2009

FEED-FORWARD IN SOFTWARE ENGINEERING WITH PARTICULAR FOCUS ON REQUIREMENTS ENGINEERING AND SOFTWARE ARCHITECTING

Shamin Ahmed

FEED-FORWARD IN SOFTWARE ENGINEERING WITH PARTICULAR FOCUS ON REQUIREMENTS ENGINEERING AND SOFTWARE ARCHITECTING

(Spine title: Feed-forward in Software Engineering)

(Thesis format: Monograph)

by

Sharmin Ahmed

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

This study is intended to determine the characteristics, impact and state of the practice of feed-forward in software engineering; in particular, in the fields of Requirements Engineering (RE) and Software Architecting (SA). Feed-forward is used in many domains such as systems engineering, neural networks, management and psychotherapy. However, in software engineering, especially in RE and SA, the concept of feed-forward is not well researched. For example, what are the characteristics of feed-forward information? What effect does feed-forward information have on architectural artefacts and software project aspects such as cost, quality, time, etc.? What is the current state of practice of feed-forward? A knowledge seeking empirical investigation including an industrial survey and an embedded case study with four projects as four units of analysis were carried out based on these questions. The overall findings of this study show that the most common types of information that are fed-forward consistently are requirements and architectural information. This information affects a multitude of aspects of a software project (such as time, cost and quality) and influences several architectural artefacts (such as tactics, patterns and decisions). The results also show that approximately 20% of software professionals have never, or rarely, practiced feed-forward in their organizations. On the other hand, approximately 66% of software professionals practice feed-forward in their organization in varying levels ("sometimes", "most of the time", "always"). 64% of software professionals find feed-forward to be useful for their organization and 4% thought that feed-forward would not be useful, citing reasons such as information overload and lack of motivation. From a researcher's perspective, determining the properties of feed-forward could provide ground work for doing further research on feed-forward such as: the practice of feed-forward in the other areas of software engineering and the comparison of feedback and feed-forward in software engineering.

Keywords: software engineering, requirements engineering, software architecting, feed-forward, empirical study.

Definitions

Feed-forward: An anticipatory activity of passing information from one location in the process to another where the recipient of the information could possibly benefit by making use of it ahead of time.

"Push" Feed-forward: A type of feed-forward in which information is proactively sent (i.e., pushed) from the source to the recipient without the recipient having to request for it.

"Pull" Feed-forward: A type of feed-forward in which information is collected (i.e., pulled) from the source by the recipient.

Acknowledgements

First and foremost, I would like to thank Dr. Nazim Madhavji for his constant guidance and supervision. Without his strong motivation, valuable inspiration and support, this work could not have been completed. Your passion for education and research are admirable.

I would also like to thank all the members in my research team. Special thanks goes to Remo Ferrari who has suggested many valuable ideas for conducting the research.

I would like to thank the participants of the studies; I greatly appreciate the energy and effort that was expended on their part, and hope that the project was a great learning value for each of them and that the research output will be of good value to everyone.

Last, but not the least, I would like to thank my husband, Muhammad Iftekher Chowdhury: my fellow research member, and continuous source of support and ideas. I am grateful for his love, patience and trust that acted as a driving force for the completion of the work.

Table of Contents

List of Figures:

List of Tables:

Chapter 1. Introduction

Requirements Engineering (RE)[1] methods [18] and Software Architecting (SA)[2] methods [5] are iterative and based on feedback. A software architect receives information from requirements engineers about new or changed requirements after the completion of an iteration of architecting and once the requirement has been validated. An architect also provides feedback to the requirements engineers after they have validated the architecture [36]. Both these processes are lengthy and unfavorable for a software process in terms of cost [19] and effort [9] [23].

Traditionally, in control engineering, feedback and feed-forward are used in combination to significantly improve performance over simple feedback control [6] [44].This is because while feedback alone can retain a control variable that has started to deviate due to any external disturbance to a system, it can only do so after the deviation has started, but when feed-forward is incorporated to the system, it measures the disturbance prior to deviation and using that measurement, feedback maintains the control variable. In some domains, such as in management, feed-forward alone has been more effective and efficient than feedback [10]. This is because it is difficult to provide professional feedback that is not taken personally, but since feed-forward is opinion about the future it is accepted more positively. The mention of feed-forward in conjunction with feedback in the field of software engineering has so far been rare [1], [26] and [21]. However, certain properties of the activity of feed-forward or the information fed-forward is not clear from the literature. For example, what are the characteristics of feed-forward information? What effect does feed-forward information have on software artefacts and software project aspects? What is the current state of the practice of feed-forward? If feed-forward were incorporated into the normal flow of information in RE and SA, it could

[1] For the rest of the thesis, the acronym RE refers to Requirements Engineering.
[2] For the rest of the thesis, the acronym SA refers to Software Architecting.

significantly improve the performance of the processes. Hence, determining the answer to such questions has its implications in software development practice in industry and in research as well.

1.1 Significance of research

From a practitioner's perspective, determining the impact of feed-forward in RE and SA may motivate the introduction of the feed-forward concept in software projects. Determining the characteristics of feed-forward could provide practitioners with a clearer idea about the types of information that are most productive when fed-forward and from which channel it needs to come from. From a researcher's perspective, determining the properties of feed-forward could provide ground-work for doing further research on feed-forward such as: the practice of feed-forward in the other areas of software engineering, tool development for capitalizing on feed-forward information, and the comparison of feedback and feed-forward in software engineering.

1.2 Originality of research

To the best of the author's knowledge, there are currently no studies that discuss the characteristics, impact and state of the practice of feed-forward in software engineering. The only mention of feed-forward in RE and SA was in [26], [21] and [1]. However, these papers did not exclusively discuss feed-forward. The process of conveying information is obvious but the activity of feed-forward for conveying information ahead of time is not a well researched area in software engineering. This thesis provides evidence that the concept of feed-forward can be useful in industry and has a significant impact in the overall process of RE and SA.

1.3 Thesis organization

The thesis is organized as follows: Chapter 2 discusses the literature related to feed-forward. Chapter 3 describes in detail the empirical study, which includes a discussion on the GQM (goal, question, metric) approach [4] to structuring of the study, the research procedures, a description of the participants and the threats and risks involved in the studies. Chapter 4 describes the findings from the survey and the case study, their interpretations and a composite analysis of the results from the study. Chapter 5 discusses the implications of the study and Chapter 6 describes the limitations, future work and conclusions.

Chapter 2. Background

Feed-forward has been used extensively in many domains in conjunction with or instead of feedback. In the software engineering process, feedback is common but using feed-forward with, or instead of, feedback is not. There are also many processes that resemble feed-forward but are not explicitly called so. Based on the diverse application of feed-forward, it can be divided into two disciplines. Section 2.1 discusses feed-forward and activities similar to feed-forward in non-Software Engineering disciplines. Section 2.2 describes feed-forward in Software Engineering disciplines. Finally, Section 2.3 discusses in brief what has been researched so far regarding feed-forward and where the research gap remains.

2.1 Feed-forward in non-Software Engineering Disciplines

A simple example of feed-forward in a familiar domain could be in the control of temperature by a thermostat inside a building [1]. The feedback of the temperature created by the heating system in a building is used by the thermostat to check if the temperature has deviated from its desired level. Based on the feedback the thermostat triggers the heating/cooling system to begin operating thus bringing the temperature back within its target range. This is use of feedback to control temperature. By adding feed-forward mechanism to the thermostat, the thermostat would measure the temperature outside the building. By obtaining this earlier indication of temperature change, the outside sensor can trigger the heating/cooling system before there is a deviation from the inside target level.

In systems engineering, control systems are mostly either open loop or closed loop [6]. An example of a closed loop control system is a feedback system. In feedback control systems, in order to give the output of a set value, the output is fed back to the input and then the input is accordingly controlled to get the desired set value output. An example of

this is the control of temperature in a thermostat as already mentioned. An example of an open loop control system is a feed-forward control system. In a feed-forward system, the input to the control is predefined and hence there is no feeding in of output but rather a feeding forward of input to maintain an output of set value. There is no closed loop and no feedback of information from output. Combined feed-forward plus feedback control can significantly improve performance over simple feedback control whenever the variable that causes the controlled variable to deviate away from the set point can be measured before it affects the process output [44].

Feed-forward neural networks are the earliest and simplest form of artificial neural networks. In such a network, a mathematical or computational model is pre-determined based on a large number of calculations and approximations and then fed-forward into the system to find the solution to a future new problem. This form of neural network is commonly used in pattern recognition for e.g. to determine an unknown pattern of a DNA. Feed-forward neural networks have been used successfully in conjunction with recurrent neural networks in projects such as the greenhouse control in [31].

Classical conditioning, for e.g. eyelid conditioning, is a form of associative learning by the brain that was first demonstrated by Pavlov [15]. Feedback use of sensory information is like that of a thermostat: a sensory input (the thermometer in this analogy) is used during the execution of the movement (the heater in this analogy) to produce accurate movement (maintain room at constant temperature). The utility of feedback control is inherently limited by its sloth and by its tendency to oscillate when forced to operate quickly. Feed-forward control can remove this problem by using sensory information available prior to movement execution to make decisions about the motor commands [24].

During the 80's, feed-forward was an innovative technique suggested by Penn for family therapy [28]. This technique was particularly useful for families with chronic illness whose concept of a future time is usually frozen. As these families typically avoid discussing the past, talking about the future opens up solutions to dilemmas and problems. Goldsmith [10] has trained more than 10,000 business professionals and has shown how feed-forward can often be preferable to feedback in day-to-day professional interactions. This is because it is difficult to provide professional feedback that is not taken personally but since feed-forward discusses something that hasn't happened yet it is not taken as personally. In [10], Goldsmith explains the ten minute exercise of feed-forward and the eleven reasons as to why it should be practiced and, in some cases, why it is preferable to feedback. In the exercise, each participant is asked to play two roles. In one role, they are asked to provide feed-forward and give someone else suggestions for the future. In the second role, they are asked to accept feed-forward and listen to the suggestions for the future and learn as much as they can. The exercise typically lasts for 10 to 15 minutes, and the average participant has 6 to 7 dialogue sessions. The exercise as described in the article is mentioned below. In the exercise participants are asked to:

- Pick one behavior that they would like to change.
- Describe this behavior to randomly selected fellow participants.
- Ask for feed-forward.
- Listen attentively to the suggestions and take notes. Participants are not allowed to comment or critique on the suggestions in any way.
- Thank the other participants for their suggestions.
- Ask the other persons what they would like to change.
- Provide feed-forward.
- Say, "You are welcome" when thanked for the suggestions.

The process is repeated with another participant until the exercise is stopped. Goldsmith has also specified the eleven reasons as to why he thinks that feed-forward is preferable to feedback. The eleven reasons to try feed-forward instead of feedback as mentioned in the article are as follows:

1. We can change the future. We can't change the past.
2. It can be more productive to help people be right, than prove they were wrong.
3. Feed-forward is especially suited to successful people.
4. Feed-forward can come from anyone who knows about the task. It does not require personal experience with the individual.
5. People do not take feed-forward as personally as feedback.
6. Feedback can reinforce personal stereotyping and negative self-fulfilling prophecies.
7. Face it. Most of us hate getting negative feedback, and we don't like to give it.
8. Feed-forward can cover almost all of the same material as feedback.
9. Feed-forward tends to be much faster and more efficient than feedback.
10. Feed-forward can be a useful tool to apply with managers peers and team members.
11. People tend to listen more attentively to feed-forward than feedback.

There are many examples of feed-forward in the literature. But there are also examples where the activity of feed-forward is apparent, but the terminology used is not "feed-forward". An example of this is "early warning" systems. Early warning systems are primarily radars used to detect targets at a long range. By detecting an intruder as early as possible, defense is able to calculate ahead of time the scale and origin of the attack, the potential damage estimate and choose an appropriate response before the intruder reaches the target [30].

Other warning systems include those used to forecast stock market behavior [37]. A machine learning algorithm is developed that predicts future conditions and issues warning signals against the possible massive selling and pullout of investors. These warning systems are also used to detect financial crisis by checking the financial market.

In [34], Scheffer et al., discusses about complex systems such as abrupt shifts in the climate in the earth system or abrupt shifts in fish or wildlife populations which may threaten ecosystem. These systems have a critical threshold called "tipping point" in which the systems change abruptly from one state to another. However, there are some generic symptoms that occur in the systems before they approach the "tipping point". If these early warning signals can be figured out ahead of time before the critical transition then something can be done regarding the changes and appropriate measure can also be taken.

During the war in Kosovo, the Serbs knew about the probable attacks by NATO ahead of time by regularly observing NATO air bases and facilities, and "phoning home" [8]. As can be seen, this is a form of feed-forward because the Serbs knew about the attacks of the NATO air bases ahead of time and hence took necessary actions to counter them.

2.2 Feed-forward in Software Engineering

RE and SA methods are usually iterative and based on feedback. In the coarse-grain activity model [18] of RE process, requirements validation information is fed back from one activity to another and the different activities are repeated until the entire requirements document is accepted. Hence if there is any change in requirements or introduction of new requirements, then it needs to go through the prolonged process of validation and analysis before it is actually handed over to the architects for architecting. In the ADD method of architecting [43], an architect receives new requirements *after* an iteration of architecting and any change to the current architecture or some new critical

requirements will only be conveyed to the architects at the end of the iteration. Both these processes of RE and SA are lengthy, time consuming and unfavorable for a software process in terms of cost [19] and effort [23] [9].

In [9], during a four-month case study at IBM Ottawa software lab, the developers commented about how they were falling behind on their schedule due to the introduction of changes that they were not aware of. The unexpected changes not only pushed back their schedule but also the schedule of other developers whose work was linked with theirs. This made the developers ask the question as to whether it is possible to get a "heads up" on unexpected changes ahead of time to help them to face it and so avoid delay. It was also found out from the study that the information could have been made available ahead of time but was not.

Feed-forward in conjunction with feedback in the field of software engineering is not very well researched. In [1], the author mentions how feed-forward information (e.g., anticipated staff-turnover in the coming weeks) can be incorporated into time-estimation models to improve the re-estimation process in software projects. In the appendix of [1], Agresti (also please see Appendix A:) provides a list of critical risks and warning signals of problems in software projects. Alongside each risk or problem is a description of the way that a feed-forward model may contribute as part of an overall risk management approach and hence help and improve the process of re-estimation. In [26], Miller et al. mentioned that feed-forward information in RE or SA processes could allow the requirements engineers to deliver critical information to the architects *prior* to the delivery of the validated new requirements. This information, if made available to the architects ahead of time, could be useful for specific architectural enhancements and change while the new requirements are still being elicited in the RE process.

In [21], Lehman first discusses about how software evolution processes can be described as multi-agent, multi-level, multi-loop feedback systems. Unanticipated circumstances and unexpected conditions, specification changes, performance problems, budget changes, for example, lead to process adjustments, adaptations and changes. Such unplanned changes are error prone and undesirable. It is not very desirable using feedback to deal with such changes and anticipating these changes and knowing about them ahead of time avoids much of the backtracking that may normally need to be done.

Also, in agile methods, the concept of feed-forward is not common. As a matter of fact, in agile architecture modeling, it is expected to consider future changes but not act on them until needed. Instead one should focus on building what is needed imminently [2]. However, in the case of a major change or new requirement, such feedback mechanisms are not useful. Delivery of the changes ahead of time can avoid backtrack and rework.

2.3 Research Gap

Feed-forward is a common term in many disciplines as was seen in the previous sections. In many of them feed-forward was used together with feedback to enhance the efficiency of the process. It can also be seen that the term feed-forward has also been suggested for use in some software engineering work. For example, in [26] it was mentioned as to how feed-forward can be used to bundle critical information from requirements engineers to the software architects. This basis can be used to further explore feed-forward and determine what the characteristics of this feed-forward information are and what impact this information can have on different aspects of software architecture.

As feed-forward is not common in software engineering organizations today, it may also be a point of analysis as to what is the current state of the practice of feed-forward? Why is feed-forward not yet practiced in organizations? If it is, how useful is it and what could

be the motivation behind feed forwarding information or expecting feed forward information?

In the exercise suggested by Goldsmith for feed-forward [10], participants provided feed-forward information and also received feed-forward information. This is similar to "Push Feed-forward" and "Pull Feed-forward" defined in the definitions section. Even though there is no explicit mention of "push/ pull feed-forward" in the literature, there are certain methods that resemble them. For example, the concept of change cases [3] , is similar to the concept of "push feed-forward". A change case is a means to report possible future new requirement or a modification to an existing requirement ahead of time. It is a template with three primary fields: the potential change, the likelihood of that change occurring and the possible impact the change can have on the system. Change cases in the form of a new potential requirement for a system or modifications to existing requirements can be identified throughout the course of the development process and provided to anyone relevant ahead of time. Change cases can be used in an agile manner and has a number of uses:

- If you know about potential changes to your system, you can make better architectural decisions.
- As you already have an idea about the future requirements, you can take action on them right now instead of learning about the change later on and spending long hours in meetings, etc.
- It is possible to get the requirements prioritized ahead of time from stakeholders.

In [23], the author poses the question of whether it is possible to pull information ahead of time from stakeholders to find out what they need and what will be useful for them.

However, there is no analysis of such "push" and "pull" of feed-forward information and needs further research.

Chapter 3. The Empirical Study

In this chapter, we describe the empirical study. Section 3.1 describes the Goal, Questions and associated Metrics that were used as the research paradigm. Section 3.2 describes the research procedure for the empirical study. Section 3.3 describes the participants of the study and finally section 3.4 describes threats and risks.

3.1 Goal, Questions and associated Metrics

The GQM [4] approach was used as the research paradigm. This approach helps to ensure that measurements taken during the empirical study are aligned with the specific research questions which, in turn, are aligned with the overall goal of the research.

In [25], it is mentioned that the key step towards understanding a process activity is to develop an understanding of the activity by characterizing the activity. This should be followed by assessing the impact the activity has on the process and identifying the improvements in process due to the activity. The final step includes enhancement of the current development policies by incorporating the new knowledge. Thus, it is important to understand characteristics and impact of feed-forward and also determine what the current state of practice of feed-forward is.

So, the overall goal for the research is formulated as:

"*To determine the characteristics, impact and state of the practice of feed-forward from the viewpoint of project stakeholders in the context of software development projects with focus on requirements engineering (RE) and Software Architecting (SA).*"

To determine the *characteristics* of feed-forward information, the specific questions that need to be answered are:

Question 1.1: What are the different types of feed-forward information?

In Chapter 4, a list is made of the different types of information that could be fed-forward, particularly in RE and SA. Data from both the survey and the case study is used to answer this question. The associated metric (Mi.j) for this question is:

M1.1: Frequency of different types (e.g. requirements, information about architectural artefacts, etc.) of feed-forward information.

Question 1.2: What are the different types of sources and recipients of feed-forward information?

Information can be fed-forward from any source to any recipient. However, it is important to determine the place where it is needed the most and so it is necessary to determine the source and destination of feed-forward. Data from both the survey and the case study was used to answer this question. The associated metrics (for this question are:

M1.2: Frequency of different source (e.g. customers, requirements engineers, software architects, etc) of feed-forward information.

M1.3: Frequency of different recipients (e.g. customers, requirements engineers, software architects, etc) of feed-forward information.

Question 1.3: What is the frequency of "pull" feed-forward as compared to "push" feed-forward?

The activity of feed-forward has already been categorized to "pull" feed-forward and "push" feed-forward. To find the different characteristics of feed-forward along with types, source and recipient it is also important to determine the frequency of "pull" feed-forward as compared to "push" feed-forward. Data from only the case study was used to answer this question. The associated metric for this question are:

M1.4: Ratio of types (push/pull) of feed-forward

To determine the *impact* of feed-forward information the specific questions that need to be answered are:

Question 2.1: What aspects of a software project are affected by feed-forward information?

In Table 17 and Table 21, a list of the different types of aspects that could be affected by feed-forward information is shown. To get a better idea about the impact of feed-forward, it is important to determine the different aspects that are influenced by feed-forward information. Data from both the survey and the case study was used to answer this question. The associated metric for this question is:

M2.1: Frequency of different aspects (e.g. cost, quality, time, etc.) affected by feed-forward information.

Question 2.2: Which software artefacts (e.g., architectural artefacts) are influenced by feed-forward information?

In Table 18, a list of the different types of artefacts that could be affected by feed-forward information is shown. To get a better idea about the impact of feed-forward, it is important to determine the different artefacts that are influenced by feed-forward

information. Data from only the case study was used to answer this question. The associated metric for this question is:

M2.2: Frequency of different software artefacts (e.g. architectural driver, tactics, patterns, etc.) influenced by feed-forward information.

To determine the *state of the practice* of feed-forward information the specific question that needs to be answered is:

Question 3: What is the current state of the practice of feed-forward?

If feed-forward were to be practiced in the industry, it would be necessary to get the opinion of relevant industry professionals about their view on feed-forward. It would be necessary to ask them about the level (refer to Section 4.1.2.1) of practice of feed-forward and the underlying reasons for the practice of feed-forward. Data from only the survey was used to answer this question. The associated metrics for this question are:

M3.1: Frequency of different levels ("Never", "Rarely", "Sometimes", "Most of the time", "Always") of practice of feed-forward

M3.2: Proportion of feed-forward that is being useful for software projects.

M3.3: Frequency of different reasons behind the absence of feed-forward in software projects.

In [45], Yin mentioned that the key to a research question is the understanding that it has "substance", (for e.g. what the study is about) and "form" by asking a "what, who, where, how many, how much, how, and why" question. Table 1 shows the possible substance of interest along with the form of the question. The table shows that all the forms of

questions have been asked to reach the overall goal. Because of these we can say that the research questions satisfy the goal.

Table 1 Possible substance of interest to satisfy the goal

Parts of goal	**Question Format**	**Substance**	**Research Question**
Characteristics of feed-forward	"What"	types of information fed-forward	Question 1.1
	"Who"	types of sources/ recipients	Question 1.2
	"How"	frequency of "pull" as compared to "push" feed-forward	Question 1.3
Impact of feed-forward	"What"	aspect of software project affected	Question 2.1
	"What"	software artefacts influenced	Question 2.2
State of the practice	"how much"	level of practice of feed-forward	Question 3.1
	"why"	why feed-forward is useful	Question 3.1
	"why"	why is feed-forward not practiced?	Question 3.1
	"Where"	RE and SA	context of the goal

It is, however important to mention that due to time and resource constraints the metrics to satisfy the questions were limited to certain aspects. For example, to determine the impact of feed-forward, the metric used was aspects affected and software artefacts influenced. However, metrics such as degree of impact or the degree of influence were not measured in this thesis due to constraints of time and resource.

3.2 Research Procedures

The following sections discuss the research procedures used for the empirical study. Section 3.2.1.1 discusses the design of the instrument and data collection for the case study. Section 3.2.1.2 discusses the analysis of data for the case study. Section 3.2.2 discusses the survey and Section 3.2.2.1 discusses the instrument design and data

collection for the survey. Finally section 3.2.2.2 discusses the data analysis for the survey.

3.2.1 Case Study

Using an exploratory research methodology [32], an interpretive [32] single case embedded case study was conducted in accordance with [33]. A single case embedded study is a single case study with embedded units of analysis. In this case, the units of analysis are four group projects that were undertaken in two graduate level Software Architecting courses, one held in spring, 2009 and the other held in summer 2009.

In each of the projects, the group members were assigned to elicit the requirements and develop the architecture for a particular system. A teaching staff acted as a customer to each of the projects. In the four projects, the systems that were to be implemented were that for a "Banking System" and a "Garage Door system". Project 1 and Project 3 group members implemented the Banking system while Project 2 and Project 4 group members implemented the Garage Door system. The group members were chosen based on their academic and/or industry experience in RE and SA and convenience sampling [41]. Convenience sampling is a non random sampling technique where subjects are chosen based on proximity, ease of access and willingness to participate. As the participants were registered students they were available and because the group members willingly completed the questionnaires even though it was not part of their coursework the sampling is considered convenience sampling. The placement of the group members in the different projects were based on their background to make sure that project members had the relevant application domain knowledge. The projects were each implemented over the span of 4 weeks. Projects 1 and 3 had two 2 hour sessions every week while project 2 and project 4 had 6 hour sessions with breaks in between once every week.

3.2.1.1 Instrument design and Data collection

According to Lethbridge et al. [22], a data collection technique of category first degree is when there is direct involvement of researchers during data collection. During the collection of data from the case study, the researcher was in direct contact with the project members hence the data collection technique was of category first degree. Also an instrument was used for data collection. After every session, a questionnaire (please see Appendix D) was handed to each of the individuals involved in the project. The teaching staff present in the role of customer was also expected to fill out the questionnaire after each session. A discussion was held between the participants of each project for 10 minutes to discuss the feed-forward during that particular session and then the questionnaire was completed. The projects had a fixed design process [32] because everything including the questionnaires were designed before the projects started and were not changed over time. The questions in the questionnaire were based on the metrics shown in section 3.1 and were limited to the area of RE and SA in accordance to the study. The questionnaire for projects 1 and 2 were in the form of an excel spreadsheet and provided to the group members and they had to fill it in and submit it after every session. Since the process seemed a bit arduous for the group members of project 1 and 2, for projects 3 and 4 the questionnaire was redeveloped (without changing the semantics or content) as a php [29] mysql [27] tool using the framework CakePHP [7] (the tool is not in the scope of the thesis). In these two projects, the requirements and artefacts developed were automatically connected for better reference. The group members accessed the tool and filled in the form and their results were later extracted from the database. The metrics measured in each of the questions in the questionnaire of the case study is shown in Table 2.

Table 2 Questions in the case study and the associated metrics measured from the questions

	Metrics					
Question #	**M1.1**	**M1.2**	**M1.3**	**M1.4**	**M2.1**	**M2.2**
2(a)		x	x			
2(b)	x					
2(c)						x
2(d)					x	
2(e)				x		
3(a)		x	x			
3(b)	x					
3 (c)				x		

3.2.1.2 Data Analysis

The data was collected from the questionnaire provided. There were both closed [17] and open ended [17] questions involved so the researcher had to check the artifacts for the first two case studies to check for coherency. In the latter projects, artifacts were automatically connected to the answer. Since there was both qualitative and quantitative data involved it was actually a "mixed methods" approach [32]. Hence, both statistical and analytical generalization was done.

3.2.2 Survey

In order to get opinions of software professionals regarding feed-forward, an industry survey was conducted. The survey was exploratory and involved both quantitative and qualitative data. The inclusion criteria [16] and exclusion criteria [16] are used to identify subjects who will or will not be participants in the survey. The survey was carried out amongst software professionals who had a good knowledge in requirements engineering, software architecting and other software engineering processes, hence this was the inclusion criteria and all other professionals were excluded. The participants for the survey were chosen based on availability sampling [12] and snowball sampling [12].

Availability sampling is the seeking of responses that fall under the inclusion criteria and are available and willing to participate in the research. Snowball sampling refers to the reliance on reference from initial respondents to generate additional respondents.

3.2.2.1 Instrument design and Data collection

The survey was of type descriptive design survey [17] where descriptors of the phenomenon are captured. There were both closed and open ended questions involved. The questions of the survey were designed based on the metrics mentioned in section 3.1.

The survey was web based, hosted in [39] and restricted to a single page. Participants had the option to fill out the survey partially, save the information and return at a later time to fill out the rest of the survey and then submit it. Definitions of relevant terms and the objective of the study were explained in general terminology upfront in the survey.

It is important to mention that one industrial survey was conducted to collect data for two studies: the first study is the one described in this thesis and the second is a complementary investigation[3]. The survey questionnaire was divided into three sections. Section 1 was the background section and development of this part can be attributed to the authors of both the studies. Section 2 and section 3 focused on individual topics and the design for these sections can be attributed to the respective researchers. Also, both the studies involved analysis of the participants demographic data found from the background section.

The metrics measured in each of the questions in the survey (please see Appendix B: Survey Questionnaire) is shown in Table 3.

Table 3 Questions in the survey and the associated metrics measured from the questions

[3] This complementary investigation is on the concept of information lost during software engineering.

Survey	Metrics								
	M1.1	M1.2	M1.3	M1.4	M2.1	M2.2	M3.1	M3.2	M3.3
Q # 15							x		
Q # 16									X
Q # 17								x	
Q # 18	x								
Q # 19					x				
Q # 20		x							
Q # 21				x					
Q # 22			x						
Q # 23				x					

3.2.2.2 Data Analysis

There were both ordinal and categorical data involved and so both statistical and analytical generalizations were done for the analysis of the data.

3.3 Participants

Determining the participants for the study was an important step in the design of the empirical study. Section 3.3.1 discusses the survey participants, including their background, organization and geographical distribution. Section 3.3.2 describes the case study participants.

3.3.1 Survey participants[4]

The participants of the survey ranged from programmers to consultants and chief technical officers with varying number of years of experience and different geographical distribution. In total there were 32 participants from a total of 23 different companies with 1 to 15 years of industrial experience. The background of the participants will be

[4] As mentioned in Section 3.2.2.1, this participant section is used as it is in another complementary investigation.

described in more detail in the following three subsections. In the participants' background subsection, the role or title of the participants, their area of expertise and number of years of experience will be described. In participants' organization section, the team and project size and the type of process models followed in the organization will be discussed. In the final subsection the geographical distribution of the participants will be discussed.

3.3.1.1 Participants' background

The role of the majority of survey participants was that of programmer (38%) or senior software engineer and analyst (38%) while only 3% of the participants were software maintenance engineer or a consultant. The main focus of this study was in the area of RE and SA. Upon verbal communication with the participants, it was determined that in the organizations the participants worked in, there was no explicit role or title of software architect or requirements engineer and the senior software engineer and system analysts were responsible for RE and SA. A reason behind this was that a good number of the participants followed agile methods and so even though an individual in an organization had one role they could have several different responsibilities. The distribution of the role of the participants in the organizations is shown in Table 4. The difference between the role of programmer and senior software engineer and analyst is that a programmer is mostly responsible for coding and low level design whereas a senior software engineer and analyst are responsible for upfront activities such as RE, SA, high level design and planning. It should also be noted that other stakeholders were also involved and responsible for RE and SA.

Table 4 Percentage distribution of role or title of the respondents of the survey

Participants Background	Percentage
Programmer	38%
Senior software engineer and analyst	38%
Quality assurance engineer	6%
Testers	6%
Software maintenance engineer	3%
Management	6%
Consultant	3%

The background experience of the participants is shown in Table 5. A high number of the participants have experience of design and coding. This may be due to the fact that in many of the organizations, the participants joined in the entry level job of programmer and/ or tester and are either promoted or switch to other areas such as RE and SA. However, a large number of the participants have background experience of both RE and SA as well.

Table 5 Frequency distribution of the background experience of the survey participants

Area of background experience	Number of participants
Requirements engineering	17
Software architecting	17
Design and Coding	28
Testing	20
Software maintenance	14
Project management	9
Quality control and/ or assurance	9
Process improvement	10

The participants have a range of experience from 1 to 15 years. One of the minimum criteria for the selection of the participants for this survey was at least one year of industrial experience in software engineering. The number of years of experience of the participants was broken down into four clusters as shown in Table 6.

Table 6 Percentage distribution of years of experience of the survey participants

Number of years of experience	Percentage
1 year	13%
2 years	25%
3 to 4 years	31%
More than 5 years	31%

3.3.1.2 Participants' organization

The software development lifecycle models followed by the participants ranged from traditional models like waterfall and iterative to a combination of agile methods like XP and Scrum. Table 7 represents the different lifecycle models followed by the participants. The "Others" models followed include lifecycle models such as rational unified process, model driven development and any other customized lifecycle model followed in the organization.

Table 7 Frequency distribution of software development lifecycle model followed in survey participants' organization

The total of the number of respondents is more than 100 because the participants had the option to choose more than one lifecycle models.

Software development lifecycle models followed	Number of respondents	Cumulative number of respondents
Waterfall	16	
Iterative	12	
Spiral	6	21
Agile-eXtreme Programming	8	
Agile-Scrum	14	
Feature Driven Development	5	20
Others	4	4

The typical team size of the projects that the participants worked in ranged from 1 to 5 to more than 10 which is why the team size were grouped into three clusters. The clusters and their percentage are shown in Table 8.

Table 8 Percentage distribution of typical team size in survey participants' organization

Typical team size (in persons)	Percentage
1 to 5	53%
6 to 10	41%
more than 10	6%

The typical project duration of participants' team ranges from less than 1 month to more than 2 years. The duration of the projects has been split into 4 ranges and is shown in Table 9.

Table 9 Frequency distribution of typical project duration in survey participants' organization

Typical project duration	Number of respondents
< 1 Month	4
>=1 month and < 6 months	20
>=6 months to <1 year	9
>=1 year to <2 years	9
>= 2 years	4

The organizations of the participants ranged from small (<50 people) to large (>2000 people). This was also broken down into four clusters as shown in Table 10.

Table 10 Percentage distribution of organization size of survey participants

Organization Size	Percentage
< 50 people	50%
>=50 people and < 200 people	15.63%
>=200 people to <2000 people	9.38%
>= 2000 people	25.00%

3.3.1.3 Participants' geographic distribution

It is important to note that the geographical distributions were based on the participants and not the organizations in which they work because some of the companies were multinational and hence had branches all over the world so it was more important to consider the location of the participation rather than the location of the main branch of the organization. The geographical distributions of the participants are shown in Table 11.

Table 11 Percentage distribution of survey participants by geographical location

Participants' geographic location	Percentage
Bangladesh	47%
Canada	31%
US	6%
Finland	6%
Australia	10%

According to government statistics reported in [42] on the experience of developers in China in 2007, it was found that 42% of the developers had less than two years of industry experience. This is comparable to the percentage of respondents in our survey with 1 to 2 years of experience (38%). The number of developers in China with experience of 2 to 5 years is 38%, which is comparable to the percentage of respondents in the survey with more than 2 and less than 5 years of experience (31%). 20% of the developers in China had an experience of more than 5 years which in the case of the survey was 31%. Thus, if the percentage distribution of experience of the survey respondents is compared to the statistics of developers in China it can be said that the experience of the respondents of the survey is equal or more than that of the developers experience distribution in China. According to a report published in 2008, China was the 4th largest software producer in the world [20].

According to a survey on 1298 software professionals by Forrester research [11], in 2009 the ratio between agile and traditional development in development teams is approximately 0.82 (45% agile: 55% traditional). If we look at the ratio between agile and traditional development in the survey, it is approximately 0.95 (63% agile: 66% traditional; please see Table 7).

So, from the above examples it can be concluded that the respondents of this survey are a good representation of a large population of software professionals.

3.3.2 Case study participants

The embedded case study had four units of analysis, namely four projects. The participants for two of the projects were drawn from the final year and graduate level Software Architecture Course at the University of Western Ontario. The participants for the other two projects were drawn from industry professionals taking a graduate course on Software Architecture. Each project had a teaching staff acting as a customer, who provided the project members with requirements and provided on-site feedback. Hence, there were a total of four customers for the four projects. Of the total 8 group members (excluding the customers) for the project 1 and project 2 that were held in a course conducted in academia, only two of the group members were undergraduate students while the remaining 6 group members where graduate level students with years of industry experience ranging from 2-5 years with an average of approximately 4 years. All the group members of project 3 and project 4 were industry professionals with industry experience ranging from 3 to 15 years with an average of approximately 9 years. The courses were conducted in winter 2009 and summer 2009. A summary of the four projects are shown in Table 12.

Table 12 Summary of case participants

	Project 1	**Project 2**	**Project 3**	**Project 4**
Number of participants	5(including one customer)	5(including one customer)	4(including one customer)	3(including one customer)
Course conducted in	Academia	Academia	Industry	Industry
Software development	Agile	Agile	Traditional Iterative	Agile

3.4 Threats and Risks

Since our study is exploratory, we are not looking for causal relationships with respect to the study constructs. Internal validity is of concern when investigating whether one factor affects an investigated factor and there is a risk that the investigated factor is also affected by a third factor. As such, we will not discuss threats to the internal validity of this study. In the following subsection, we will discuss separately the threats involved during the case study and the threats involved during the survey.

3.4.1 Threats to Case study

This section describes the threats to the case study that was conducted. Section 3.4.1.1 describes triangulations such as data triangulation, observer triangulation and methodological triangulation of the findings of the case study. Section 3.4.1.2 describes the external validity of the results of the case study.

3.4.1.1 Triangulation

Triangulation [32] is important for empirical research where qualitative data is involved. It is also relevant for qualitative data to compensate for measurement errors. It is a method of establishing the accuracy of a study's findings by taking different angles towards the studied object. There are different types of triangulation that can be used together to form a strong basis of validity. Three different triangulations were used for this study.

Data Triangulation: This triangulation refers to the use of more than one data source or collecting the same data at different occasions [32]. If there is consistency in the data

collected in the different occasions, then the data is not invalidated. In the case study, the data from the project members were collected after every session. To verify that the project members were reporting accurate feed-forward information, the artefacts and requirements implemented were checked against the reported feed-forward. In project 3 and project 4 the process of manual checking against requirements and artfacts was avoided because the participants had the facility to tag the particular requirements and artefacts they were referring to during their feed-forward. Hence, more than one data source was used for the collection of data.

Observer Triangulation: This refers to the use of more than one observer for the study [32]. After each session, all the project members sat for approximately 10 minutes to discuss the different information fed-forward during that session and then they reported their individual feed-forward. Thus, all the project members were observers in identifying feed-forward in the case study. Also two researchers were responsible for the collection of data.

Methodological Triangulation: This refers to the triangulation of combining different types of data collection methods [32]. In this study, both qualitative and quantitative data collection method was used and the quantitative data was consistent with the qualitative data. For example, when a project group member provided information regarding feed-forward, they also provided their rationales behind feed-forwarding the information. This rationale was consistent with the information that they had fed-forward. This consistency establishes methodological triangulation for the study.

3.4.1.2 External Validity

Population Validity: This validity is concerned with the extent to which it is possible to generalize the findings [32]. Using students as participants is a threat to the generalization

of the findings in industrial context. However, this threat is partially reduced by the fact that all the group members for project 3 and project 4 are industry professionals so that the settings would more closely mirror real-world context. In project 1 and project 2, only 20% of the group members were senior level undergrad students while the rest of the group members were all graduate students with previous industry experience. Also, recent research in software engineering [13] has shown that senior level students perform similarly to "novice" industry professionals with 1-2 years of industry experience.

3.4.2 Threats to survey

This section describes the threats to the survey that was conducted. Section 3.4.2.1 describes the construct validity of the survey including content and face validity. Section 3.4.2.2 describes the internal validity. Section 3.4.2.3 describes the external validity of the findings of the survey. Section 3.4.2.4 describes the reliability of the findings of the survey and finally section 3.4.2.5 describes the conclusion validity of the results of the survey.

3.4.2.1 Construct Validity

This refers to the extent to which the operational measures studied represent what is being investigated according to the research questions [32]. There are two types of construct validity that can be used as a strong basis of validation for the survey questionnaire and these are discussed below.

Content Validity: This refers to the extent to which the content of the questionnaire measure the metrics of the research question [35]. The questionnaire is based on GQM [4] and was reviewed, in several iterations, by at least three other researchers to iron out any issues with the questionnaire. In the survey, the options (such as requirements,

rationales, etc.) for the characteristics of feed-forward, the aspects (such as cost, quality, time, etc) influenced, and the artefacts affected (such as tactics, patterns, etc.) were all rooted in literature (e.g., [18], [5] and [36]). However, each survey question also had additional space that could be used by the participants to enter any additional options that may have been missing from the list but used in the organization of the particular participant. The idea of pull and push were completely new but agreed upon by several researchers.

Face Validity: Face validity is a measure of how representative a research project is 'at face value,' and whether it appears to be a 'good project' [40]. Face validity is strengthened by the consensus of experts [35]. In the survey, consensus was taken from three other researchers who specialized in RE and SA.

3.4.2.2 Internal Validity

Since no causal relationships were being inferred and since the study was exploratory threats to internal validity will not be discussed.

3.4.2.3 External Validity

This aspect of validity is concerned with the extent to which it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case [32]. If we look at the profile of the participants of the study, it becomes clear that a wide range of software engineers from various companies took part in the survey and to make the survey questions applicable for the diverse participants, more options were added as compared to the options in the case study. For example, in the aspects affected section in the case study, the options were limited to RE and SA type aspects within the scope of the case study but in the survey more options such as coders,

testers, integrators, etc., were added. This way, the survey results are not just valid for RE and SA, but also other areas of software engineering.

3.4.2.4 Reliability

The objective of reliability is to be sure that if an investigator later carried out the same study all over again then they would arrive at the same findings and conclusions [45]. Research bias was removed from the survey in the following ways:

- The objective of the survey was defined in general terminology in the beginning of the survey
- To make sure that the questions of the survey were not limited to focus of the research, several open ended questions were asked where applicable
- Another researcher was involved during the data collection and data analysis to remove research bias and error

3.4.2.5 Conclusion Validity

Conclusion validity is defined as "the degree to which conclusions we reach about relationships in our data are reasonable" [40]. The conclusions made were based on the findings of the study. Section 4.2 discusses composite analysis of the common questions asked in both the survey and the case study and it was found both the studies yielded similar findings. Thus, the conclusion for the empirical study can be considered valid.

Chapter 4. Results and Interpretations

This chapter describes the results and their interpretations. Section 4.1 discusses the results and interpretations for the case study and section 4.1.2 discusses the results and interpretations for the industrial survey. Section 4.2 discusses the composite analysis of the results of the case study and survey. It is important to note that the interpretations made based on the results are limited to the study and should not be generalized widely.

4.1 Results from Case study

Table 13 shows the frequency of feed-forward in the different projects. There were a total of 24 feed-forward with project 1 having the highest number of feed-forward (13) and project 4 having the lowest number of feed-forwards (2). On average there were 6 feed-forwards in each of the projects.

Table 13 Frequency of feed-forward in each of the projects

Unit of Analysis	Number of feed-forward incidents
Project 1	13
Project 2	4
Project 3	5
Project 4	2
Total	24
Mean	6

It can be seen from Table 13 that the number of feed-forwards were highest in project 1 followed by project 3, project 2 and project 4. It is worth noting that the project 1 had the highest number of requirements implemented (56 requirements) followed by project 3 (48 requirements), project 2 (36 requirements) and project 4 (27 requirements). Hence the number of feed-forward increased with increase in number of requirements. Also, feed-forward was more frequent in the banking projects (project 1 and project 3) than in the

garage door projects (project 2 and project 4). The banking project had a higher number of subsystems (4 subsystems) as compared to the garage door project (3 subsystems). These results can be used to formulate questions for future research such as whether the number of requirements and the application domain influences the number of feed-forward in a project.

4.1.1 Findings

The first research question was the following:

Q1.1 *What are the different types of feed-forward information?*

In the case study, after every session of the project meetings, the project members were asked to identify if there were any feed-forwards during that session and if there was what type of information it was. The project members were asked to select from a list of types of information as shown in Table 14. As already mentioned in section 3.4.2.1, the options provided to the participants were all rooted in literature (i.e., [18], [5] and [36]).

Table 14 Potential list of types of information fed-forward during case study

Type of information	Description
Requirements	This refers to one or more requirements
Requirement(s) related information	This refers to information such as assumption, rationale, relationship between requirements
Information about requirements	This refers to information such as requirements decomposition, refinement, etc
Information about requirement characteristics	This refers to information such as cost of implementation, implementation effort, resources needed, prioritization, change-related info, etc
Information about the architecture	This refers to information about the architecture to be implemented
Other	Any other type of information is specified here

The types of information that were reported by the project members as feed-forward information in the four projects are represented in Figure 1.

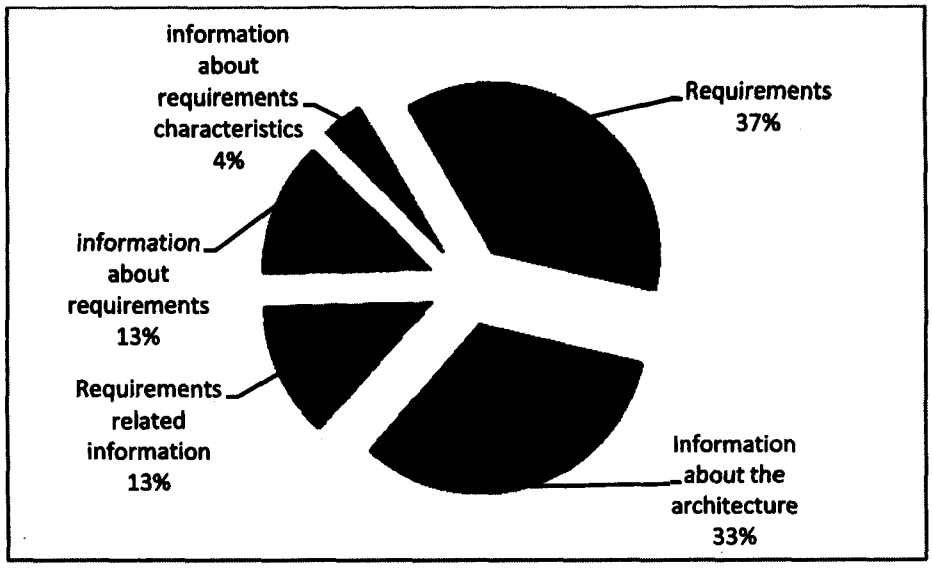

Figure 1 Percentage distribution of types of information fed-forward in the case study

There is a high proportion of feed-forward information which is based on requirements (37%) and information about the architecture (33%). The main deliverables for the projects were the requirements and software architecture which is why information about these was fed-forward the most. However, the results may be different in a development project where the code in the main deliverable. The findings calls for further research in a project where architecting, development and software testing were done to see whether actual "code snippets" or test cases get fed-forward.

Requirements (37%): A high amount of the information fed-forward was requirements as was reported by the members of the projects. They were asked also to explicitly state the particular requirements that they think was fed-forward. The participants were provided with a list of all the requirements that they had elicited till that particular session as a reference.

Example: A requirement being fed-forward during the project was seen in the very first session. The customer for project 2 was explaining the requirements to the requirements engineer. The requirements engineer, based on his previous experience of working in a similar project, asked the customer whether a web based system was needed to access the log of entry through the garage door for the whole day. To this, the customer mentioned that it is an important requirement that was supposed to be provided after 3 weeks because of the priority of the requirement. However, from an architectural perspective, this requirement would have required a considerable overhauling of the architecture if it were known 2 weeks later. Thus a requirement was collected from the customer by "Pull feed-forward".

Q1.2. *Who are the sources of feed-forward information?*

Q1.3. *Who are the recipients of feed-forward information?*

In the questionnaire provided to the project group members, if they mentioned that they received feed-forward from someone, they would have to explicitly state who or what the source was. On the contrary, if the project group members mentioned that they fed information forward to someone, they would have to explicitly state who they feed-forward the information to. It is possible that a project group member can claim as a source that they had feed-forwarded information but the recipient may not really count it as any sort of useful feed-forward information. To verify this, the recipient was shown the particular claim from the source and if it was not accepted by the recipient, it was rejected. Based on this, a path of communication of feed-forward information from source to recipient can be traced as shown in Table 15.

Table 15 Percentage distribution of the roles for the source and reception of feed-forward

Recipients (%)	Sources (%)		
	Customer	Requirements engineer	Software architect
Customer	0.00	4.35	21.74
Requirements engineer	26.09	0.00	0.00
Software architect	8.70	0.00	39.13

A plausible reason behind the high proportion of software architect to software architect feed-forward may be due to the interchangeable roles of the group members in each session (please see section 3.3.2). When a requirements engineer had some information to feed-forward to a software architect or a group of software architects, due to their interchangeable roles, the requirements engineer interprets the information from an architect's point of view and when the information gets fed-forward, it was recognized as a feed-forward from a software architect to another software architect. This also explains why there was no feed-forward from a requirements engineer to a software architect or from a software architect to a requirements engineer. More emphasis was placed in architecting (as it was a project in an architecting course) than detailed modeling of the requirements which, along with the other interpretations provided earlier, might contribute to the fact that there was no feed-forward from a requirements engineer to another requirements engineer.

From an agile point of view, the results clearly indicate intra team feed-forward. However, the result also calls for further research in an isolated project where there are no overlapping roles to check for feed-forward from one activity to another (RE, SA, Coding, Testing, etc.).

Overall, the communication path in the case study can be represented as in Figure 2.

Figure 2 Percentage communication of Feed-forward in the case study projects

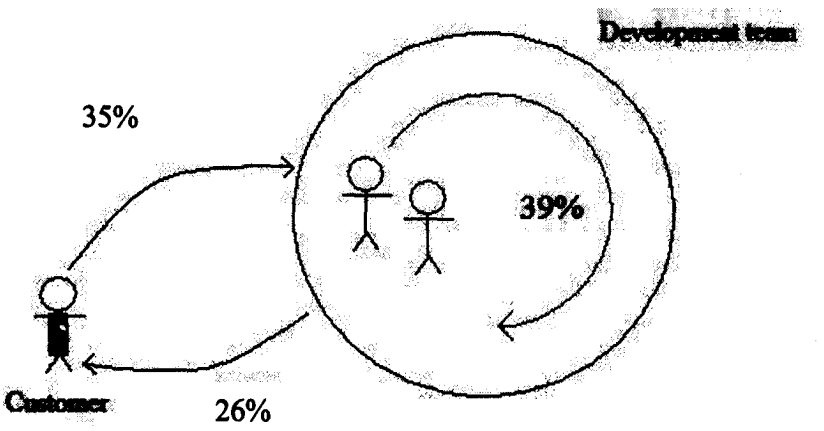

Figure 2 shows that a customer provides more feed-forward (35%) than they receive feed-forward (26%). On the contrary, the highest amount of feed-forward is within the development team (39%). It is possible that development team members provided more feedback to the customer of their work as compared to feed-forward. Whereas, within the team, the project members knew what information might be useful ahead of time for whom and so information was effectively fed-forward.

Q1.4 *What is the frequency of "pull" feed-forward as compared to "push" feed-forward?*

According to the definition of "pull" feed-forward and "push" feed-forward (please see Definitions section), the source participants were asked whether the information that they had fed-forward was done on their own accord (i.e., without a request from the recipient) The opposite of this case is "pull" feed-forward where the recipient were asked whether they had to request for the information that they had received as feed-forward. The aggregated percentage distribution from the four projects for "push" feed-forward and "pull" feed-forward is shown in Table 16.

Table 16 Percentage of push feed-forward and pull feed-forward in the case study

Type of feed-forward	Distribution (%)
Push	75
Pull	25

Push (75%): A majority of the feed-forward in the four projects has been "push" feed-forward. A plausible reason behind this could be that the recipients were busy doing their current tasks and were usually "late" anyway and so they did not have the luxury of thinking ahead in time. It was seen in Figure 2 that there was a lot of intra team feed-forward. Breakdown of the source and recipients of push feed-forward shows that approximately 80% of the sources of push feed-forward were software architects and the corresponding recipients were also, not surprisingly, software architects. Hence, there was a tendency to help out a fellow software architect by proactively providing them with information ahead of time that they may need.

Q2.1: *What aspects of a software project are affected by feed-forward information?*

Once the project members had identified the types of information that were feed-forwarded, they were also asked to mention the particular aspect of the project that they think the information had an effect on. As stated in section 3.4.2.4, all the different aspects mentioned in the list are rooted in literature (i.e., [36]). The list of aspects that the participants were asked to choose from is shown in Table 17.

Table 17 Potential list of aspect affected in the software projects of the case study

Aspect	Explanation
Cost	Whether the information could cause a potential saving of expense (denoted by effort)
Time	Whether the information could potentially save time and things could get done earlier/ on time instead of late
Quality	Whether it could enhance the quality of the architecture of the software system and potentially save much evolution and rework

The percentage distribution of the different aspects affected during the course of the projects is shown in Figure 3.

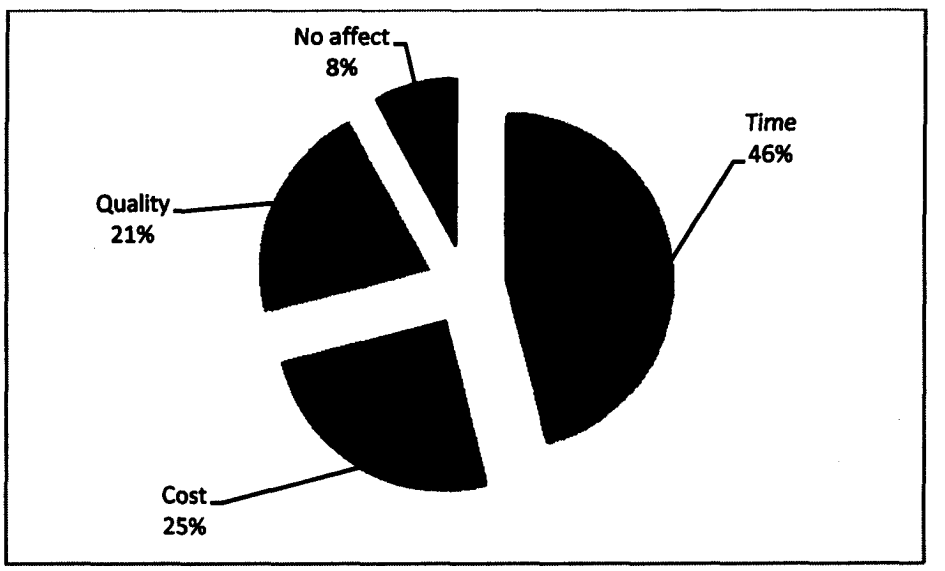

Figure 3 Percentage distribution of software aspects affected by feed-forward information in case study projects

Time (46%): When the participants selected the aspect that they thought was influenced by the feed-forward information, they were also asked to provide the rationale behind their selection. It is obvious that a major aspect influenced by getting information early would be time.

Cost (25%): This is the second major aspect influenced by feed-forward information. When information was fed-forward from a domain expert it obviously saved the effort that would have otherwise been spent on hours with another domain expert. Hence cost was an important aspect.

Since this was a class project and the work to be done in the project was to be evaluated, there were two main constraints in the project: time and quality. The project members had a clear idea about the timelines (project deadline) and the expected quality of the project was explained to them as a guideline rather than a reference. Whenever a project member got an opportunity for feed-forward that could have affect on time, it was done with much motivation. Even though all the project members had a good background in RE and SA they were not as experienced to rationalize the potential impact feed-forward information can have on the overall quality and cost of the software. This might be the reason why time was affected a lot more than quality. This result suggests that a constrained aspect that project members had experience in might be affected more by feed-forward information.

It is important to note that some participants reported feed-forward but also claimed that no aspect was influenced. In cases such as these the participants were also asked to provide their rationale behind their opinion. After going through the rationale it was found that even though information was fed-forward, it was not confirmed as to whether the information will be implemented. For example, in one of the projects the customer talked about a requirement on inter-banking support so that bank customers can transfer money from one bank account to another bank account through ATM. This requirement

was not part of the current architecture and the customer seemed confused about whether it would be implemented in the future. Eventually that requirement was never implemented and so even though there was a feed-forward, it was not put to any particular use.

Q2.2: *Which software artefacts are influenced by feed-forward information?*

Feed-forward could potentially have an influence on any software aretfact (e.g., software code, test-cases, change related information, etc.). For this study, the group members were required to complete two major steps of the software process-RE and SA. Since the study was conducted in an architecting course, architecting was the main focus of the project and requirements were not modeled in detail. For this reason, the main influence of feed-forward was observed to be in the architectural artefacts. The findings of these influence is shown in Figure 4. However, it is acknowledged that feed-forward can have an influence on artefacts other than simply the architectural artefacts.

Based on the type of information that was fed-forward, the participants were also asked to select the software artifact that was influenced by the feed-forward information. As in previous sections, the list of different architectural artefacts is rooted in literature (e.g. [5]). The list of architectural artefacts that the project members selected is shown in Table 18.

Table 18 Software architectural artefacts influenced by feed-forward information in case study

Software architectural artefacts	Explanation
Architectural Driver	A collection of functional, quality, and business requirements that shape an architecture [5]
Pattern	An architectural pattern is a description of element and relation types together with a set of constraints on how they may be used. [5]
Tactics	a design decision that influences the control of a quality attribute response. [5]
Decisions	A description of the set of architectural additions, subtractions and modifications to the software architecture, the rationale, and the design rules, design constraints and additional requirements that (partially) realize one or more requirements on a given architecture [14]
Architectural Element	Modules which encapsulate some functionality of the system [5]
Quality Scenario	Attribute specific requirements of a systems [5]
No Influence	This means that no architectural artefact was influenced

Figure 4 shows the percentage distribution of software artefacts influenced by feed-forward information.

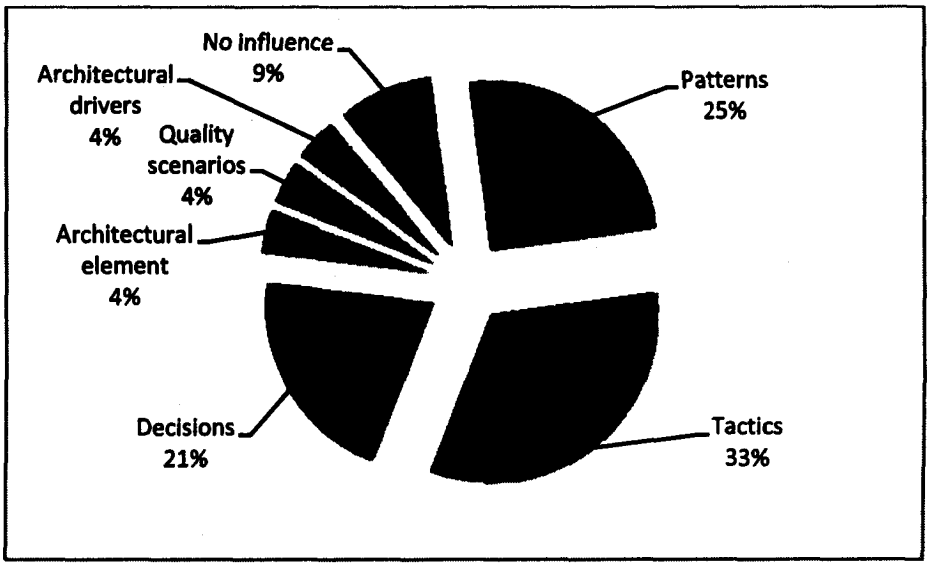

Figure 4 Percentage distributions of software architectural artefacts influenced by feed-forward in case study

A high proportion of the architectural artefacts affected by feed-forward were tactics and patterns. For all the projects, redundant tactics and well known architectural patterns were used. For example, in the garage door projects, there was redundancy of sensors and in the banking door projects there was redundancy of database. In project 1, the group members used façade which is a well known architectural pattern while in project 3 the group members used a combination of multitier and distributed application. Since low level design decisions were beyond the scope of the project only high level design decisions were taken. Tactics, patterns and decisions were in higher levels of abstraction whereas architectural driver, architectural elements and quality scenarios were modeled in more detail. The results from Figure 4 show that artefacts which are at a higher level of abstraction are influenced more by feed-forward information. A plausible explanation for this could be that enough details were not known regarding the artefacts at a lower level of abstraction to be fed-forward ahead of time.

4.1.2 Results from Industrial survey

As we have already said, the research question was sent to 29 professionals and through snowball sampling [12] the survey was referred to 12 more people. Out of this total of 41 people approx 78% of the participants, i.e., 32 software professionals completed the survey. Hence the response rate was 78%.

4.1.2.1 Findings

The first research question was the following:

Q1.1 *What are the different types of feed-forward information?*

Selected industry professionals with the necessary background were asked to select from a list of different types of information and select appropriate information(s) which has been fed-forward. The list of different types of information is shown in Table 19 and as was mentioned in section 3.4.2.4, each of the different type of information is rooted in literature (e.g., [18], [5] and [36]). The survey involved a larger diversity of software professionals and so more options were added to the potential list of types of information fed forward as compared to the case study projects (where the roles of the respondents were limited to customer, requirements engineer and software architect).

Table 19 List of types of Information fed forward as shown in the survey

Types of Information
Prospective requirements
Requirements
Rationale, priority, source and assumptions
Architectural artefacts

Other artefacts
Architectural relevance of requirements
Cost, effort and change related information
Process related information
Domain related information
Constraints
Information about artefact decomposition
Others

The percentage distribution of the type of information feed-forwarded by industry professionals is as follows. It may be noted that the aggregate of the percentages is not 100% because the participants were given the option to select more than one type of information as they felt necessary.

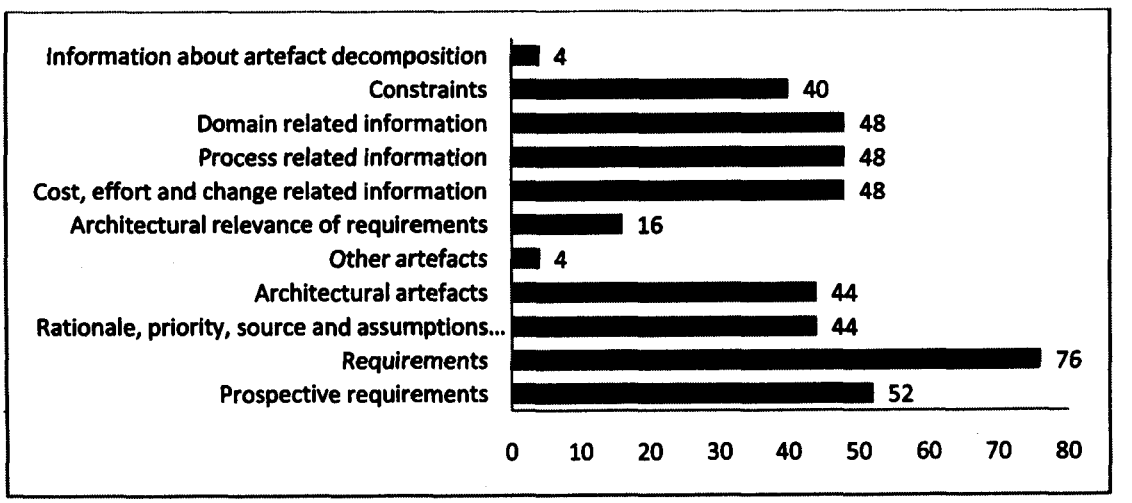

Figure 5 Percentage distributions of the different types of information as reported by respondents of survey

The major types of information fed-forward are:

- Requirements (76%)
- Prospective requirements (52%)

- Cost, effort and change related information (48%)
- Process related information (48%)

Just as in the case study projects, in the survey the software professionals also reported that requirements were the most popular type of information fed forward. The other most popular types of information were information related to requirements. And according to a lot of the respondents, process and domain related information was also fed forward. Since domain and process were not constraints in the case study projects, they were not found in the case study. However, these findings suggest that in industry, process and domain related information are also prominently fed forward.

Q1.2. *Who are the sources of feed-forward information?*

Q1.3. *Who are the recipients of feed-forward information?*

In the survey, the software professionals were also asked to select from a list of sources and recipients whom they think are the sources of the feed-forward information were and who the recipients were. The list of sources and recipients that the software professionals were asked to choose from are shown in Table 20.

Table 20 List of sources/ recipients of feed-forward information as reported from survey respondents

Source/ Recipients of feed-forward information
Customers
Requirements engineers
Software architects

Developers
Managers
Others

Again the aggregate of the percentage distributions for both the sources and the recipients is not 100% because the participants had the option to choose more than one source and more than one recipient.

The percentage distribution of the sources of feed-forward is shown in Figure 6.

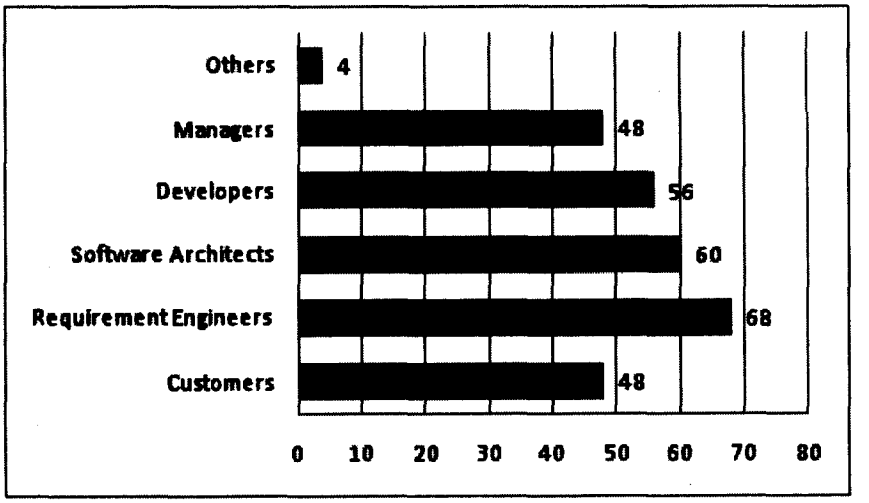

Figure 6 Percentage distributions of different sources of feed-forward as reported by respondents of survey

The major sources of feed-forward are:

- Requirements engineers (68%)
- Software architects (60%)
- Developers (56%)

The percentage distribution of the recipients of feed-forward is shown in Figure 7

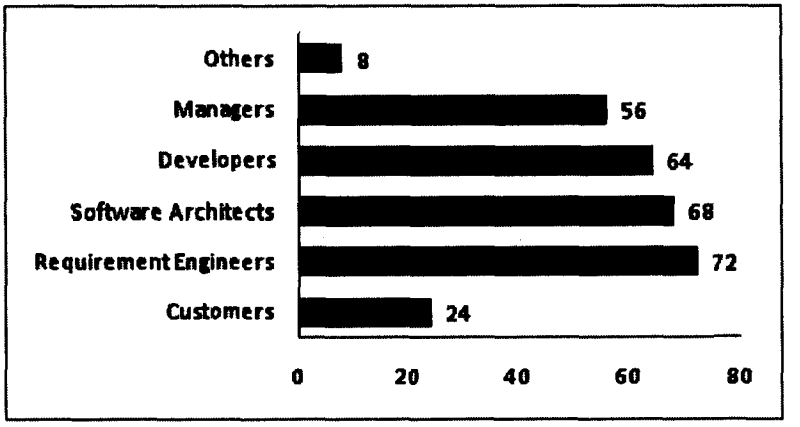

Figure 7 Percentage distributions of different recipients of feed-forward as reported by respondents of survey

The major recipients of feed-forward are:

- Requirements engineer (72%)
- Software architect (68%)
- Developers (64%)

It is interesting to note that while a customer was a moderately high source of feed-forward information (48%), they were a comparatively low recipient of feed-forward information. This means that while a customer does feed-forward a lot of information, a software project member more often feeds back information to a customer. There is also high consistent feed-forward between software project members. This can be seen from the fact that requirements engineers, software architects and developers were always consistently high sources and recipients of feed-forward information. Thus, it is more common to feed-forward information to internal stakeholders than external stakeholders.

Q2.1: *What aspects of a software project are affected by feed-forward information?*

The software professionals were asked to select from a list of aspects the aspects of a software project that they think are affected by feed-forward information. Table 21 shows a list of aspects reported by the respondents of the survey. As in the previous section, the options for the aspects are all rooted in literature (i.e., [36]).

Again the aggregate of the percentage distributions for aspects is not 100% because the participants had the option to choose more than one aspect.

Table 21 List of Aspects reported by survey respondents

Aspect affected by feed-forward information
Cost
Quality
Time
Requirements engineer and Software architect
Morale
Designers, coders, testers, integrators
Management and Quality assurance
Customers
Process improvement agents
Process
Resources
Others

The percentage distribution of the aspects is shown in Figure 8

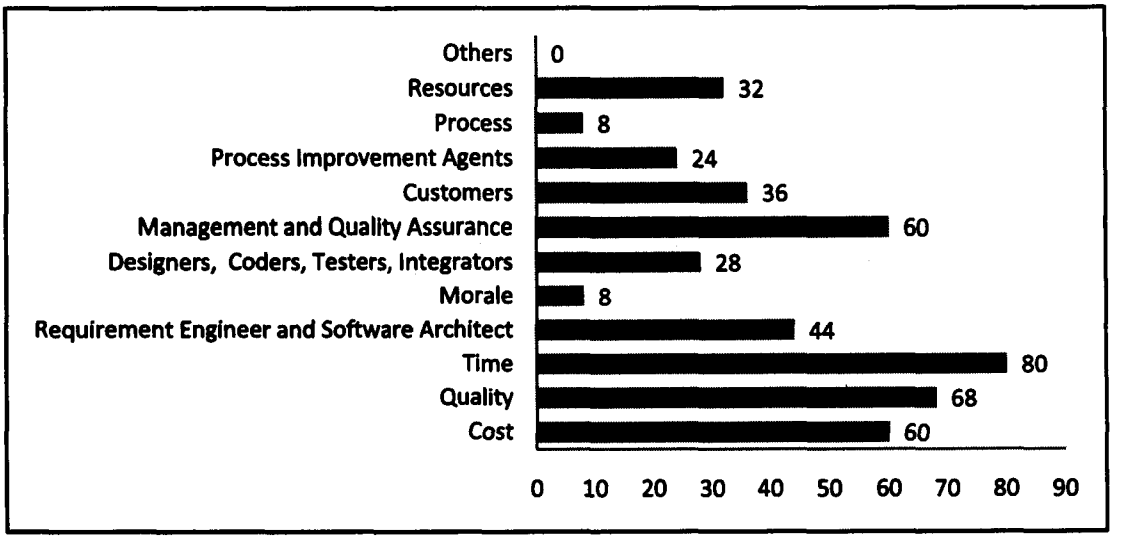

Figure 8 Percentage distributions of different aspects affected by feed-forward as reported by survey respondents

The major aspects affected by feed-forward information are:

- Time (80%)
- Cost (68%)
- Quality (60%)
- Management and Quality assurance (60%)

These findings are in unison with the findings of the case study in that time, cost and quality were also noted as important aspects affected by feed-forward information. However, other aspects such a management and quality assurance were not present in the case study and hence provide a more realistic picture of the aspects affected by feed-forward in industry. This result indicates that process aspects such as management and quality assurance are also affected by feed-forward along with product aspects such as time, cost, quality, etc.

Q3: What is the current state of the practice of feed-forward?

To determine the possible implications of feed-forward in the industry, it is important to analyze what the current state of the practice of feed-forward is like. Three things were measured from the survey to determine the answer to this question.

M3.1: Frequency of different levels ("Never", "Rarely", "Sometimes", "Most of the time", "Always") of practice of feed-forward

Here the different levels refer to the ordinal scale (Never, Rarely, Sometimes, Most of the time and Always) [12] with an extra scale called "Not Sure" added. This was the most preliminary question added and based on the answer to the question a participant would have to answer differently for the rest of the sections of the survey. The professionals who chose "Never" and "Rarely" had to answer an extra question regarding the reason behind absence of feed-forward in their organization. Since these professionals have very little experience with feed-forward, questions such as type of information fed forward and aspects affected where asked in the form of opinions (e.g., what type of information do you think could be fed forward?).The actual frequency of the response from the survey participants is shown in [39]. Table 22 shows the aggregation of the answers provided by the 32 participants regarding whether feed-forward is practiced in their organization or not. The later rows show the median and the 95% confidence intervals for the answers. Since the data was ordinal, a median was taken. A median of 2.999993 as shown in Table 22 means that the median frequency of usage of feed-forward in industry is "Sometimes". This implies that information is fed forward sometimes in industry but not very prominently. The low frequency of "Always" and high frequency of "Not Sure" and "Never" also shows that feed-forward is not well recognized in industry. A 95% confidence interval (CI) was also found of the median value. If the 95% CI does not

include zero (i.e., either the upper and lower bound of CI are positive or both of them are negative) then the medians are considered statistically reliable [38]. Hence, the median for the frequency of usage is statistically reliable.

Table 22 Percentage frequency of practice of feed-forward in organizations as reported by respondents of survey

	Number of response	Percentage
Never	4	12.50
Rarely	3	9.38
Sometimes	17	53.13
Most of the time	3	9.38
Always	1	3.13
Not sure	4	12.50
Median of ordinal scale	**2.999993 ≈ "sometimes"**	
95 % Confidence interval (+)	**3.000034**	
95 % Confidence interval (-)	**2.499967**	

M3.3: Frequency of different reasons behind the absence of feed-forward in software projects.

7 of the total 32 participants answered either "never" or "rarely". These respondents were asked to provide their opinion regarding why feed-forward was not practiced in their organization. Upon breakdown and categorization of the answers, the following major reasons were found as cited by the respondents as the reasons behind the absence of feed-forward in organizations:

- The structure of their organization does not support feed-forward (42.86%)
- The participant has never heard of feed-forward before (28.57%)

- Feed-forward was not considered important for their organization (28.57%)

M3.2: Proportion of feed-forward that is being useful for software projects.

Irrespective of the frequency of feed-forward of software projects, respondents were asked, based on their understanding of feed-forward, to provide their opinion as to whether they think that feed-forward will be useful for software projects in their organizations. The three options for the respondents were "Yes", "No" and "Not Sure". The percentage distribution of the response from the participants is shown in Figure 9.

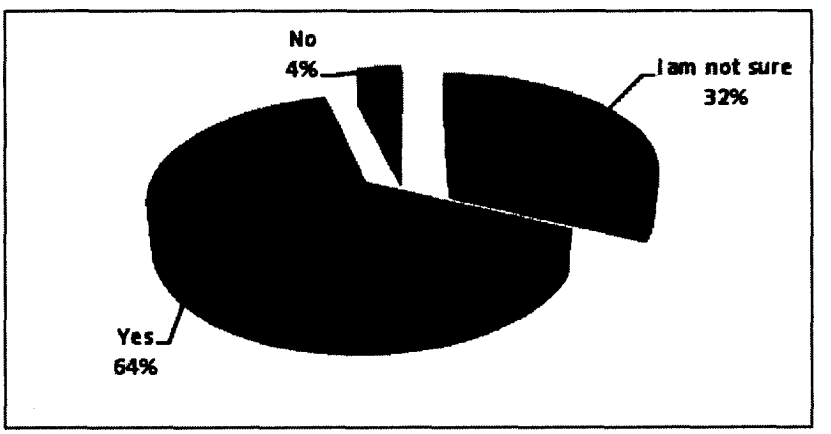

Figure 9 Percentage distribution of usefulness of feed-forward in software projects as reported by survey respondents

As can be seen from the percentage distribution, majority of the respondents reported that feed-forward is or could be useful in software projects in organizations. The fact that feed-forward is unknown to some of the participants is shown by the 32% response of "I don't know". Only a few respondents didn't find feed-forward to be useful for their organization. However, it is important to analyze the reasons why the respondents perceived feed-forward as not useful. One rationale of a respondent against feed-forward was:

"I think it's information overload. In our daily work we already get too many notifications and emails on topics that only remotely affect our work. It's difficult to pin-point the feed-forwarded information that is relevant to our work because we are so focused on the work being done at the moment. Too often the information pertaining to some future piece of work will just get lost in one's inbox, and then you have to dig through the old emails when that information is actually needed."

It is clear from this response that one of the possible disadvantages of feed-forward is information overload. It was also mentioned in [1] by Agresti that if a feed-forward apparatus was sensitive to every disturbance, then there would a constant fluctuation in software estimation because every event would signal an update to the estimates. This cautions us about what information should be fed forward. Feeding forward too much information will result in information overload such that they will get lost and will be put to no use ahead of time. Thus, this comment signifies the necessity of knowing the characteristics and impact of feed-forward information so that a better idea can be gained about which information would be most useful when fed-forward.

Another rationale from another respondent was as follows:

"Developers are told pretty specifically the tasks that are expected by management, regardless of what feed-forward information they may receive from other stakeholders. I find that businesses will not adjust their work processes unless they know that will be getting paid for it and it has value (i.e., new or changed requirements being fed-forward have no impact until they are contractually accepted). Also, I'm not sure if feed-forward

is something you 'do or not do', it is a more natural phenomenon that occurs given a particular situation in a project... ".

This comment suggests that for feed-forward to be incorporated into a software process, it is necessary to promote feed-forward in an organization. Unless, a software professional is encouraged to feed-forward information, they would not always do so. As was already seen in the results in section 4.1.2.1, information is fed forward sometimes in an organization but it is not done frequently because it is not an established activity. If the usefulness of feed-forward is established, it can then be accepted explicitly as an activity in software processes in industry and organizations can motivate their employees to feed-forward information ahead of time.

4.2 Composite Analysis of results and interpretations

Q1.1. *What are the different types of feed-forward information?*

Table 23 Composite percentage distribution of the different types of feed-forward information in case study and survey

Types of information in case study	Percentage	Types of information in Survey	Cumulative Percentage
Requirements	37%	Requirements (76%), Prospective requirements (52%)	78%
Requirement(s) related information such as assumption, rationale, relationship between requirements	13%	Rationale, priority, source and assumptions (44%)	13%
Information about requirements such as requirement decomposition, refinement, etc	13%	Information about artefact decomposition and refinement (4%), Constraints (40%)	56%

		Domain related information (48%)	
Information about requirement characteristics such as cost of implementation, implementation effort, resources needed, prioritization, change-related info, etc	4%	Cost, effort and change related information (48%)	4%
Information about the architecture	33%	Architectural artefacts (44%),	53.13%
		Architectural relevance of requirements (16%),	
		Information about artifact decomposition and refinement(4%)	

Table 23 shows the percentage distribution of the types of information from the case study and survey. As there were more options in the survey as compared to the case study, certain options in the survey are grouped to represent a particular type of information comparable to the case study. In this case, requirements and prospective requirements were grouped to compare with requirements. Information about artefact decomposition and refinement, constraints and domain related information were grouped to compare with information about decomposition, refinement, etc. Architectural artefacts, Architectural relevance of requirements and information about artifact decomposition and refinement were grouped to compare with information about the architecture. The cumulative percentage of the groups were taken (individual percentages are mentioned in brackets beside each individual types in the groups). In both the case study and the survey, requirements are the major type of information fed-forward. Information about the architecture was also a major type of information fed-forward. However, while information about requirements such as requirement decomposition, refinement, etc was not a major type of information fed forward in the case study, it was the second highest type in the survey. Upon closer inspection, it will be noticed that

individual feed-forward about decomposition and refinement type of information was low in both the case study and the survey and constraints and domain related information reported in the survey was comparatively higher. Constraint and domain related information was not dealt with in detail in the case study which is why the feed-forward of information related to these were low in the case study. However, its higher frequency in the survey results give us a better idea about the type of information that is fed-forward in industry level projects where domain related information and constraints are important factors.

Q1.2. *Who are the sources of feed-forward information?*

Q1.3. *Who are the recipients of feed-forward information?*

Table 24 Comparison between the source and recipients of case study and survey

	case study		survey	
	Mutually exclusive percentage		Overlapping percentage	
	Source	Recipients	Source	Recipients
Customers	34.78	26.09	48	24
Requirements engineers	4.35	26.09	68	72
Software architects	60.87	47.83	60	68
Developers	-	-	56	64
Managers	-	-	48	56
Others	-	-	4	8

As there were three roles recognized in the case study, the other rows of roles that are present in the survey results are missing from the results of the case study.

From the findings of the case study and the survey, it was seen that in both the case study projects and in the survey, customers fed forward more information than they received as feed-forward. Other than the common roles of customer, software architect and requirements engineer, Table 24 also shows that developers and managers are also major source and recipients of feed-forward information. Thus the findings provide a clearer idea about the sources and recipients of feed-forward in industry where the main deliverable is the software and a large number of stakeholders are involved.

Q2.1 What aspects of a software project are affected by feed-forward information?

Table 25 Comparison between the aspects affected in case study and survey

Aspect affected in case study		**Aspect affected in survey**	
Name of aspect	**Mutually exclusive percentage**	**Name of aspect**	**Overlapping *percentage***
Cost	45.83	Cost	60
Quality	25.00	Quality	68
Time	20.83	Time	80
-	-	requirements engineer and software architect	44
-	-	morale	8
-	-	designers, coders, testers, integrators	28
-	-	management and quality assurance	60
-	-	Customers	36
-	-	process improvement agents	24
-	-	Process	8
-	-	Resources	32

-	-	Others	0
No affect	8.33	-	-

From the findings of the case study, it was seen that feed-forward information has an effect on product aspects such as cost, quality, time (here time refers to time of delivery of product), etc. However, there was no significant affect in process aspects. However, the findings from the survey provide a clearer picture of industry process aspects (such as management and quality assurance) that are affected by feed-forward information. Thus, the findings from this study shows that feed-forward has an impact on software process aspects as well.

Chapter 5. Implications

This chapter discusses the implication of the study in various areas. Section 5.1 discusses the implications in industry. Section 5.2 discusses the implications of the results in further empirical work.

5.1 Implications in Industry

The results from Table 22 shows that feed-forward is practiced "sometimes" in organizations and that while many professionals are not aware feed-forward, it is considered by many as a useful activity (please see Figure 9). These findings can act as a motivation for the incorporation or enhancement of feed-forward in software processes in industry. Knowing the characteristics and impact of feed-forward has implications in industry so that software professionals have a better idea about which information would be most effective as feed-forward. In order to understand the effect of feed-forward information in a software process, empirical investigations such as case studies or control studies could be conducted. Even though in principle a control study can be conducted in industry in reality it would be extremely difficult. This is due to the unavailability of equivalent projects and the inability to impose research control (i.e., feed-forward in software engineering processes). However, it is possible to do archival analysis of software projects to get an idea about feed-forward in industry software projects and whether it had an impact on software aspects and influenced software artefacts. The GQM developed for this study can also be used as a primary template for the archival analysis of feed-forward in software processes in industry.

5.2 Further Empirical Work

Based on the findings of the exploratory study, the following emergent hypotheses can be raised:

H1: The constrained aspects of a project will have more impact from feed-forward in a software project as compared to the other aspects of a project.

Results from Figure 3, Figure 5 and Table 25 support the hypothesis stated above in that information is fed forward such that it has an effect on aspects that are constraints in the project. In the case study, since time and quality were constraints for the projects, they were affected more by feed-forward as compared to the other aspects. The results from the survey supported the results from the case study and it was also seen that key constraints to a project such as time, quality and cost were affected more by feed-forward information as compared to other aspects of a software project.

H2: Software artefacts at a higher level of abstraction are affected more by feed-forward information as compared to artefacts at a lower level of abstraction.

The findings from Figure 4 shows that architectural artefacts that were implemented at a higher level of abstraction were affected more by feed-forward as compared to artefacts at a lower level of abstraction. While the findings of Figure 4 may seem intuitive, further testing of this hypothesis in other areas of software engineering (such as development, testing, etc.) as well as other software aspects (such as software codes, test cases, etc.) would only confirm as to whether this finding is generalizable across other areas of software engineering. Also further testing of the hypothesis in different domains and project contexts would also help confirm whether the results are generalizable across different setting and indicate the variance across the different settings.

When software professionals were asked the reasons behind the absence of feed-forward in software organizations, the most common reasoning was that their organizational

structure doesn't support feed-forward. This finding can be used to formulate a research question for future research:

Q1: Which organizational structure is most suitable for feed-forward?

The answer to this question can help researchers decide whether feed-forward will be suitable for their organization or not.

Chapter 6. Limitations, Future Work and Conclusions

This chapter will describe the limitations, future work and conclusions. Section 6.1 will discuss the limitations and future work that can be generated from this work and section 6.2 will make conclusions based on all the findings.

6.1 Limitations and Future work

To the best of the author's knowledge, this study was a first of its kind study of feed-forward in RE and SA. While these findings contribute new scientific knowledge concerning feed-forward in RE and SA, it is important to note that the survey and case study were both exploratory studies. Significant as it is, caution is advised when making business or project decisions based on the findings of this foundation study alone. Confirmatory studies in other areas of software engineering and contexts are encouraged to help build a grounded theory on feed-forward in software engineering.

In this empirical study, the impact of feed-forward on different software project aspects and influence of feed-forward on different architectural artefacts were identified. But it is also important to determine the *extent* of the impact of feed-forward information.. This aspect is a limitation of this study.

However, it is important to mention that data from the four projects was captured using video and audio but due to time constraint could not be analyzed. Thus, as part of a continuation of this study, further analysis of the projects will be done using the multimedia data. Based on the data, a comparison can be made between the amount of

feedback and feed-forward in the four projects and an analysis can be done on the extent of impact of feed-forward information on software artefacts and aspects.

6.2 Conclusions

While providing feedback and communication across the various areas of software engineering has been discussed several times before (e.g. in [9] and [21]), not many papers have discussed feed-forward in software engineering [26] [21] and [1]. And while feed-forward has been used with or instead of feedback in many non-software engineering domains, there has not been any particular research on the properties of feed-forward in RE and SA. In this paper, we described two empirical studies including a case study and a survey to determine the characteristics, impact and state of the practice of feed-forward in software engineering.

In summary, it was found that number of feed-forward is influenced by the number of requirements and the application domain of the system (please see Table 13). The most common type of information fed forward were requirements, prospective requirements, information about the architecture and process and domain related information (please see Table 23). It was interpreted that the type of information fed forward in a project relies on the deliverables of the project (please see Figure 1). It was found that intra team feed-forward was more common and external stakeholders such as customers did not receive as many information as feed-forward as they had provided (please see Figure 2). Aspects such as time, quality and cost were the major aspects affected by feed-forward information and it was seen that such aspects which act as constraints for a software project are most affected (please see Table 25). It was found that architectural artefacts such as tactics and patterns were most affected by feed-forward information (please see Figure 4) and architectural artefacts implemented at a higher level of abstraction were influenced most by feed-forward information. Factors such as occupation with current task and no time to think "ahead of time" attributed to the high percentage of "push feed-forward" as compared to "pull feed-forward". While the activity of feed-forward is not

prominent in software industry, the findings suggests that feed-forward is practiced "sometimes" in organizations while many of other professionals are not aware of it (please see Table 22). Reasons attributing to the absence of feed-forward in software organizations include inability of organization structure to support feed-forward, zero knowledge on feed-forward and perceived insignificance of feed-forward. However, majority of software professionals acknowledge the usefulness of feed-forward (please see Figure 9). Those who don't consider feed-forward to be useful cited reasons such as information overflow and lack of motivation from organization.

It is expected that this study can act as a stepping stone towards the conduction of further research on feed-forward in software engineering.

References:

[1] Agresti, W. W. (2006). A Feedforward Capability to Improve Software Reestimation. In N. H. Madhavji, J. C. Fernández-Ramil, & D. E. Perry (Eds.), *Software Evolution and Feedback* (pp. 443-458). John Wiley & Sons.

[2] Ambler, S. W. (n.d.). *Agile Architecture: Strategies for Scaling Agile Development.* Retrieved November 2009, from Agile Modeling (AM) Home Page: Effective Practices for Modeling and Documentation: http://www.agilemodeling.com/essays/agileArchitecture.htm

[3] Ambler, S. W. (n.d.). *Introduction to Change Cases.* Retrieved November 2009, from Agile Modeling (AM) Home Page: Effective Practices for Modeling and Documentation: http://www.agilemodeling.com/artifacts/changeCase.htm

[4] Basili, V. R., Caldiera, G., & Rombach, H. (1994). The Goal Question Metric Approach. In J. J. Marciniak (Ed.), *Encyclopedia of Software Engineering* (2nd ed., Vol. 2, pp. 528-532). John Wiley & Sons, Inc.

[5] Bass, L., Clements, P., & Andkazman, R. (2003). *Software Architectures in Practice* (2nd ed.). Boston,MA: Pearson Education Inc.

[6] Brosilow, C. B., & Joseph, B. (2002). *Techniques of model-based control.* Upper Saddle River, NJ: Prentice-Hall.

[7] *CakePHP: the rapid development php framework.* (n.d.). Retrieved November 2009, from CakePHP: the rapid development php framework: http://cakephp.org/

[8] Cordesman, A. H. (2002). Threat Assessment. In J. G. Cordesman, *Cyber-threats, information warfare, and critical infrastructure protection: defending the U.S. homeland* (pp. 35-37). Westport, CT: Praeger Publishers.

[9] Damian, D., Izquierdo, L., Singer, J., & Kwan, I. (2007). Awareness in the Wild: Why Communication Breakdowns Occur. *International Conference on Global Software Engineering* (pp. 81-90). Munich: IEEE Computer Society.

[10] Goldsmith, M. (2002). Try Feedforward Instead of Feedback. In J. LeBoeuf, F. Hesselbein, & F. Hesselbein (Ed.), *Leader to Leader* (Vol. 25).

[11] Grant, T., & West, D. (n.d.). *Agile Adoption In The Real World.* Retrieved November 2009, from Forrester Research: http://www.forrester.com/rb/teleconference/agile_adoption_in_real_world/q/id/5881/t/1

[12] Grinnell, R. M., & Unrau, Y. A. (2008). *Social work research and evaluation: foundations of evidence-based practice.* New York: Oxford University Press.

[13] Höst, M., Regnell, B., & Wohlin, C. (2000). Using Students as Subjects—A Comparative Study ofStudents and Professionals in Lead-Time Impact Assessment. *Empirical Software Engineering , 5* (3), 201 - 214.

[14] Jansen, A., & Bosch, J. (2005). Software Architecture as a Set of Architectural Design Decisions. *Working IEEE/IFIP Conference on Software Architecture* (pp. 109--120). Pennsylvania, USA: IEEE Computer Society.

[15] Kimble, G. A. (1967). *Foundations of Conditioning and Learning.* New York: Appleton-Century-Crofts, Inc.

[16] Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., et al. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering , 28* (8), 721 - 734.

[17] Kitchenham, B., & Pfleeger, S. L. (2001–2002). Principles of survey research – parts 1–6. *SIGSOFT Software Engineering Notes , 26,27* (6,1,2,3,5), 16–18,18–20,20–24,20–23,17–20.

[18] Kotonya, G., & Sommerville, I. (1998). *Requirements Engineering: Process and Techniques.* John Wiley & Sons.

[19] Kwan, I., Damian, D., & Storey, M. (2006). Visualizing a Requirements-centred Social Network to Maintain Awareness Within Development Teams. *International workshop on Requirements Engineering Visualization* (p. 7). IEEE Computer Society.

[20] LaLonde, J. (2008). *China becomes world's 4th largest Software producer.* Retrieved November 2009, from China Bits: http://bits.typepad.com/chinabits/2008/06/china-becomes-w.html

[21] Lehman, M. M., & Ramil, J. F. (2002). Software Evolution and Software Evolution Processes. *Annals of Software Engineering , 14* (1-4), 275 - 309.

[22] Lethbridge, T. C., Sim, S. E., & Singer, J. (2005). Studying Software Engineers: Data Collection Techniques for Software Field Studies. *Empirical Software Engineering , 10* (3), 311 - 341.

[23] Lung, C.-H. Z. (2005). Reflection on Software Architecture Practices - What Works, What Remains to Be Seen, and What Are the Gaps. *Working IEEE/IFIP Conference on Software Architecture* (pp. 221 - 222). Pittsburgh, Pennsylvania, USA: IEEE Computer Society.

[24] Mauk, M., Medina, J., Nores, W., & Ohyama, T. (2000). Cerebellar function: Coordination, learning or timing? *Current Biology* , 10 (14), R522–R525.

[25] McGarry, F., Pajersk, R., Page, G., Waligora, S., Basili, V., & Zelkowitz, M. (1994). *Software Process Improvement in the NASA Software Engineering Laboratory.* Technical.

[26] Miller, J. A., Ferrari, R., & Madhavji, N. (2009). An Exploratory Study of Architectural Effects on Requirements Decisions. *Journal of System and Software* , Submitted.

[27] *MySQL :: The world's most popular open source database.* (n.d.). Retrieved November 2009, from MySQL :: The world's most popular open source database: http://www.mysql.com/

[28] Penn, P. (1985, September). Feed-Forward: Future Questions, Future Maps. *Family Process* , 299-310(12).

[29] *PHP: Hypertext Preprocessor.* (n.d.). Retrieved November 2009, from PHP: Hypertext Preprocessor: http://php.net/index.php

[30] Podvig, P. (2002). History and the Current Status of the Russian Early-Warning System. *Science and Global Security , 10* (1), 21-60.

[31] Fourati, F., & Chtourou, M. (2007). A greenhouse control with feed-forward and recurrent neural networks. Simulation Modelling Practice and Theory , 1016 - 1028.

[32] Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering , 14* (2), 131-164.

[33] Scholz, R. W., & Tietje, O. (2002). *Embedded case study methods: integrating quantitative and qualitative knowledge.* Thousand Oaks, CA: Sage.

[34] Scheffer, M., Bascompte, J., Brock, W. A., Brovkin, V., Carpenter, S. R., Dakos, V., Held, H., van Nes, E. H., Rietkerk, M., and Sugihara, G. (2009). Early-warning signals for critical transitions. *Nature*, 461(7260):53-59.

[35] Sim, S. E., Perry, D. E., & Easterbrook, S. M. (2006). Case Studies for Software Engineers. *International Conference on Software Engineering* (pp. 1045 - 1046). Shanghai, China: ACM.

[36] Sommerville, I. (2007). *Software Engineering* (8th ed.). Essex, UK: Pearson Education Ltd.

[37] Son, I. S., Oh, K. J., Kim, T. Y., & Kim, D. H. (2009). An early warning system for global institutional investors at emerging stock markets based on machine learning forecasting. *Expert Systems with Applications: An International Journal , 36* (3), 4951-4957.

[38] *StatisticalHelp from StatsDirect.* (n.d.). Retrieved November 2009, from StatsDirect: Software to improve statistical practice: http://www.statsdirect.com/help/statsdirect.htm

[39] Survey on Documentation and Feed forward information in Software Projects (http://survey.obviousdesign.net/)

[40] Trochim, W. M. (2006). *Research Methods Knowledge Base.* Retrieved November 2009, from Social Research Methods: http://www.socialresearchmethods.net/

[41] Urdan, T. C. (2005). Statistical Significances, effect size and confidence intervals. In T. C. Urdan, *Statistics in Plain English* (2nd ed., pp. 57-72). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

[42] Wheeler, J. (2009, January). *Overcoming the Software Developer Experience Gap in China.* Retrieved November 2009, from China software outsourcing and Chinese offshore development blog: http://www.daoofoutsourcing.com/software-developer-experience-in-china/

[43] Wojcik, R., Bachmann, F., Bass, L., Clements, P. C., Merson, P., Nord, R., et al. (2006). *Attribute-Driven Design (ADD), Version 2.0.* Technical.

[44] Ye, X. (n.d.). *The introducton of automatic process control.* Retrieved November 2009, from http://zdh.course.ecustmde.com/exres/info/jaxz/0800.ppt

[45] Yin, R. K. (2003). *Case Study Research Design and Methods* (3rd ed.). Thousand Oaks, CA: SAGE Publications, Inc.

Appendix A: Role of a feed-forward model

The role of a feed-forward model to address project risks or Warning Signal of Problem

Topic area of risk	Project risk or Warning Signal of Problem	How a feedforward model would provide support to risk management
Requirements and specifications	Number of TBD (To Be Determined) requirements higher than norm or not declining	Make projections from any parameters that relate to the extent to which users and customers know what they want; and make those projections in the same form as the data being used in the project, namely, tell the project manager the expected number of TBDs at various points in development. In this way, the manager could see how the actual TBDs compare to the projected number to know if the estimate is realistic or needs revision. This problem area from is based on their practice of explicitly recording and tracking TBD requirements.
	High number of specification modifications received versus number completed; continuing stream of requirement changes	Make projections from the parameters it is using for requirements volatility, and make those projections in terms of the data being used in the project, namely, expected number of changes to requirements and specifications at various points in development. In this way, the manager could see how the actual changes compare to the projected number to know if the estimate is realistic or needs revision.

	Developing the wrong software functions and user interface	Prompt the manager (e.g. on a biweekly frequency) with related questions, such as, 'When is the last time the current user interface and system functionality were checked with the customer?' 'Does the development process provide prototypes or other ways to gain early customer buy-in and agreement?' 'How do you know this is what your customer wants?'
	Gold plating	Prompt the manager (triggered when requirements are added or functionality is reviewed) with key questions, 'What is the source of this feature or requirement – does the customer really want this?' 'Is this new requirement required by the contract?' 'Is the customer paying extra for this – for example, via a change order?'
	Number of completed units increases dramatically prior to the scheduled end of a build/release (the 'miracle finish') and/or effort drops dramatically just after a milestone is reached	Issue alerts at each milestone for the manager to check if a miracle finishoccurred orifeffort dramatically drops off after the milestone, and, if so, to check the quality of the work completed during the miracle finish and the reason for the drop off.
	Testing phase was significantly compressed	Make projections on the needed length of the testing phase; alert the manager to the importance of not compressing the testing phase.
	The number of errors found during	Make projections from the model
	testing is below the norm	on the expected number of errors to be found; issue alerts to be on guard to check the actual error data, and, if lower than expected, to ensure that adequate resources and an effective process are being applied to testing.
	More than one person controls the configuration	Generate questions periodically for the manager to check on the number of people controlling the product configuration.

	Capabilities originally planned for one time period are moved to a later time period	Make projections from the model on the estimated functionality (e.g. in function points or features) that is planned for each time period or product release; generate alerts for the manager to check actuals to ensure the plan is followed and to take action.
	Real-time performance shortfalls	Make projections from the model if the system to be built is identified as one with real-time performance requirements, so the manager is alerted early in the project to conduct off-line studies (e.g. simulations and lab exercises) to ensure that real-time requirements will ultimately be met as the product takes form.
	'Corrected' errors reappear	Issue an alert to check on the frequency of errors reappearing, from the error data being reported.
	Continual schedule slippage	Make projections on expected numbers of components designed, coded, tested or integrated, and pose advisories to the manager to check against actual values.
	Personnel shortfalls and turnover	Monitor actively the size and composition of the team at all times; alert management when changes occur based on comparison to projected size and composition.
	Unrealistic schedules and budgets	Prompt managers to check realism of parameter values; if the model parameters continue to be accurate, use the model to make schedule and budget projections; if actual resources are not sufficient based on projections, issue prompts to raise visibility of lack of realism

	Straining computer-science capabilities	Monitor action items from reviews for indications that there are science and technology shortfalls; prompt manager to consider review by external expert panel to get independent view of extent to which project is pushing stat-of-the-art and relying on unproven technology.
	Change or decrease in planned use of methods or procedures occurs	Make projections from model parameters on expected development platform, tools and maturity of development process; prompt manager to compare these projections to reality.
	Shortfalls in externally furnished components and tasks	Monitor receipt and integration of externally furnished components; monitor completion of externally furnished tasks; compare to projected dates and issue alerts if differences exist.

Appendix B: Survey Questionnaire

The first section of the survey consisted of six questions that were intended to determine the background of the respondents. The questions in the third section were meant to collect data for measuring the metrics for this study. This appendix focuses on questions of the second section. The complete questionnaire can be found at [39].

Title: A Survey on Documentation and Feed forward information in Software Projects.

Questions about Feed-forward

Question 15: Do you think feed-forward is practised in software projects at your organization?

- Never
- Rarely
- Sometimes (Jump to question 17)
- Most of the Time (Jump to question 17)
- Always (Jump to question 17)

Question 16: Which of the following do you think are the reasons behind the absence of feed-forward in software projects at your organization? (Please select all that apply)

- Organizational structure does not support
- Never heard of it before
- Not considered important
- Requirement Engineers and Software Architects work isolated from one another with little room for communication

- Requirement Engineers and Software Architects work in an integrated environment such that feed-forward is not necessary
- Others (Please specify)

Question 17: Do you think that feed-forward might be/is useful in software projects at your organization?

- I am not sure
- Yes
- No

Additional Comment :

Question 18: Which of the following types of information could be/is being fed-forward? (Please select all that apply)

- Prospective requirements
- Requirements
- Rationale, priority, source and assumptions behind requirements
- Architectural artefacts
- Other artefacts (Please specify)
- Architectural relevance of requirements
- Cost, effort and change related information
- Process related information
- Domain related information
- Constraints

- Information about artefact decomposition and refinement
- Others (Please specify)

Question 19: Which of the following aspects of a project can be/is being influenced by that fed-forward information? (Please select all that apply)

- Cost
- Quality
- Time
- Requirement Engineer and Software Architect
- Designers, Coders, Testers, Integrators
- Management and Quality Assurance
- Customers
- Process Improvement Agents
- Morale
- Process
- Resources
- Others (please specify)

Question 20: Which of the following stakeholders do you think could be/are the sources of the feed-forward information? (Please select all that apply)

- Customers
- Requirements Engineers

- Software Architects
- Developers
- Managers
- Others (please specify)

Question 21: If a stakeholder has some information that can be fed-forward, how often would they /do they pass the information on to appropriate recipients even if these recipients didn't request for it?

- Never
- Rarely
- Sometimes
- Most of the time
- Always

Question 22: Which of the following stakeholders do you think could be/are the recipients of the feed-forward information? (Please select all that apply)

- Customers
- Requirements Engineers
- Software Architects
- Developers
- Managers
- Others (please specify)

Question 23: How often do you think (/do) the recipients obtain information as feed-forward from a stakeholder only after they requested for it?

- Never
- Rarely
- Sometimes
- Most of the time
- Always

Appendix C: Survey Data

Raw data of results from the survey

Level of Feed-forward practice	Percentage
Never	12.50%
Rarely	9.38%
Sometimes	53.13%
Most of the time	9.38%
Always	3.13%
Not sure	12.50%

Reasons behind absence of feed-forward in organization	number of respondents
Organizational Structure doesn't support	3
Never Heard of it before	2
Not Considered Important	2

Usefulness of feed-forward	Number of respondents	Number of respondents ("never"+"rarely")
I am not sure	8	4
Yes	16	3
No	1	0

Type of feed-forward Information	Number of respondents	Number of respondents ("never"+"rarely")
Prospective requirements	13	2
Requirements	19	6
Rationale, priority, source and assumptions behind requirements	11	6
Architectural artefacts	11	4
Other artefacts (Please specify)	1	
Architectural relevance of requirements	4	1
Cost, effort and change related information	12	3
Process related information	12	4
Domain related information	12	1
Constraints	10	3
Information about artefact decomposition and refinement	1	0
Others (Please specify)	0	0

Aspects influenced by Feed-forward Information	Number of respondents	Number of respondents ("never"+"rarely")
Cost	15	2
Quality	17	6
Time	20	4
Requirement Engineer and Software Architect	11	6
Morale	2	2
Designers, Coders, Testers, Integrators	7	1
Management and Quality Assurance	15	6
Customers	9	4
Process Improvement Agents	6	3
Process	2	1
Resources	8	3
Others	0	0

Sources of Feed-forward Information	Number of respondents	Number of respondents ("never"+"rarely")
Customers	12	4
Requirement Engineers	17	6
Software Architects	15	3
Developers	14	1
Managers	12	3
Others	1	1

Recipients of Feed-forward Information	Number of respondents	Number of respondents ("never"+"rarely")
Customers	6	2
Requirement Engineers	18	4
Software Architects	17	5
Developers	16	4
Managers	14	3
Others	2	1

How often information is fed-forward to appropriate recipients	Number of respondents	Number of respondents ("never"+"rarely")
Never	0	0
Rarely	5	2
Sometimes	12	4
Most of the time	6	2
Always	1	0

Recipients obtain information without having to request for it	Number of respondents	Number of respondents
Never	3	2
Rarely	7	0
Sometimes	20	5
Most of the time	2	0
Always	0	0

Appendix D : Case study questionnaire

Case study questionnaire

1. Did you receive any information as feed-forward from any of your group member/sub group/customer in today's session?
   - Yes
   - No

   If yes, Please answer the following questions.

   1.1. Who was the source of that information?
   - Name:
   - Role:

   1.2. What type of information was it?
   - actual one or more requirements
   - information tightly related to the requirement(s), such as assumption, rationale, relationship between requirements .
   - information about requirement decomposition, refinement, etc.
   - information about requirement characteristics (such as cost of implementation, implementation effort, resources needed, prioritization, change-related info, etc.)
   - information about software architecture
   - Other (please specify)

   1.3. Were you able to use that information in your architectural design?
   - Yes
   - No

If No,

1.3.1. Why could you not use that information?

If Yes, Please answer the following questions.

1.3.2. What was affected by the information that was fed-forward to you?
Example:

- A software artefact (e.g., an architectural artefact such as : (Architectural Driver, Element, Interface, Pattern, Quality Scenario, Decisions, Rationale, implications, Relationships, Tactics, Use Cases, etc., or others-state which))
- A process activity (state which)
- A decision of some sort (state which)

1.3.3. Why do you think it was helpful to receive that fed-forward information?

1.3.4. Which aspect (cost, quality, time, people, morale, etc.) of your architecture was influenced by that fed-forward information? Why do you think that particular aspect was influenced?

1.3.5. How much effort (in person-hours) was reduced (if any) by receiving that fed-forward information? What is your rationale behind this estimation?

1.4. Did you have to request for the information?

- Yes
- No

2. Did you feed-forward any information to any of your group member/sub group/customer in today's session?

- Yes

- No

If yes, Please answer the following questions.

2.1. Whom did you feed-forward that information to?

- Name:
- Role:

2.2. What type of information was it?

- actual one or more requirements
- information tightly related to the requirement(s), such as assumption, rationale, relationship between requirements .
- information about requirement decomposition, refinement, etc.
- information about requirement characteristics (such as cost of implementation, implementation effort, resources needed, prioritisation, change-related info, etc.)
- information about the architecture (please specify the type of the information)
- Other (please specify)

2.3. Did anyone have to request you for the information?

- Yes
- No