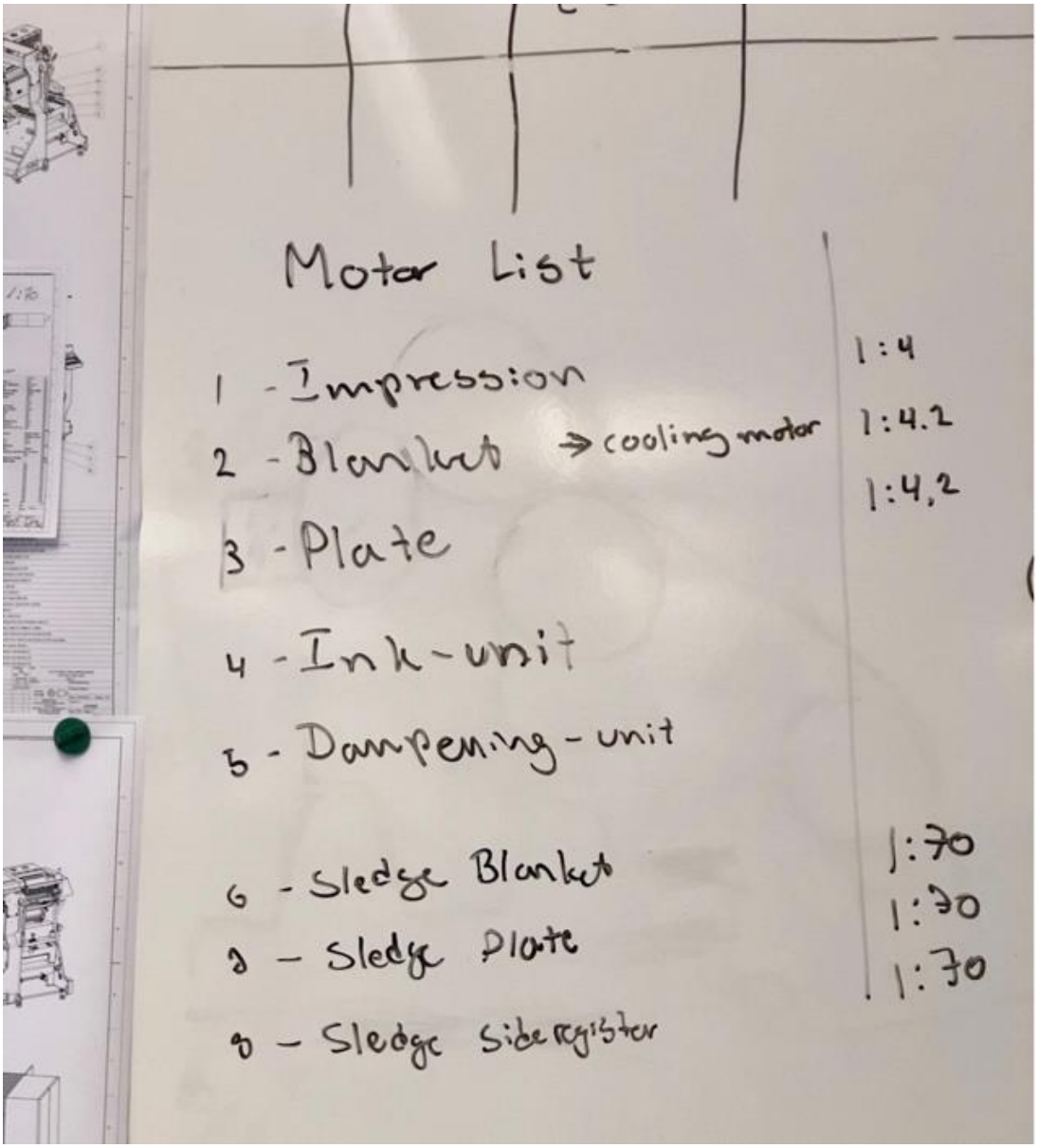


Fig. 1. Motor list and gear ratios listed on a whiteboard during a customer meeting

Hi,

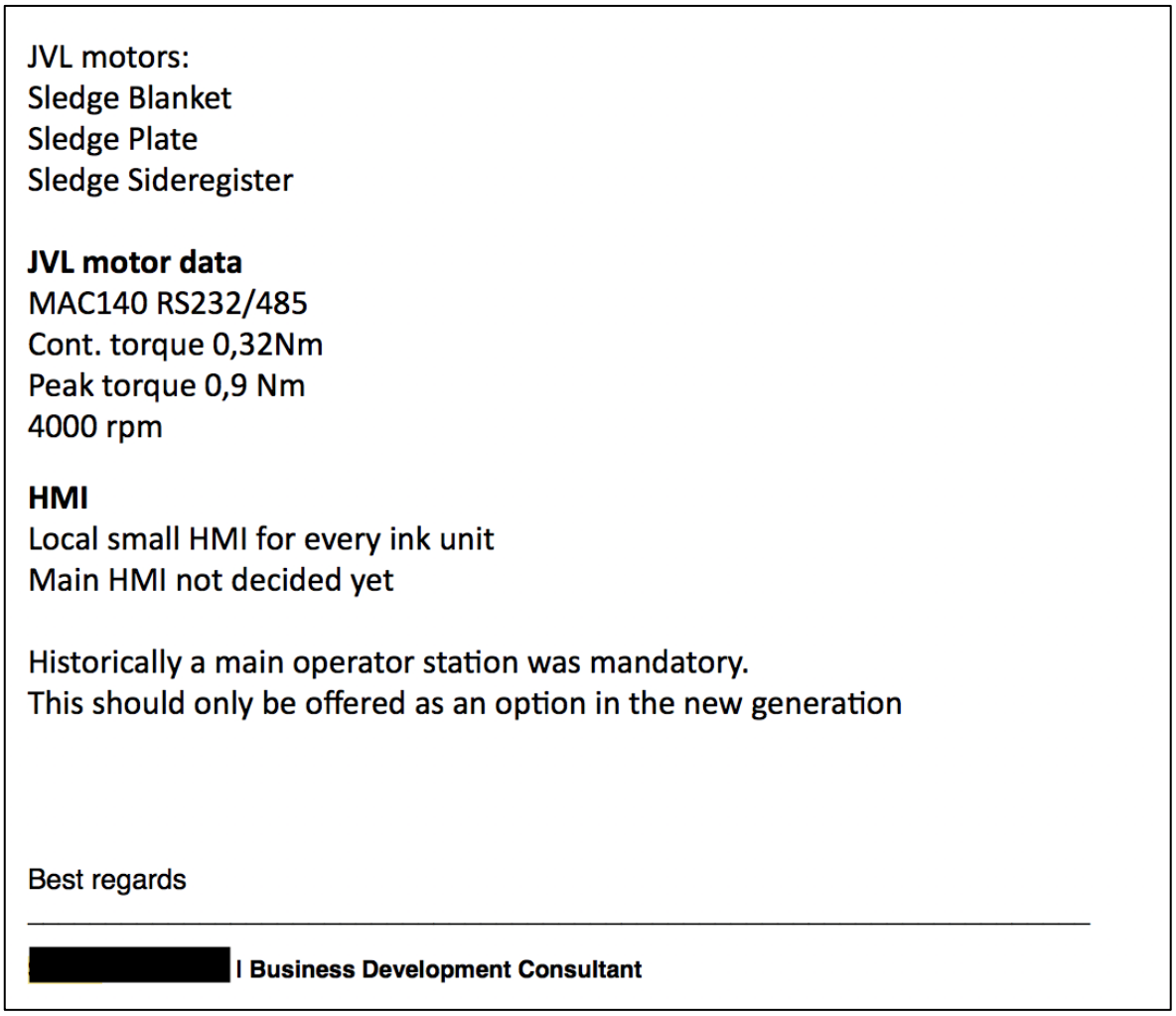
This is the notes I collected today:

Specifications
Webspeed 200m/min.
Webtension 80-300N
Ink motor needs to be increased +25% compared to the current motor
Blanket motor is very closed to limit in some jobs
Impression is comparable to flexo technology

The current machine is based on [redacted] drives

Motor list
Impression
[redacted]
Blanket (with cooling)
[redacted]
Plate
[redacted]
Ink unit (perhaps one additional motor)
Not motor data ready yet
Dampening unit
No motor date ready yet

Fig. 2. Notes collected by an interviewee at a customer meeting


JVL motors:
Sledge Blanket
Sledge Plate
Sledge Sideregister

JVL motor data
MAC140 RS232/485
Cont. torque 0,32Nm
Peak torque 0,9 Nm
4000 rpm

HMI
Local small HMI for every ink unit
Main HMI not decided yet

Historically a main operator station was mandatory.
This should only be offered as an option in the new generation

Best regards

[redacted] I Business Development Consultant

Fig. 3. Notes collected by an interviewee at a customer meeting

3.3 Qualitative Data Analysis of the Interviews and Observations

According to Seaman (1999), qualitative data analysis methods are generally used in case studies as case studies are flexible (cited in Runeson and Host, 2009). The fundamental goal of the analysis is to arrive at conclusions and maintaining a visible series of evidences. The analysis of the interviews and observations conducted in this thesis work will serve two purposes, the first being the confirmation of the hypothesis of the problem that occurs in the organization and the second being the generation of requirements for the software architecture to be developed in order to solve the problem.

The following steps will be used to analyse the interviews and observations. They are based on the steps depicted in Runeson and Host (2009).

1) Coding – The data is coded into parts of text which are contextually similar based on common themes, areas, constructs etc. There can be a many to many relationship between codes and text. The codes may also lead to hierarchical structures with sub-codes. “Memos” are the researcher’s comments and views and they can also be a part of the coded texts.
2) Identification of Hypothesis – The next step involves the going through the material to obtain an initial set of hypothesis which can, for example, include common of similar phrases in the different data locations, patterns and differences etc. Meanwhile, generalizations can be composed and they may result in a ‘formalized body of knowledge’.

These steps are often used iteratively and simultaneously when the data is being collected. The end result of such an analysis is a body of knowledge and in the case of this thesis work, a set of software requirements which will address the problem of stakeholder communication in an organization. Tabulation of the data is often helpful in order to have an overview of the data collected and analysed.

3.3.1 Coding of the Information

These steps are often used iteratively and simultaneously when the data is being collected. The end result of such an analysis is a body of knowledge and in the case of this thesis work, a set of software requirements which will address the problem of stakeholder communication in an organization. Tabulation of the data is often helpful in order to have an overview of the data collected and analysed. The data from the interviews is assigned to eight different nodes based on the dominant context. The seven nodes are,

1) Capturing information
2) Different mind-sets
3) No common language
4) Difficult kick-off of development
5) Changing specifications
6) Conveying company offerings to the customer
7) Need for more employees
8) Proposed solutions

The number of references to these nodes is shown in the following table,

Table 1. Coding references

Node	**Number of Coding References**
Capturing information	28
Different mind-sets	11
No common language	4
Difficult kick-off of development	4
Changing specifications	4
Conveying company offerings to the customer	2
Need for more employees	1
Proposed solutions	18

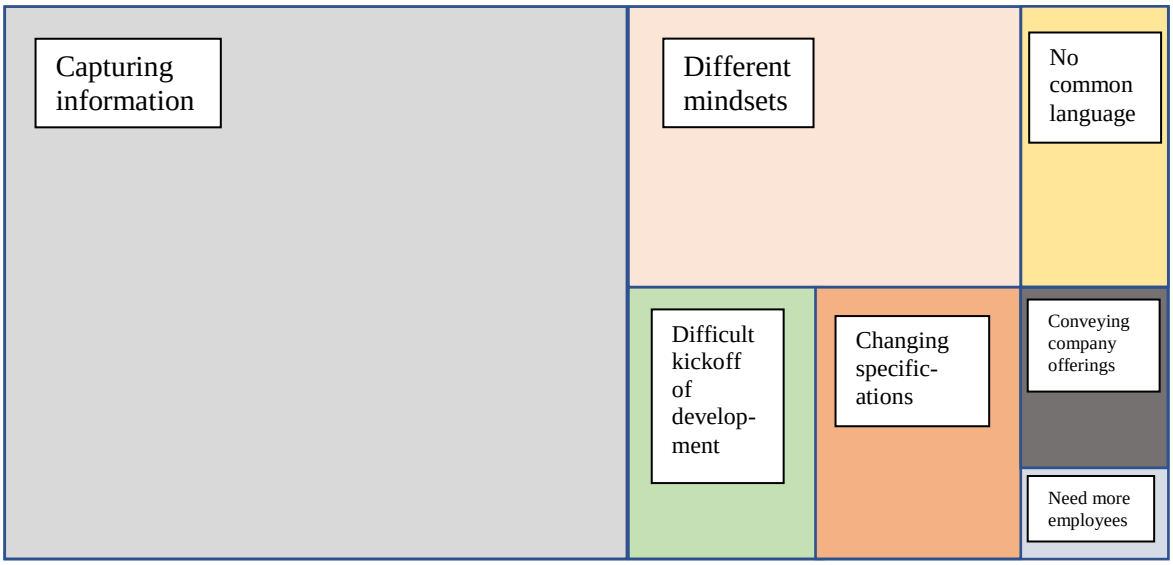


Fig. 4. Treemap of the nodes compared by coding references

3.3.1.1 Node 1: Capturing information

These are the elements from the interview referring to this node.

- Communication between stakeholders happens through email and it lacks culture and facial expressions. It is possible that the written word is misunderstood. This type of communication is not prioritized. Also, information exchanged through phone calls and face to face meetings is difficult to ‘save’.

- We need a structured way of communicating with mechanical engineers.

- I need to understand the machine first. The actual movements of the machine need to be translated into software. I also need to understand the sub-parts of the machine and then make the architecture.

- I need to see if there are any reusable software components. I must understand the machine completely and see how the customer wants it. Amongst the stakeholders from the initial stages, it is the customer who knows more about the machine.

- Before implementing the specific details, I need to get the complete, big picture of the machine. It is required to visualize often. The best way to start is to actually observe the machine. Sometimes, there is no machine available.

- Customers sometimes bug fix the old code and this fact is not mentioned in the specifications.

- It is important to obtain as many cases and specifications in the beginning because often, various use cases appear after commissioning, and then the engineers have to go back in.

- Sometimes it is required to ‘re-specify the specifications’ which consumes a lot of time.

- Sometimes, state machines need to be drawn.

- We do not even get written specifications sometimes. We receive a user manual of the machine which doesn’t have any specifications at all.
- Obtaining pseudocode or state machines before starting the implementation would be helpful in reducing the time to implement.

- What the customer says, and the contents of the specification document are completely different sometimes. In such cases we need to re-specify the specifications and that takes a lot of time as the customer needs to be contacted again.

- The specifications are misinterpreted sometimes and then the decisions are made by me based on my experience. Such decisions may turn out to be wrong sometimes.

- 50% of the functionality is usually not specified by the customer. 25% can be guessed and I need to go back to the customer to ask the rest.

- I would prefer to read specifications and not write them.

- Sometimes, the information we receive is not specific enough and sometimes, it is extremely specific.

- Information is lost as it passes through the hierarchy in the company.

- The tasks which are listed in the Kanban board are just about a third of the actual requirements.

- The information transfer between the business development consultant and the application engineers is lossy.

- We need to go back to the stakeholders from the company to get opinions or ideas on how the project should be implemented and also to ask about the design or hardware choices.

- We also need to go back to the customer to obtain the possible states of the machine and other specifications.

- We lose much of the information until we reach the development phase in the project. This leads to a delay in the initiation of the implementation of the project.

- How much information do we gain in the initial stages that can be used in the development stage?

- What does it take to implement a solution?

- We need to capture this data in a nicer way.

- An engineer from the visiting company made a list of motors on the white board by using inputs from the host company. The gear ratios of each of the motors were listed. (see figure 1) The engineers then tried to understand what the function of each of the motors was. The motors were given names ‘on-the-fly’.

- Information from this meeting was not being recorded exhaustively. Employees from the visiting company were taking notes on their computers.

3.3.1.2 Node 2: Different mind-sets

These are the elements from the interview referring to this node.

- Also, different mind-sets throughout the process makes it difficult to be consistent.

- Other engineers (mechanical engineers who are customers) have no clue what software is.

- They can design the machine, know the flow and sequence of the machine.

- In most of the cases, software engineers get a mechanical design and are expected to make it work.

- It would be interesting to know how mechanical engineers see how a machine works. Do they think in states?

- The software developers come in very late and the sales people make promises to the customers which are too big.

- Customers start specifying details about the machine by discussing the faults in the machine, they believe that it shouldn't be hard for engineers to fix these issues.

- The specifications are misinterpreted sometimes and then the decisions are made by me based on my experience. Such decisions may turn out to be wrong sometimes.

- I try to be modular while writing code, but the machine has been coded in a 'spaghetti code' manner and then all coding needs to start from scratch.

- Also, coding standards are not followed, and this delays the process.

- The visiting company tried to explain the advantages of software to the host company, but they were hesitant or unwilling to adopt new technologies.

3.3.1.3 Node 3: No common language

These are the elements from the interview referring to this node.

- There is no common language when different engineers communicate with each other. For example, a mechanical engineer (customer) speaks a language which is difficult to understand by a software (application) engineer.

- We have to adapt to the customers' different domains and expertise.

- Each employee within the organization has their own unique language.

- A manager from the visiting company wanted to understand the flow of material in the machine. He was trying to set a common language comprising of terms and phrases to describe the machine and the processes. He would present them and verify from the host company if they were correct. By doing so, he was trying to unify and create a set of common words, so that everyone understood each other.

3.3.1.4 Node 4: Difficult kick-off of development

These are the elements from the interview referring to this node.

- The culmination of the above-mentioned issues contributes to a harder kick-off briefing to the colleagues who are suddenly joining the product development process which might have been ongoing from the past six to twelve months. This means that they have to absorb a colossal amount of information and filter through everything

- We spend 40 to 50% of our time on understanding how and what to program. It varies from project to project but the logic can be difficult and complex sometimes.

- What the customer says, and the contents of the specification document are completely different sometimes. In such cases we need to re-specify the specifications and that takes a lot of time as the customer needs to be contacted again.

- We do not have a proper project implementation start-up meeting.

3.3.1.5 Node 5: Changing specifications

These are the elements from the interview referring to this node.

- Also, different mind-sets throughout the process makes it difficult to be consistent.

- Other engineers (mechanical engineers who are customers) have no clue what software is.

- They can design the machine, know the flow and sequence of the machine.

- In most of the cases, software engineers get a mechanical design and are expected to make it work.

- It would be interesting to know how mechanical engineers see how a machine works. Do they think in states?

- The software developers come in very late and the sales people make promises to the customers which are too big.

- Customers start specifying details about the machine by discussing the faults in the machine, they believe that it shouldn't be hard for engineers to fix these issues.

- The specifications are misinterpreted sometimes and then the decisions are made by me based on my experience. Such decisions may turn out to be wrong sometimes.

- I try to be modular while writing code, but the machine has been coded in a ‘spaghetti code’ manner and then all coding needs to start from scratch.

- Also, coding standards are not followed, and this delays the process.

- The visiting company tried to explain the advantages of software to the host company, but they were hesitant or unwilling to adopt new technologies.

3.3.1.6 Node 6: Conveying company offerings to the customer

These are the elements from the interview referring to this node.

- Another issue is that the customers need greater clarification as to what the company has to offer. The company is not a product supplier, but it is a service provider which also provides products.

- The real game changer in the machine industry is software and that is where all the difference lies. This is not so obviously visible to the customers.

3.3.1.7 Node 7: Need for more employees

These are the elements from the interview referring to this node.

- There are not enough resources in the company. We need more people.

3.3.1.8 Node 8: Proposed solutions

- These are the elements from the interview referring to this node.

- One solution can be to bring the A-Team when interviewing the customer in order to cover all the bases.

- Another solution can be to have a unified data collection system.

- An ideal person is a mechanical engineer who can program.

- Two programmers, one programs on the simulator and the other puts it on the machine and then they bring in the mechanical engineer. The pros of this method are that it works very well, and the design decisions are quickly discussed. The con is that everyone needs to be near the machine physically and this is not always possible.

- Having a common interface for everyone where we can see how the machine works, physically.

- Finding out whether the mechanical engineers have modularized their systems or not. It is helpful if the machine has already been represented as modules.

- We need a structured way of communicating with mechanical engineers.
- Obtaining pseudocode or state machines before starting the implementation would be helpful in reducing the time to implement.

- Auto generated templates for coding which contain a shell of the project, libraries and coding standards would be helpful.

- Better communication tools to obtain specifications form the customer might also be helpful.

- A visual tool could be helpful for the engineers to understand the project and for the customers to specify the requirements.

- Starting up of the coding correctly with proper inputs from stakeholders in the previous stages would save time, for example, having libraries, communication standards, methodology, architecture etc.

- We need to be able to generate code from packML diagrams.

- State diagrams would be amazing to have before starting to write the code.

- Maybe, data could be formally transferred from the initial phases to the development phase.

- We need to capture this data in a nicer way.

- As a solution, a tool would be used in the stage before development where it would facilitate the winning of a customer and save as many project specifications at the same time.

- It would help to implement a ‘project start-up’ package or a system and then give it to the developers.

Further coding of the ‘Proposed solutions’ node leads to seven sub nodes, namely,

1) Capturing information efficiently
2) Visualization
3) Code generation
4) Structured communication
5) Bring the ‘A-Team’ to customer meetings
6) Hiring ideal employees
7) Meeting dynamics

The number of references to these nodes is shown in the following table,

Table 2. Coding references

Node	**Number of Coding References**
Capturing information efficiently	9
Visualization	3
Code generation	2
Structured communication	2
Bring the ‘A-Team’ to customer meetings	1
Hiring ideal employees	1
Meeting dynamics	1

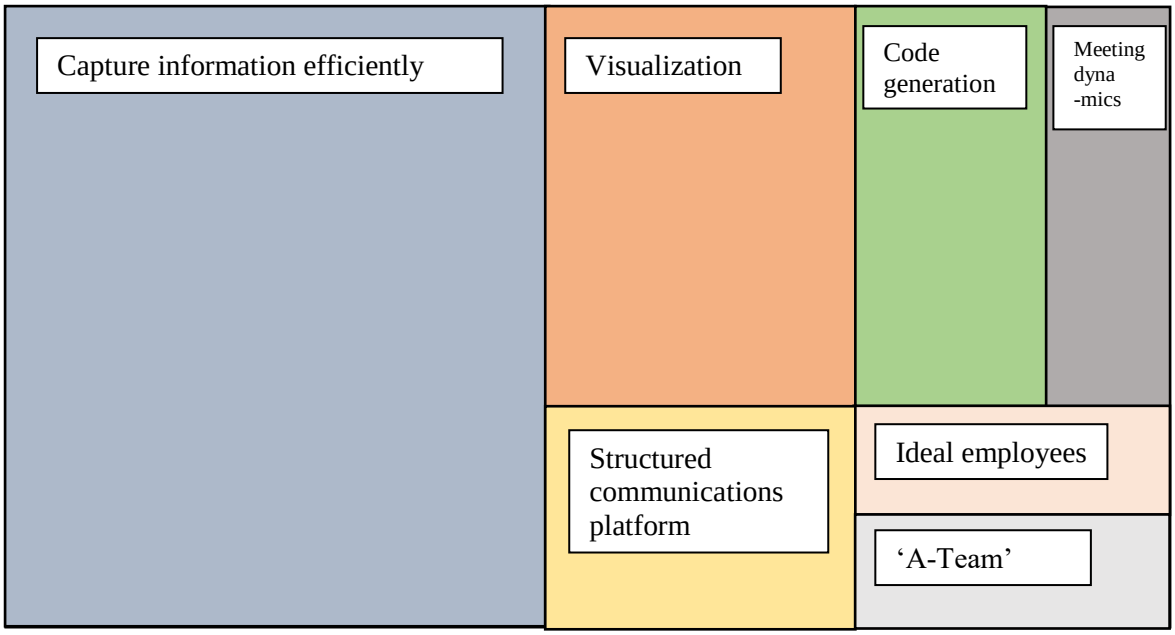


Fig. 5. Treemap of the nodes compared by coding references

3.3.1.9 Sub node 1: Capturing information efficiently

These are the elements from the interview referring to this node.

- Another solution can be to have a unified data collection system.

- Finding out whether the mechanical engineers have modularized their systems or not. It is helpful if the machine has already been represented as modules.

- Obtaining pseudocode or state machines before starting the implementation would be helpful in reducing the time to implement.

- Starting up of the coding correctly with proper inputs from stakeholders in the previous stages would save time, for example, having libraries, communication standards, methodology, architecture etc.

- State diagrams would be amazing to have before starting to write the code.

- Maybe, data could be formally transferred from the initial phases to the development phase.

- We need to capture this data in a nicer way.

- As a solution, a tool would be used in the stage before development where it would facilitate the winning of a customer and save as many project specifications at the same time.

- It would help to implement a ‘project start-up’ package or a system and then give it to the developers.

3.3.1.10 Sub node 2: Visualization

These are the elements from the interview referring to this node.

- Having a common interface for everyone where we can see how the machine works, physically.

- A visual tool could be helpful for the engineers to understand the project and for the customers to specify the requirements.

- As a solution, a tool would be used in the stage before development where it would facilitate the winning of a customer and save as many project specifications at the same time.

3.3.1.11 Sub node 3: Code generation

These are the elements from the interview referring to this node.

- Auto generated templates for coding which contain a shell of the project, libraries and coding standards would be helpful.

- We need to be able to generate code from packML diagrams.

3.3.1.12 Sub node 4: Structured communication

These are the elements from the interview referring to this node.

- We need a structured way of communicating with mechanical engineers.

- Better communication tools to obtain specifications form the customer might also be helpful.

3.3.1.13 Sub node 5: Bring the 'A-Team' to customer meetings

These are the elements from the interview referring to this node.

- One solution can be to bring the A-Team when interviewing the customer in order to cover all the bases.

3.3.1.14 Sub node 6: Hiring ideal employees

These are the elements from the interview referring to this node.

- An ideal person is a mechanical engineer who can program.

3.3.1.15 Sub node 7: Meeting dynamics

These are the elements from the interview referring to this node.

- Two programmers, one programs on the simulator and the other puts it on the machine and then they bring in the mechanical engineer. The pros of this method are that it works very well, and the design decisions are quickly discussed. The con is that everyone needs to be near the machine physically and this is not always possible.

4 DESIGNING THE SOLUTION

4.1 Addressing the problems through software

Analysis of the interviews shows the various problems which lead to inefficiency. The problems are,

1) Capturing information
2) Different mind-sets
3) No common language
4) Difficult kick-off of development
5) Changing specifications
6) Conveying company offerings to the customer
7) Need for more employees

Solutions, as brainstormed by the employees are

1) Capturing information efficiently
2) Visualization
3) Code generation
4) Structured communication
5) Bring the ‘A-Team’ to customer meetings
6) Hiring ideal employees
7) Meeting dynamics

4.2 The ‘Lifecycle Tool’ and its layers

The problems mentioned above will be solved with a layered approach, addressing problems at different levels and combining the relevant data from each layer. The three layers in the solution are,

1) The standard communication rules layer (SCR)
2) The employee-customer interaction layer (ECI)
3) The code-generation layer (CG)

The entire software tool which would encompass these layers will be called as the ‘Lifecycle Tool’. The main reason for the choice of this name is the fact that the tool would work on the entire lifecycle of the machine automation product, which will be the deliverable of the

tool. It is expected that the completion of such a software tool would greatly enhance the efficiency in the organization and would foster not only shorter delivery times but also data archiving and retrieval, analysis of working methods, greater visibility and transparency throughout etc.

The layers within the 'Lifecycle Tool' can be visualized as follows,

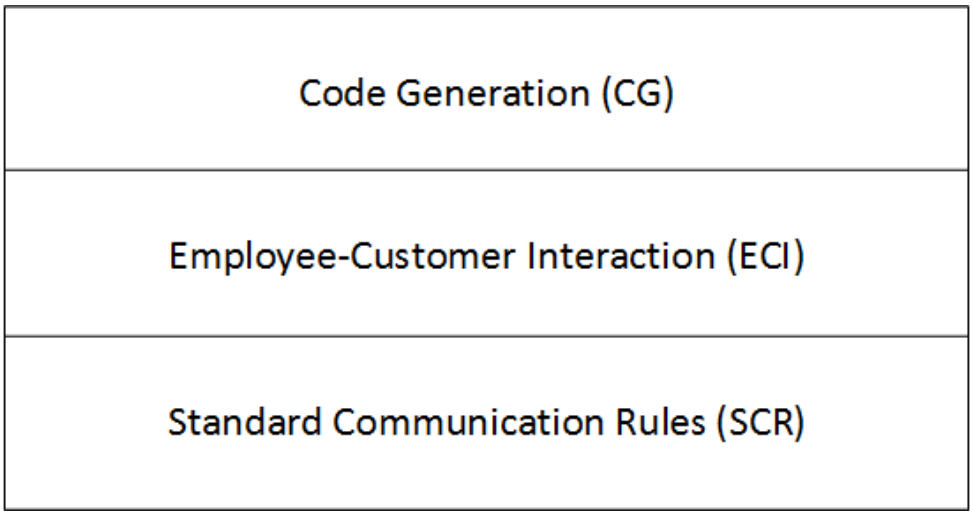


Fig. 6. Three layers of the 'Lifecycle Tool'

These layers build on top of each other, as the higher layers make use of the information in the lower layers. The SCR lays down a foundation for systematic and structured communication and is applied throughout the tool and all of its activities. The ECI makes employee-customer interaction streamlined and all the data in the various meetings and interactions is archived. The CG layer then makes use of all the information obtained in the ECI layer and automatically generates code and code templates which are used by employees and customers. The flow of information is not strictly a one way communication.

4.2.1 The standard communication rules (SCR) layer

One of the major and recurrent problem within the organization was observed to be in the communication between the employees and customers. While explaining the business requirement, customers would choose terminology from within their domain, for example, printing. Some senior employees would understand the terminologies right away and respond within context, but other employees were unable to understand what the customer

was talking about. As an effort to resolve this, employees would stop the ongoing conversation to agree upon a set of common terminologies first. As seen in figure 1, one of the employees wrote down the names of motor and units in a printing system so that everyone in the meeting agreed upon their names and had a continuous conversation.

The standard communication layer aims to build a database which would archive all the possible terminologies used in different domains. The data for such an archive can be obtained from various sources like ongoing meetings, experienced employees, documentation from customers etc. Then, it will be possible to retrieve the data with simple queries on the database, for example, before a meeting with a customer in the packaging industry, employees go through the packaging industry terminologies and possibly use and understand these terminologies in the meetings. There will be an interface where an employee would be able to look up a certain term or a concept on-the-fly in an ongoing meeting. The interface will also enable an employee to add new terms to the database so that they can be used in upcoming meetings. Terminologies can be domain-specific or customer-specific and can be retrieved accordingly.

It is also possible that the customers do not agree upon specific terminologies themselves. The SCR would aim at streamlining this issue. If one party sticks to a certain set of terminologies independent of others, everyone will eventually agree upon those terminologies and concepts.

The SCR would enforce its rules over the entire ‘Lifecycle Tool’. The terminologies will remain fixed over the entire lifecycle. This would eliminate any misunderstandings that might occur while talking about a complex mechatronic system. For example, if a certain module in a printing system was called ‘Dampening Unit’ by the customer, it will be known as ‘Dampening Unit’ throughout the product lifecycle, also in the generated code. Hence, SCR will be applied to the CG layer. Due to the complexity of certain automation projects, it is possible that human stakeholders refer to certain modules by incorrect or incomplete names at later stages in the project. This problem can be eliminated by making use of SCR. The rights to make edits to the SCR will be given to responsible employees who ensure that no unauthorized changes are made to the SCR. A git-like system would ensure that all changes are tracked and archived for reverting if needed.

Following are the rules which will be implemented in the SCR layer. The rules specify how the database should be designed in order to enable proper functioning of the SCR and eventually, the 'Lifecycle Tool'.

1) Building the database
    a. Entries for names of specific mechatronic objects, for example, 'blanket', 'blanket with motor' etc.
    b. Entries for names of domains, for example, 'printing', 'packaging' etc.
    c. Entries for names of companies/organizations
    d. Links between the above mentioned entities, for example, 'blanket with motor in the printing company named 'xxx'.

2) Building an interface to the database
    a. Read the entries using specific queries combining domain and/or company names, for example, 'retrieve all the units named 'servo' in the company 'xxx'.
    b. Write entries to specific domains and/or companies
    c. Remove entries from specific domains and/or companies
    d. Access, authorize only specific stakeholders to read from and/write to the database

3) Versioning in the database
    a. Enable tracking and archival or any changes made to the database and make it possible to revert back to particular edits

4.2.2 The employee-customer interaction (ECI) layer

The ECI is the layer which would contain all the data which is critical towards development of the automation solution. This layer consists of three major modules,

1) The communications platform (CP)
2) The company offerings platform (CO)
3) The machine visualization interface (MVI)

Together, these three modules aim at capturing, archiving and showcasing all the interaction which happens between employees and customers. As mentioned in section 3.2.1, the ECI layer will work in close contact with the SCR layer, following all the terminology and concepts defined therein. The ECI will archive and showcase data specific to customer companies and their individual products. This data will include meeting minutes, electronic interactions with the customer, data from customers like machine modules, 3D models of their machines, drawings, photos, pseudocode, packML diagrams etc. The presence of such knowledge about a customers' product in one place will greatly increase the efficiency in the development of automation solutions.

The three modules within this layer are described as follows,

4.2.2.1 The Communications Platform (CP)

As seen from the interviews and the field data, employee-customer interactions happen in an unstructured way. Employees take notes in the meetings on their personal computers in a text-editor or other such similar software. Some notes are taken mentally. Some notes exist in the form of images of whiteboards taken during the meetings. These methods are lossy as some information may be forgotten by the employees. The problem with taking notes on text-editors is that it is not organized and shared across all the relevant stakeholders. It is evident that all this information is captured in an unstructured manner and the data is scattered across stakeholders.

The CP aims to tackle these problems by gathering the data in single a software environment. The CP module in the 'Lifecycle Tool' will allow one person in the meeting to create a project. This project will be assigned to a name and a company. All employees visiting the meeting will collaboratively edit the project adding information as it surfaces. It will be possible to not only enter plain-text data but also code, freehand drawings, images, audio, video, animations, 3D diagrams etc. As mentioned earlier, the module would fetch data from the SCR and make recommendations while the employee is typing. This would ensure that the terminologies used across the project are consistent, thus avoiding ambiguities and confusion. The CP would also have an audio recording feature. This will record the

conversations in a meeting (if permitted) and transcribe the contents in plain text for future reference.

A communication channel between the customer and the employees will be a part of the CP. This would enable one-to-one communication between the engineers and the customers so that they can get their questions answered right away. Data from all these communications will be saved. This will facilitate various benefits like preventing loss of critical information, presence of information in a single place etc.

4.2.2.2 The Company Offerings Platform (CO)

The CO is a central interface to the customer which conveys the offerings and the capabilities of the company to the customer. It is different than a mere product catalogue in way that it is interactive and constantly updated. The main objective of the CO would be to convince the customer of the advantages of software and automation in their industrial machines. This would involve not only examples and animations and/or videos of the capabilities of software but also statistics about improvements, displayed using graphs and other visualizations. The CO would essentially convey to the customer about the services the company has to offer and convince them to choose to have software in their machines.

4.2.2.3 The Machine Visualization Interface (MVI)

For an engineer to write code for a machine, it is essential that the engineer understands how the machine functions. It was discovered in the interviews that a visual tool, where everyone can see how the machine works physically would help engineers to understand better, how a machine works. It may not be possible in the case of all the customers to have a 3D file of their machines, but whenever it is available, the MVI would take this file as in input and enable the visualization of the machines or the modules within the machines. Engineers would be able visualize the machines or their modules in slower speeds to understand their working.

The MVI would also include an interface which displays the composition of the machine, for example, what modules are present in the machine and how do they interact with each

other. It will be possible to generate a hierarchical diagram of the machine and edit each module's properties to reflect the real machine. It will be possible to save such configurations and reuse them as required in the future. A visual overview of the entire machine will facilitate in better and faster understanding of complex machines. It will be possible to create a new machine configuration in the MVI during the customer meeting. This would facilitate winning the customer and also save the maximum number of specifications at the same time.

4.2.3 The Code Generation (CG) layer

The code generation layer would save time by automating certain repetitive coding tasks. As discovered in the interviews, all software projects generally have a common template and a standard set of libraries. The parameters which vary across projects are the project names, machine and module names, their specific configurations and their relationships with each other. Taking information about names, specific module configurations and their relationships from the configurations in the MVI layer, the CG layer will generate an actual project with code templates and libraries in it. The CG layer will be designed to output code compatible with various industry leading automation software.

State machine diagrams are often used in the automation industry to express machine and their functionality. The CG layers aims to make it possible to generate code from state diagrams of machines. Such a tool, known as 'DiagramDraw', currently exists as proprietary software. Guo et al. (2007) claim that such a tool can considerably assist designers to create state machines and convert them to code in Structured Text (Guo et al. 2007). Guo et al. (2007) also claim that the application of such a tool in the real world will help in significant reduction in the product development cycle.

Based on the idea of 'DiagramDraw', the CG layer will take a state diagram designed in the MVI layer and convert it to code in the Structured Text language. It may be beneficial to have code generated in other languages as well to facilitate flexibility.

The overall design of the lifecycle tool can be visualized as follows,

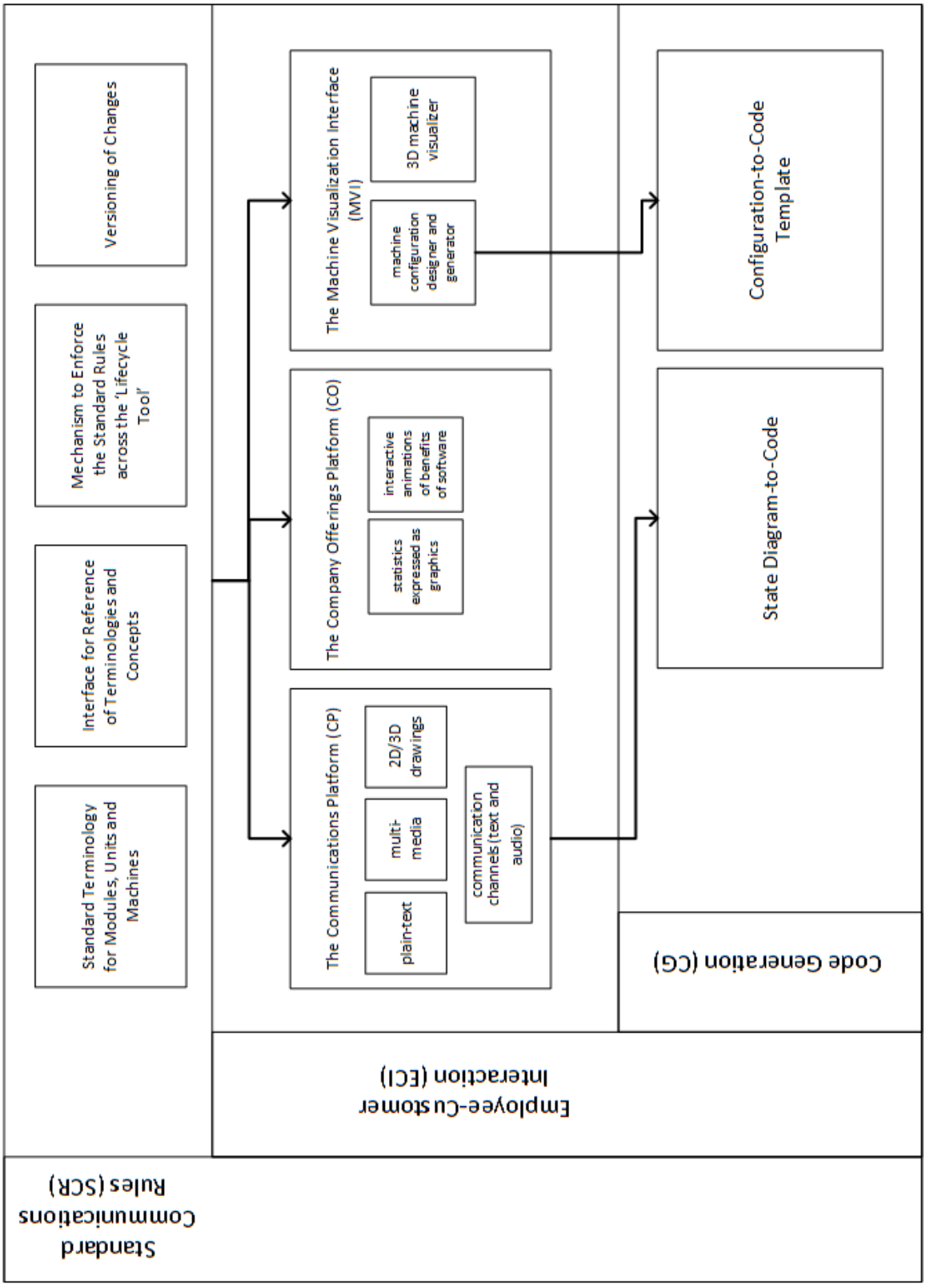


Fig. 7. Overview of the 'Lifecycle Tool'

5 RESULTS

This thesis work was carried out with the goal of discovering problems within the development lifecycle and proposing possible solutions towards solving those problems. Various issues were discovered in the interviews, meetings and workshops. Some of these issues were found to have various undesirable consequences. It was the goal of this thesis to address some of the problems through software. A solution called ‘Lifecycle Tool’ was designed and proposed in order to overcome the issues discovered. ‘Lifecycle Tool’ takes a holistic view of the entire product lifecycle and acts as a layer which serves the development process in various stages. Currently, the machine visualization interface (MVI), one of the modules of the ‘Lifecycle Tool’ proposed in this thesis is under development.

6 DISCUSSION AND CONCLUSIONS

It is evident from the interviews and their analysis that there are various actions leading to inefficiencies in the product lifecycle of an automation solution. It is often said that discovering the problem is a major step in solving the problem itself. Analysis of the interviews revealed that ‘capturing information’ was the most recurrent problem in the lifecycle. By making use of software and rules enforced, this problem can be overcome.

As mentioned earlier, one of the modules of the ‘Lifecycle Tool’ is already under development at the firm where the research was carried out. This module of the ‘Lifecycle Tool’ can be used as soon as its development is complete. It is independent of the other modules in the main tool, except of the fact that the SCR (standard communication rules) has not been implemented yet.

Once the module is commissioned for use by the company, new information will be generated about how the module and ‘Lifecycle Tool’ affect the development lifecycle. This would lead to further evolution of the design on the tool. As design science research is an iterative process, this thesis work is only the very inception of the final design and the tool. The design will evolve as new ideas and requirements arise. Some of the future work which has the potential to be carried out will be,

1) Development of the standard communications rules layer (SCR)
2) Development of the communication platform (CP) and the company offerings (CO)
3) Development of the code generation layer (CG)
4) Evaluation of the developed design/tool
5) Further development of the tool by iterative design and evaluation cycles

Several different sources were employed in the case study conducted in this thesis work. This included six structured interviews, one unstructured open interview and one employee-customer meeting. Similar conclusions were drawn from these different sources and such triangulation led to robust conclusions.

Employees who took part in the case study worked across varied projects. This factor was not considered during the interviews and it may have acted as a limitation in the case study conducted. It is recommended that future work and further case studies take into account the differences between projects and different employees' roles. Also, as the company is multinational, it can be assumed that there exist various issues related to cultural and linguistic differences when the stakeholders interact. It is recommended that these issues are taken into consideration as well.

According to Hevner et al. (2004), excessive focus on technological artefacts and insufficient theory may lead to a well-designed artefact which is of no use in the real world. This is one of the threats when carrying out design-science research. These problems were overcome in this thesis by making sure that sufficient data was collected before designing the software architecture which ensured that the theory base wasn't insufficient.

In order to overcome the issue of excessive focus on technological artefacts, the interviews were designed together with two other master's degree students who were not studying the development of software. These students were studying 'service design' (Stickdorn and Schneider, 2011), which encompasses the design of not only software solutions, but also solutions in the form of any other services. This ensured that the interview questions did not bias the interviewees towards the development of technological artefacts only. But, as the design of the solution was carried out individually, it is likely that researcher bias may have led to an excessive focus on the development of a technological artefact in this thesis work. Now, it remains to be seen if the designed solution is effective in the real world, or not.

The research work in the thesis was carried out at a specific, multinational industrial automation company. It can be safely assumed that at least some of the problems discovered in this company also occur in other multinational industrial automation companies. It is possible that more or less problems occur in different companies depending on their business models and ways of operation.

Similar, but very specific research from year the 2007 was found, where a software tool called 'DiagramDraw' (Guo et al. 2007) was developed in order to facilitate faster development through graphical designing of machine modules, followed by code generation

in the programming language Structured Text. The occurrence of such research work indicates that organizations are constantly engaged in making internal processes better and more efficient by designing and developing services and tools. Hence, a claim can be made that the research work carried out in this thesis is applicable to at least similar industrial automation organizations or newly formed or forming industrial automation organizations who are interested in designing efficient internal procedures.

7 SUMMARY

Design science and case study techniques were used in the writing of this thesis work. Problems and issues in the development lifecycle of industrial automation solutions were discovered and ‘Lifecycle Tool’ was proposed. Once commissioned, this tool will overcome the issues and problems, thus enhancing the overall efficiency in the firm. The ‘Lifecycle Tool’ is a holistic tool which would contribute to the entire development process. The firm has started the development on one of the modules from the ‘Lifecycle Tool’. Also, after commissioning, the design will evolve iteratively as the process would involve both evaluation and designing.

In order to ensure validity in the research, triangulation was used in the case study. Certain limitations were identified in the discussion and ways to cope with them in the future work were discussed. Also, the dangers in design science research were identified and addressed. The issue of excessive focus on the technological artefacts was overcome in the interviews by including different perspectives while designing the interviews. The issue of an insufficient theory base was addressed by collecting sufficient data in the interviews and meetings.

The applicability of this research work lies in the domain of present, multinational industrial automation companies as well as new and upcoming industrial automation companies who are interested in employing efficient procedures and avoiding problems which may occur.

REFERENCES

1. Baskerville, R.L., Stage, J., 1996. Controlling Prototype Development through Risk Analysis. MIS Quarterly 20, 481.
2. Beck, K., 1999. Embracing change with extreme programming. Computer 32, 70–77.
3. Boehm, B.W., 1988. A spiral model of software development and enhancement. Computer 21, 61–72.
4. Cooper, R.G., 1993. Winning at New Products; Accelerating the Process from Idea to Launch. Reading, MA, Addison-Wesley Publishing Company.
5. Denning, P.J., 1997. A New Social Contract for Research. Communications of the ACM 40, 132–134.
6. Gavish, B., Gerdes, J., 1998. Anonymous mechanisms in group decision support systems communication. Decision Support Systems 23, 297–328.
7. Guo, W., Zhou, M. and Wei, D. (2007). DiagramDraw: A State Machine Diagram Designer for Flexible Automation. *2007 IEEE International Conference on Automation Science and Engineering*.
8. Hevner, A., March, S., Park, J. and Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), pp.75-105.
9. Kemerer, C.F., 2002. Editorial Notes. Information Systems Research 13, iii-iv.
10. Nunamaker, J., Chen, M., 1991. Systems Development in Information Systems Research. Journal of Management Information Systems 7, 89–106.
11. Robson, C., 2002. Real World Research. Blackwell.
12. Runeson, P., 2012. Case study research in software engineering: guidelines and examples. Wiley, Hoboken, NJ.
13. Runeson, P. and Höst, M. (2008). Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, 14(2), pp.131-164.
14. Seaman, C., 1999. Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering 25, 557–572.
15. Simon, H.A., 1996. The Sciences of the Artificial. MIT Press, Cambridge, MA.

16. Silver, M.S., Markus, M.L., Beath, C.M., 1995. The Information Technology Interaction Model: A Foundation for the MBA Core Course. *MIS Quarterly* 19, 361–390.
17. Stickdorn, M., Schneider, J., 2011. This is service design thinking: basics, tools, cases. BIS Publ., Amsterdam.
18. Stern, R.N., Pfeffer, J., Salancik, G., 1979. The External Control of Organizations: A Resource Dependence Perspective. Contemporary Sociology 8, 612.
19. Tillquist, J., King, J.L., Woo, C., 2002. A Representational Scheme for Analyzing Information Technology and Organizational Dependency. MIS Quarterly 26, 91.
20. Tsichritzis, D., 1998. The Dynamics of Innovation, in: Denning, P., Metcalfe, R. (Ed.), Beyond Calculation: The Next Fifty Years of Computing. Copernicus Books, New York, pp. 259–265.