SOFTWARE ENGINEERING PROCESS IMPROVEMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BÜLENT SEZER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

APRIL 2007

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Semih BİLGEN
Supervisor

Examining Committee Members

Prof. Dr. Uğur HALICI	(METU, EEE)	_____
Prof. Dr. Semih BİLGEN	(METU, EEE)	_____
Assoc. Prof. Dr. Onur DEMİRÖRS	(METU, IS)	_____
Asst. Prof. Dr. Cüneyt BAZLAMAÇCI	(METU, EEE)	_____
Sinan ÇEÇEN (M.Sc)	(ASELSAN)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have cited and referenced all material and results that are not original to this work.

Name, Last Name: Bülent SEZER

Signature :

ABSTRACT

SOFTWARE ENGINEERING PROCESS IMPROVEMENT

SEZER, Bülent

M.S., Electrical and Electronics Engineering Department

Supervisor: Prof. Dr. Semih BİLGEN

April 2007, 113 pages

This thesis presents a software engineering process improvement study. The literature on software process improvement is reviewed. Then the current design verification process at one of the Software Engineering Departments of the X Company, Ankara, Türkiye (SED) is analyzed. Static software development process metrics have been calculated for the SED based on a recently proposed approach. Some improvement suggestions have been made based on the metric values calculated according to the proposals of that study. Besides, the author's improvement suggestions have been discussed with the senior staff at the department and then final version of the improvements has been gathered. Then, a discussion has been made comparing these two approaches. Finally, a new software design verification process model has been proposed. Some of the suggestions have already been applied and preliminary results have been obtained.

Keywords: Software Process Improvement, Metrics, Design Verification

ÖZ

YAZILIM MÜHENDİSLİĞİ SÜRECİ İYİLEŞTİRME

SEZER, Bülent

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Semih BİLGEN

Nisan 2007, 113 sayfa

Bu tez bir yazılım mühendisliği süreci iyileştirme çalışması sunmaktadır. Yazılım süreci iyileştirme üzerine literatür incelenmiştir. Daha sonra Ankara'da faaliyet gösteren X Firmasının Yazılım Mühendisliği Bölümlerinden birinde uygulanan tasarım doğrulama süreci analiz edilmiştir. Bu bölüm için yakın zamanda önerilmiş olan bir yönteme göre statik yazılım geliştirme süreci metrikleri hesaplanmıştır. Bu çalışmanın neticesinde elde edilen metrik değerlerine dayanarak bazı iyileştirme önerileri yapılmıştır. Bununla beraber, yazarın ortaya koyduğu iyileştirme önerileri bölümde çalışan üst düzey personelle tartışılmış ve daha sonra iyileştirme önerilerinin son hali bir araya getirilmiştir. Bu iki iyileştirme yöntemi üzerine bir tartışma yapılmıştır. Son olarak, yeni bir tasarım doğrulama süreci modeli önerilmiştir. Önerilerin bazıları uygulanmaya konmuş ve ilk sonuçlar alınmaya başlanmıştır.

Anahtar kelimeler: Yazılım Süreci İyileştirme, Metrik, Tasarım Doğrulama

ACKNOWLEDGMENTS

I would specially like to thank to my supervisor Prof. Dr. Semih BİLGEN for his valuable guidance, advice, encouragements throughout the thesis. It was a real pleasure and chance to work with him.

I would also like to thank to the senior staff of the SED at the X Company for their cooperation and valuable contributions to this study.

I would also like to thank to Assoc. Prof. Dr. Onur DEMİRÖRS for his valuable comments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

BDD	Block Design Document
BID	Block Interface Document
CASE	Computer Aided Software Engineering
CM	Configuration Management
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
DT	Design Template
IEC	International Electro technical Commission
IS	Information Systems
ISO	International Organizational for Standardization
RAD	Rapid Application Development
RTM	Requirement Traceability Matrix
SAD	Software Architecture Design
SADR	Software Architecture Design Review
SARGD	Software Architecture Review Guideline
SDD	Software Detailed Design
SDDD	Software Detailed Design Documents
SDDR	Software Detailed Design Review
SDDRG	Software Detailed Design Review Guideline
SDN	Software Development plaN
SDP	Software Development Process
SED	Software Engineering Department in X Company
SEI	Software Engineering Institute
SOP	Step On Paper
SPI	Software Process Improvement
SRNo	System Requirement Number
SRS	Software Requirements Specification

SwRNo	Software Requirement Number
UDD	Unit Design Document
UID	Unit Interface Document
VM	Version Management
XP	eXtreme Programming

CHAPTER 1

INTRODUCTION

Software has become an integral part of our lives with the penetration of electronic equipments into our everyday lives. Whether we are aware of it or not, the electronic equipments we use have software controlling their functionality.

Ed Yourdon in his foreword for the "Managing Software Requirements" book [16] describes software systems as being, by their nature, intangible, abstract, complex, and – in theory at least – "soft" and infinitely changeable. All these indicate that developing software is a complex activity.

This complexity gives rise to many problems to cope with. First of all, software development is an expensive business. Both the human and the technology resources needed require a lot of investment. Second, most of the software projects can not be completed on time and within budget. Third, a considerable part of the software projects are stopped before they are completed. Last but not least, many projects are completed with defects to be fixed after delivery which results in customer dissatisfaction and poor quality products. All these are the major problems that the software industry has been trying to solve for many years.

The software business's significant impact on today's economy generates considerable interest in making software development more cost effective and producing higher quality software [1]. An InformationWeek survey conducted in 2003 found that 62 percent of their respondents feel that the software industry has trouble producing good quality software [3]. Losses due to inefficient development practices lead to inadequate quality that cost the US industry approximately $60 billion per year [4].

To manage the intrinsic complexity of software development, to solve the problem we need some methods. We have to bring some discipline to the way we develop software.

The methods, the relationships between the methods, and the use of these in development of software can be called “software process”. Pressman simply defines “software process” as: “… a framework for the tasks that are required to build high-quality software” [17].

Any software development process usually has Requirements Analysis, Design, Implementation, Test and Maintenance parts. The well-known Waterfall software development life-cycle model is given below in Figure 1–1:

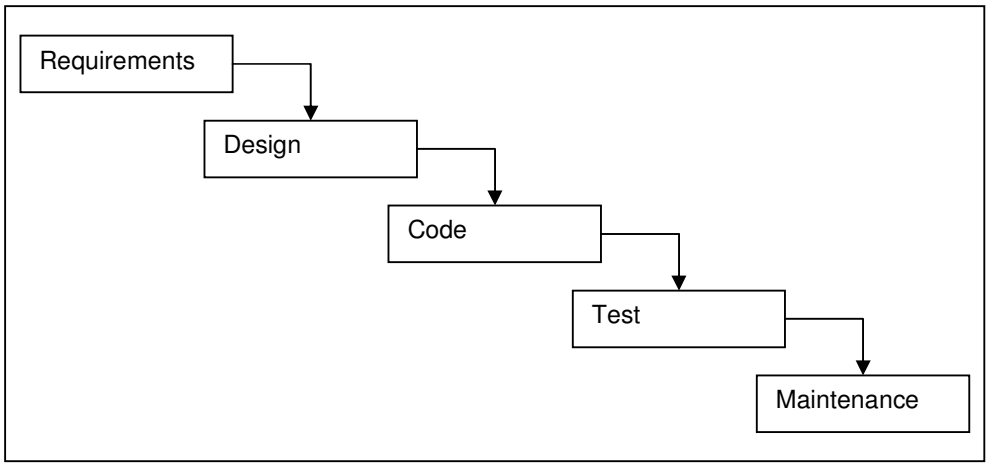

Figure 1-1 Waterfall software development life-cycle model

Software design process is usually further divided in two phases: Software Architecture Design (SAD), and Software Detailed Design (SDD). Architecture design is usually referred as high-level design as well. It is the activity of deciding on the software components, their deployment, and allocation of functions to these components. In SDD, these components,

their functionality and interfaces are further detailed to fulfill the requirements [24]

There are integral processes such as verification, Configuration Management (CM), etc., along with these main activities as well. These integral processes provide software engineering process support that helps to ensure the completeness and quality of the software development activities. A productive process uses a constructive approach that satisfies the completion criteria of the integral processes (e.g. testing, verification, etc.) during the software development activities [25]. The software development life-cycle model enhanced with these integral processes is given below in Figure 1–2:

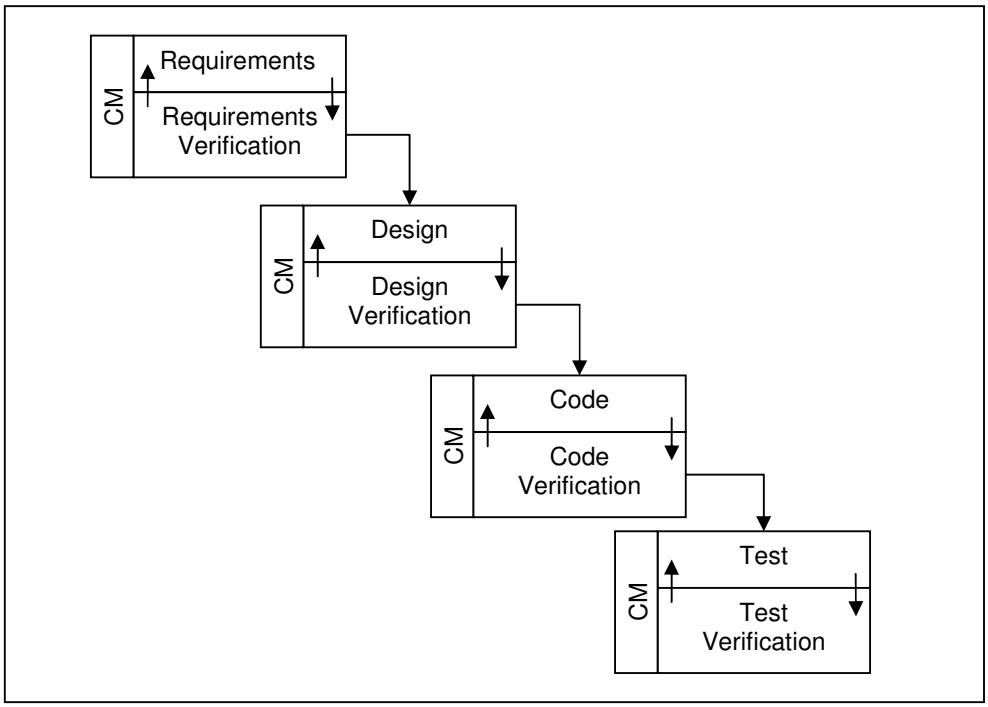

Figure 1-2 Software development life-cycle with integral processes

The verification process, which includes planning, requirements definition, and compliance activities, begins early and continues throughout the project life cycle [26].

Software Design Verification is the process of confirming that deliverable software is in compliance with requirements. It is a crucial activity that ensures the software design fulfills the software requirements established in the previous stage, and that software is ready to proceed to next coding phase with a high degree of confidence [26].

There has been a lot of research carried out to improve software development practices, namely software processes of organizations. The Capability Maturity Model (CMM) [7], the ISO/IEC IS 15504 [9], and the Bootstrap [10] are the major Software Process Improvement (SPI) models that are widely accepted around the world.

SPI is the act of creating a new and improved software process in order to obtain a benefit [15]. Usually, the major goals are to increase efficiency, decrease costs, shorten time to market, and improve quality and reliability.

SPI has been proven to increase product and service quality as organizations apply it to achieve their business objectives. Software process assessment and improvement is widely acknowledged as one of the most important means for achieving competitive and effective software industry [8].

ISO/IEC 9126 Software Product Quality Model [11] describes a software product assessment model for developing or selecting high quality software products. It presents a comprehensive specification and evaluation framework for ensuring software product quality. The software product is evaluated for every relevant quality characteristics in the model, by using validated and widely accepted metrics.

There is a close relation between the software process and the software product. For instance, process and software product have similar logical structures. While a process is defined with inputs, a set of activities

and outputs, a software product is defined with input parameters, a set of functions and output parameters [14]. This relation makes it possible to assess a software process using the ISO/IEC 9126 model and then further improvement study can be possible based on this assessment.

Selçuk Güceğlioğlu in his recent study has formed a new SPI model based on ISO/IEC 9126. The objectives of Güceğlioğlu's study were [14]:

- Providing complementary information about the process quality to the available time and cost related models.
- Providing the usage of process definitions for measuring the process quality.
- Providing a model for measuring the effects of Information System (IS) on the organizational impact dimension.
- Investigating IS effects on organizational impact dimension in terms of process quality attributes.

In his study, Güceğlioğlu has redefined software quality characteristics in ISO/IEC 9126 according to the process specific attributes and identified new characteristics unique to the processes to extend the model. Based on these definitions he has specified process metrics [14].

The present thesis deals with a software process improvement study for the design and design verification processes of the Software Engineering Department at the X [*] Company (SED).

First, the current design verification process at the SED is analyzed. Then the current process is assessed using Güceğlioğlu's study [14]. The improvement suggestions are grouped in two sections: First, suggestions gathered from the staff. Second, suggestions based on the assessment of the current process according to Güceğlioğlu's study. A discussion is made on these two approaches. Finally, an improved design verification process is modeled based on these discussions and information gathered.

---

[*] The X Company did not give permission to reveal its identity. Therefore, the company will be called "X Company" throughout the document.

The thesis is organized as follows:

A brief review of the relevant literature is given in Chapter 2. In Chapter 3, first the current process model is presented. Next, applicable metrics according to Selçuk Güceğlioğlu's study [14] are calculated. Then, the improvement suggestions gathered from the staff and improvement suggestions derived according to the metric values are outlined. This chapter is completed with a discussion on these improvement suggestions. In Chapter 4, the improved process model is given. Finally, in Chapter 5, an evaluation is presented and the study is concluded.

CHAPTER 2

LITERATURE

It was 1989 when Victor R. Basili had made the following statement about software development: “We have begun to understand that software development is not an easy task. There is no simple set of rules and methods that work under all circumstances” [2].

During the past 18 years, and even since much earlier, there have been a lot of studies and research trying to close the gap between the desired software quality and the actual one. Effort has been spent for having better software development models either for addressing the existing or emerging needs of the software industry.

There have been many software process models introduced already. There are also new ones appearing in the scene. Waterfall, prototyping, incremental, Rapid Application Development (RAD), spiral, eXtreme Programming (XP) and TSP are some of the major process models to mention.

All of these models aim to help mastering the problem of software development. They are introduced because it is early recognized that software development is not an easy task. But, why do we have so many process models? It is understandable that as the software technology and the software industry evolves, the need for new process models arise, or modification of the current models is needed. But, introducing new ones and refining the available models have not helped much to solve the major problems such as high costs, budget and schedule overruns or simple failures, poor quality software delivery, etc. in the software industry.

Margaret Davis in her article [18] has pointed towards this briefly as “… the software community redefines the problem by shifting its focus from product issues to process issues... While the natural tendency of a

pendulum is to come to rest at a point midway between two extremes, the software community's focus constantly shifts…"

We should not interpret the disappointing results as the failure of the process models chosen. In fact, choosing a process model or creating one from scratch, is not easy. It is not like going to a dress store and choosing the one that fits you. Moreover, you can never be sure if you have the right process without applying it. To make a decision about whether things are better after applying the process you have to be able to measure and assess your process. To do this you need assessment of both the old and new processes.

Assessment, and its pre-requisite, measurement, may be the weakest point of any software organization. De Marco and Brady [22] have made the following stunning statement about measurement:

> *Only in software do people cling to the illusion that it's OK to come up with estimates of the future, even though you've never measured anything in the past.*

Michael E. Fagan's comment, which goes back to 1976 [23], for "successful management of any process" describes very well why measurement is important:

> *Successful management of any process requires planning, measurement, and control. … there must be some means of measuring completeness of the product at any point of its development by inspections or testing. And finally, the measured data must be used for controlling the process.*

Obviously, measuring or assessing how a process works is one thing, but improving it is another. Being aware that applying software processes have not solved all the problems, the solution has been introduced: Software Process Improvement.

Software process improvement has become the primary approach to improve software quality and reliability, employee and customer satisfaction, and return on investment [19].

Based on the software CMM [7], ISO/IEC-9126 [11] and similar norms, SPI aims to build an infrastructure and culture that support effective methods, practices, and procedures and integrate into the ongoing way of doing business. This goal has proven difficult to reach, however, because SPI involves organizational change on a scale and of a complexity that requires commitment, resources, and skills at all organizational levels, and it carries a large risk of failure [6].

Over the past decade or so, organizations have spent vast resources on SPI [6]. Although there have been a lot of studies aiming at better software, the results have not changed dramatically since those studies and research has begun.

Being ready to change may be the hardest step to take. Because, from developers to managers, SPI efforts need commitment at all levels. Many managers lack experience in projects that transform organizations. To implement software process improvement, they must know the context, the organizational elements, and the tactics that facilitate successful change.

SPI can be used to improve any part of a Software Development Process (SDP). For example, if it is used for project management process, the new process may result in faster cycle times, shorter time to market, higher customer satisfaction, accurate time and budget accounting, and better cost and schedule performance [15].

Usually SPI studies start with the assessment of the current process. As a result of this assessment the points that need improvement are figured out. Sometimes, it may be decided to change the whole process.

Since most of the companies do not have accurate measurements, successful assessment is not possible all the time. Some of them even do not have a measurement system. In 1994, 75 percent of the industry was not collecting metrics, hence could not be measured [21].

The ones who have a measurement system generally do not make the measurements as they should. A recent Software Engineering Institute report suggests that many SPI initiatives have difficulty managing the

changes required after the initial assessment. By August 2004, 2,561 organizations had reported assessments conducted between 1987 and June 2004. Only 630 of these organizations (approximately 25 percent) had been reassessed [19].

Apart from the difficulties associated with SPI itself, the process or organization oriented difficulties mentioned above are on the way as well.

Although the literature acknowledges that SPI implementations face various problems, some cases report success, detailing dramatic improvements. There are observations around *Cocomo II* indicating that each increased CMM level (aside from other effects) means a productivity gain of 4 to 11 percent [20].

Two impressive success stories among others are from the Systems, Engineering, and Analysis Support (SEAS) Center of the Computer Sciences Corporation (CSC), US, [23], and a Danish company which has not been revealed [19]. SEAS from US, by initiating an SPI program in 1995, had become the sixth organization in the world to attain the Software Engineering Institute's CMM Level 5, back in 1998. Although SEAS has been a company with a long experience with processes and carrying out SPI programs, it is still interesting that they have achieved this in just three years. Just a year after SEAS had kicked off its SPI program, it was predicted that at the average rate of increase it would take 10 years to advance from CMM Level 2 to Level 3. The Danish company, which was a medium-sized software house (140 employees), in 1997, had aimed for reaching CMM Level 3 in less than three years. Although the organization had not reached its goal in three years, it was formally assessed at CMM Level 3 in 2002 and at CMMI Level 4 in 2004.

Analyzing carefully the results obtained from the industry, SPI seems to be the ultimate approach to achieve better software development practices. However, there is no magic SPI program that could help every organization to reach its improvement goals. Furthermore, there are not enough examples to sample from or, the worse, since every organization has its own

characteristics – staff, culture and similar notions –, it is not possible to be sure about whether a successful SPI program can fit you or not.

Assessment of the current process is very important before making an SPI attempt. The assessment of the current process will form a basis and guide the SPI study.

The ISO/IEC 9126 standard is one of the available models that can be used to assess a software process or product. Using the ISO/IEC 9126 model it is possible to evaluate the product or process from quality point of view.

Software product quality can be evaluated by measuring internal attributes (typically static measures of intermediate products), or by measuring external attributes (typically by measuring the behavior of the code when executed), or by measuring quality in use attributes. The objective is for the product to have the required effect in a particular context of use.

ISO/IEC 9126 describes a software product evaluation model for developing or selecting high quality software products. It presents a comprehensive specification and evaluation framework for ensuring software product quality. The software product is evaluated for every relevant quality characteristics in the model, by using validated and widely accepted metrics. ISO/IEC 9126 categorizes software quality attributes into six characteristics as functionality, reliability, usability, efficiency, maintainability and portability, which are further sub-divided into sub-characteristics. The sub-characteristics can be measured by internal or external metrics [14].

Process quality (the quality of any of the lifecycle processes) contributes to improving product quality, and product quality contributes to improving quality in use. Therefore, assessing and improving a process is a means to improve product quality, and evaluating and improving product quality is one means of improving quality in use. Similarly, evaluating quality in use can provide feedback to improve a product, and evaluating a product can provide feedback to improve a process [27].

An SPI approach for evaluating metrics based on static descriptions of software development processes, rather than dynamic observation and evaluation of the processes as applied, has recently been proposed by Güceğlioğlu [14]. Using the analogy of software product evaluation via product metrics such as cyclomatic complexity, interoperability, testability etc., Güceğlioğlu suggests that an organization can benefit from product based models and also process quality based measurements for selecting the most suitable alternative. The model proposed can also be used by itself in the process improvement studies. By means of the model, organizations can measure impacts of the process improvement studies on their process quality [14].

The model is provided with a suggested set of process quality metrics. The users of this model are encouraged to modify the current metrics that are defined, and/or also to develop new metrics not listed if needed. The metrics provide the users with the ability to measure the quality of the activities and thereby predict the quality of the process. This allows the users to detect quality issues and take corrective actions during the early stages of the development. The users can measure the extent to which the process meets quality considerations [14].

Güceğlioğlu's approach brings predictability to some extent and mention about adopting ISO 9126-3 to an organization's needs which the author also agrees. But, as discussed later in detail in section 3.5, the author thinks, both the Güceğlioğlu's study and the ISO 9126 standard does not reflect the effectiveness of the process. These models consider all activities as same without weighing the effect/contribution of activities to the whole process.

In a similar study that has applied Güceğlioğlu's method in a specific process in a software development company, H. Seçkin [28] has found that SPI work based on that method is viable, but a more comprehensive improvement effort is needed to bring tangible benefits.

To develop high quality software, to manage software development, every organization should first formally define its software development process. Then, it should tailor the process to itself, following the industry standards, rather than by just picking a process from the shelf and trying to apply it as it is. Next, an organization must assess and measure its process.

> *Because poor-quality work is not predictable, quality is a prerequisite to predictability.*
>
> *Watts S. Humphrey [24]*

Without assessment and measurement, a process is like a car without tires. Tires are not the most expensive and important part of a car. But you can not get to the place you want without them. So, after making the assessment and analyzing the measurements, you can start improving your process. Again, you should take advantage of the existing guidelines like CMM or CMMI for your SPI program, but sticking to them as they are, without adapting to your organization, may not produce the expected outcome.

CHAPTER 3

CURRENT PROCESS MODELS AND IMPROVEMENT SUGGESTIONS

In this chapter first the current on-paper and As-Is SAD, SADR, SDD, and SDDR are given respectively. Then the As-Is process is assessed according to Güceğlioğlu's study [14].

There are several reasons why Güceğlioğlu's study has been chosen as the basis for the assessment and improvement studies in this thesis. First of all, it was not possible to apply CMM, or ISO/IEC 15504 based SPI program within the duration that this thesis has to be completed. Also, CMM, ISO/IEC 15504 like models require more than one person to be dedicated to the improvement studies. Last, but not least, in an IS development project, frequently, processes of an organization are analyzed and a system is designed with new process definitions. Most of the studies in the IS literature employ time and cost based models and attributes such as productivity growth, return on investment and market share for measuring effects of IS projects on the organizations. These models can provide the organizations with crucial information about IS effects, but, naturally, they can only be measured during or after the processes are executed. The processes should be modified according to these measurements and re-executed to measure the effects of new arrangements. This kind of iterations requires much effort and cost [14]. However, with the model proposed by Güceğlioğlu, which is based on ISO/IEC 9126, it is possible to predict the quality of the process before it is executed. As a result, Güceğlioğlu's SPI model has been used for the assessment and improvement purposes in this thesis.

3.1 Current Design and Review Processes at SED

In this section the on-paper and as-is Architecture Design (SAD), Detailed Design (SDD), Architecture Design Review (SADR) and Detailed Design Review (SDDR) processes applied in SED are presented.

3.1.1 On - paper SAD Process

3.1.1.1 Inputs

1. Software Requirements Specification (SRS).
2. Hardware Architecture Plan.

3.1.1.2 To-Do

1. Requirements are analyzed.
2. The software blocks required for the equipment/system, their functions and place on the hardware are determined.
3. Work load and timing information is updated if necessary.
4. Tasks are updated if necessary.
5. Units and their functions are determined.
6. The block and unit interfaces are determined at high level (only info flow is presented at this stage).

3.1.1.3 Roles

1. Software Project Manager
2. Software Engineers
3. Software Test Engineers

3.1.1.4 Output

Software Architecture Design Document (SADD).

3.1.2 On - paper SAD Review Process

3.1.2.1 Inputs

1. SRS.
2. SADD.

3.1.2.2 To-Do

Software architecture design is reviewed and updated if necessary.

3.1.2.3 Roles

Same as SAD (section 3.1.1.3).

3.1.2.4 Output

1. SAD review report.
2. Updated Software Architecture Design Document.

3.1.3 On - paper SDD Process

3.1.3.1 Inputs

1. SRS.
2. SADD.

3.1.3.2 To-Do

1. Block interfaces are defined.
2. Communication mechanism between blocks is determined (hardware and software).
3. Units are detailed. Functions are determined. If necessary, a unit can involve other unit(s).
4. Unit interfaces are defined.
5. Communication mechanism between units is determined (ie. Message, function call, semaphore).
6. For every process that will be carried out by the software, sequence diagrams are prepared per block and/or unit basis.
7. Data structures are formed.

3.1.3.3 Roles

1. Software Project Manager.
2. Software Engineers

3.1.3.4 Outputs

1. Block Design Document (BDD) which describes a block in detail (function of the block, inputs, outputs, etc.) for each block.
2. Block Interface Document is prepared and referenced from the relevant BDDs.
3. If necessary, Unit Design Document describing a unit in detail is prepared (function of a unit, inputs, outputs, etc.). This document can be placed in BDD.
4. Unit Detailed Interface Document. This document can be placed in the BDD.
5. Sequence diagrams per block and/or unit basis.
6. The header files holding the values and data structures that will be used at the interfaces.

3.1.4 On - paper SDD Review Process

To review the SDD, the “Outputs” section of the SDD is not waited to completion. Review process is carried out along with the design process.

3.1.4.1 Input

The outputs of the SDD section are used as inputs to this section.

3.1.4.2 To-Do

1. It is checked that all the requirements are covered in the design.
2. The block interface documents are reviewed by the participation of the relevant designers. Interface documents are updated if necessary.

3.1.4.3 Roles

Same as SDD.

3.1.4.4 Outputs

1. SDD review report.
2. If there are any changes, updated versions of the documents in the “SDD Outputs” section.

3.1.5 As-Is SAD Process

As-Is processes are presented together with on-paper processes to see differences at a glance. First SAD is presented. Then the SDD follows. The as-is architecture design process is given in Table 3-1:

Table 3-1 As-Is SAD Process

Step	On – paper	As – Is
Inputs	1. SRS.	Usually, SRS is missing and Product Description Document is used instead. This point is considered as the first weakness to be improved in the process.
	2. Hardware Architecture Plan.	Hardware Architecture Plan is usually present.
To Do	1. Requirements are analyzed.	Requirements are analyzed by all relevant departments. Sometimes internal meetings are held to have a better understanding of the requirements in every department prior to a general meeting. But usually, because of lack of time, meetings are arranged by the participation of every department. There are no formal records keeping track of these meetings. The design decisions are discussed. The decisions held in such meetings are put in paper form (word doc usually) and sent to all relevant participants via e-mail for review after the meeting. Also, technology needs are discussed and if necessary, CASE tools for blocks and units are determined.
	2. The software blocks required for the equipment/system, their functions and place on the hardware are determined.	Same as on-paper.

Table 3-1 (cont'd)

Step	On – paper	As – Is
To Do	3. Work load and timing information is updated if necessary.	Same as on-paper.
	4. Tasks are updated if necessary.	Same as on-paper.
	5. Units and their functions are determined.	Not available at this stage. Usually done at the Detailed Design Phase.
	6. The block and unit interfaces are determined at high level (only information flow is presented at this moment).	Only the block interfaces are determined at this stage.
Roles	1. Software Project Manager.	Usually the department manager is the project manager.
	2. Software Engineers.	Same as on-paper. But carry out other roles as well: test engineer, field engineer. Even sometimes system engineer.
	3. Software Test Engineers.	Not available. The software engineers themselves carry this role.
Output	Software Architecture Design Document.	Same as on-paper. But the changes happening in the life cycle of the projects are not reflected to this document instantly.

3.1.6 As-Is SDD Process

The as-is detailed design process is given in Table 3-2:

Table 3-2 As-Is SDD Process

Step	On – paper	As – Is
Inputs	1. SRS.	SRS is sometimes missing or Product Description Document is used instead.
	2. SADD.	Same as on-paper.
To Do	1. Block interfaces are defined.	Same as on-paper. But, the changes happening in the life cycle of the projects are not reflected to this document instantly.
	2. Communication mechanism between blocks is determined (hardware and software).	Same as on-paper.
	3. Units are detailed. Functions are determined. If necessary, a unit can involve other unit(s).	Usually left to the implementation stage.
	4. Unit interfaces are defined.	Usually left to the implementation stage.
	5. Communication mechanism between units is determined (i.e. Message, function call, semaphore)	Usually left to the implementation stage.
	6. For every process that will be carried out by the software, sequence diagrams are prepared per block and/or unit basis.	Partially done. If there is time, scenarios are prepared and all of them are tried to be covered. But usually all of the scenarios are not covered due to lack of time. Nothing done for some of the blocks/units at all. Also, these diagrams are not updated accordingly because of the changes happening in the life cycle of the projects.

Table 3-2 (cont'd)

Step	On – paper	As – Is
To – Do	7. Data structures are formed.	Usually left to the implementation stage.
Roles	1. Software Project Manager.	Usually the department manager is also the project manager.
	2. Software Engineers.	Same as on-paper. But carry out other roles as well: test engineer, field engineer.
Outputs	1. Block Design Document (BDD) which describes a block in detail (function of the block, inputs, outputs, etc.) for each block.	Usually not very detailed or missing.
	2. Block Interface Document is prepared and referenced from the relevant BDDs.	Sometimes missing, or not very comprehensive.
	3. If necessary, Unit Design Document describing a unit in detail is prepared (function of a unit, inputs, outputs, etc.). This document can be placed in BDD.	Usually missing even when necessary.
	4. Unit Detailed Interface Document. This document can be placed in the BDD.	Usually missing. Left to the implementation stage. At the implementation phase, usually header files are used as interface documents.
	5. Sequence diagrams per block and/or unit basis.	Partially present. But sometimes, the present ones are not updated accordingly. i.e. when an update/change is done in the following stages.
	6. The header files holding the values and data structures that will be used at the interfaces.	Usually left to the implementation stage.

3.1.7 As-Is SADR and SDDR Processes

The process model presented in this section has been constructed using the documented process description, even though incomplete, in SED, as well as the personal experience that the author has gained through active duty in SED for 9 years, and also through interviews with colleagues, supervisors and subordinate software developers. The presented process models have been reviewed by colleagues and supervisors and their agreement has been obtained on the fact that the model does reflect the situation correctly.

As-Is SADR Process and As-Is SDDR Process are outlined in Table 3-3 and Table 3-4 respectively:

Table 3-3 As-Is SADR Process

Step	On – paper	As – Is
Inputs	1. SRS.	SRS is sometimes missing or Product Description Document is used instead.
	2. Software Architecture Design Document.	Same as on-paper.
To – Do	Software architecture design is reviewed and updated if necessary.	If there is enough time SAD is reviewed properly. After SAD is formed every department goes over it either by internal formal meetings or by informal meetings. Points of concerns or unclear parts are noted. Then, a review meeting is arranged by the participation of all relevant departments and these points are discussed. If necessary the SAD is updated. Lack of human resources is a key issue at this point. Since there is not enough system engineers or, same people carry out different roles at the same time (including system engineering role as well), defects can be easily missed even when reviews are held.
Roles	1. Software Project Manager.	Usually the department manager is also the project manager.
	2. Software Engineers.	Same as on-paper. But carry out other roles as well: test engineer, field engineer. Even sometimes system engineer.
	3. Software Test Engineers.	Not available. The software engineers themselves carry this role.
Outputs	1. SAD review report.	This report is missing usually due to lack of time or because it is not assigned as a duty to anyone.
	2. Updated Software Architecture Design Document.	Same as on-paper.

Table 3-4 As-Is SDDR Process

Step	On – paper	As – Is
Input	The outputs of the SDD section are used as inputs to this section.	Usually review is missing or not done properly.
To Do	1. It is checked that all the requirements are covered in the design.	Not checked formally. Although software is designed to cover all the requirements, there is no formal design verification method followed to check this. But, there are formal/informal meetings held where such points are discussed. Sometimes, by the help of CASE tools (SDT, OPNET, etc.) simulation is used for this purpose. If there is enough time, scenarios are prepared to cover all requirements and possibilities. By the help of simulation the design is checked against these scenarios. When there is not enough time, which is often the case, simulation is carried out only for the basic scenarios. Lack of human resources is an important issue yielding this situation. Software engineers usually carry out different roles at the same time such as System Engineer, Test Engineer, and Field Engineer. Maintaining old projects is another issue. Since there is not enough human resources same people always take place in new projects and these people have to maintain old projects as well. This last part can be considered as bad project planning.
	2. The block interface documents are reviewed by the participation of the relevant designers. Interface documents are updated if necessary.	If there is time and if they exist, these documents are reviewed by organizing meetings. But this is often not the case. Usually, such documents are prepared and not reviewed properly due to lack of time. When a problem or uncovered situation occurs in the implementation phase, document is updated.

Table 3-4 (cont'd)

Step	On – paper	As – Is
Roles	1. Software Project Manager.	Usually the department manager is also the project manager.
	2. Software Engineers.	Same as on-paper. But carry out other roles as well: test engineer, field engineer.
Outputs	1. SDD review report.	Not present.
	2. If there are any changes, updated versions of the documents in the "SDD Outputs" section.	Same as on-paper (If review is done).

3.1.8 As-Is SADR Process Model

The as-is SADR process is modeled as given in Figure 3-1:

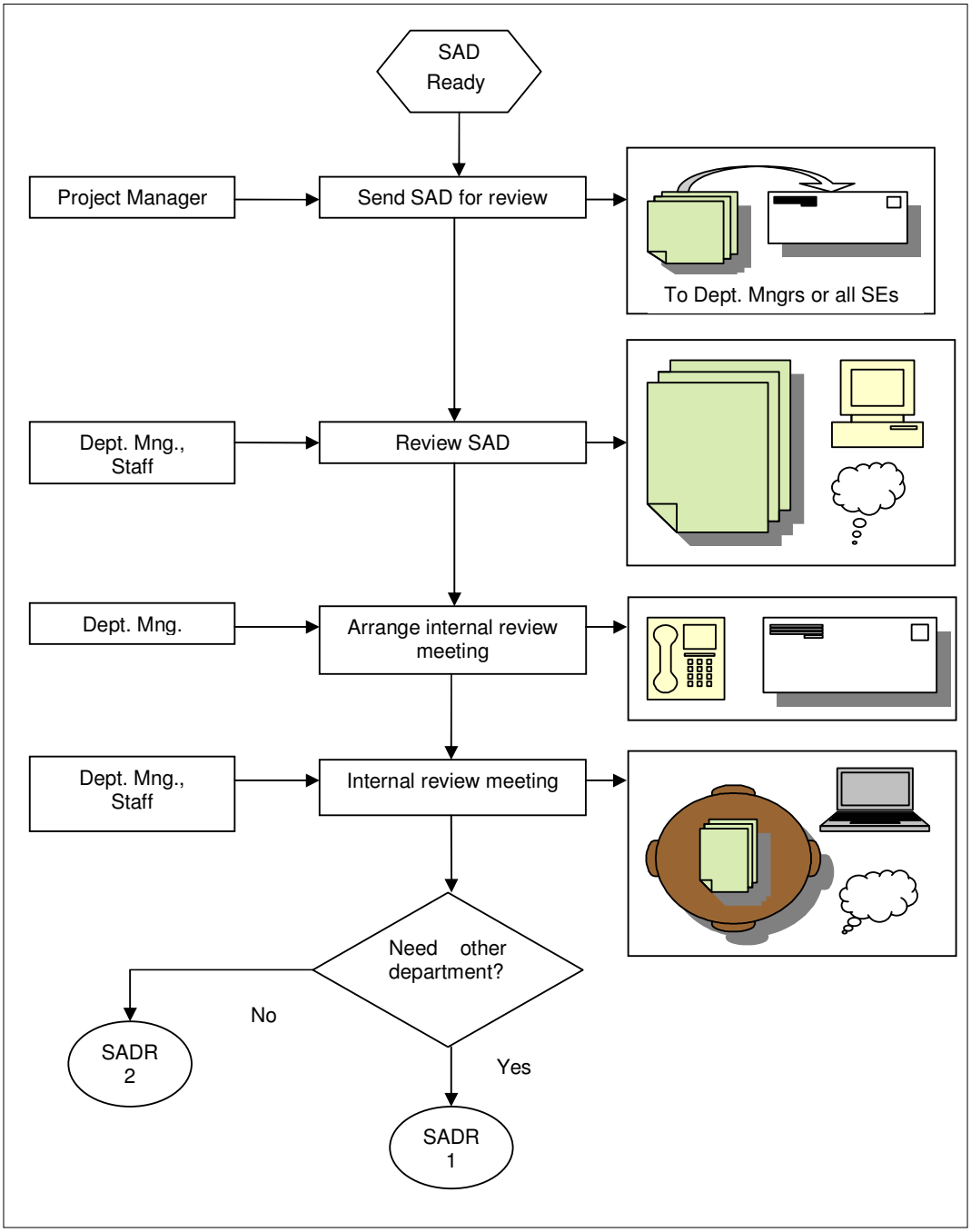

Figure 3-1 As-Is SADR Process Model

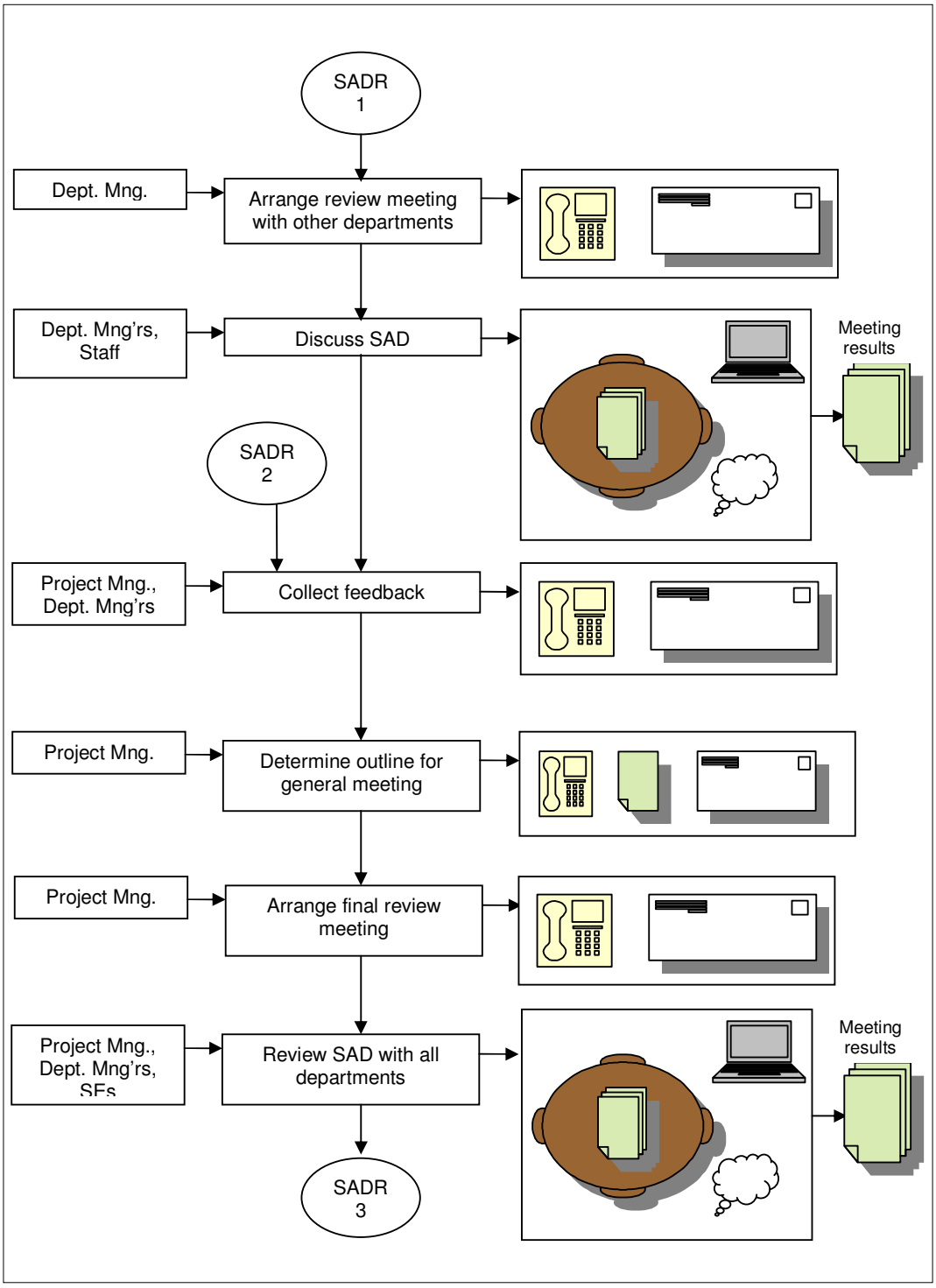

Figure 3-1 (cont'd)

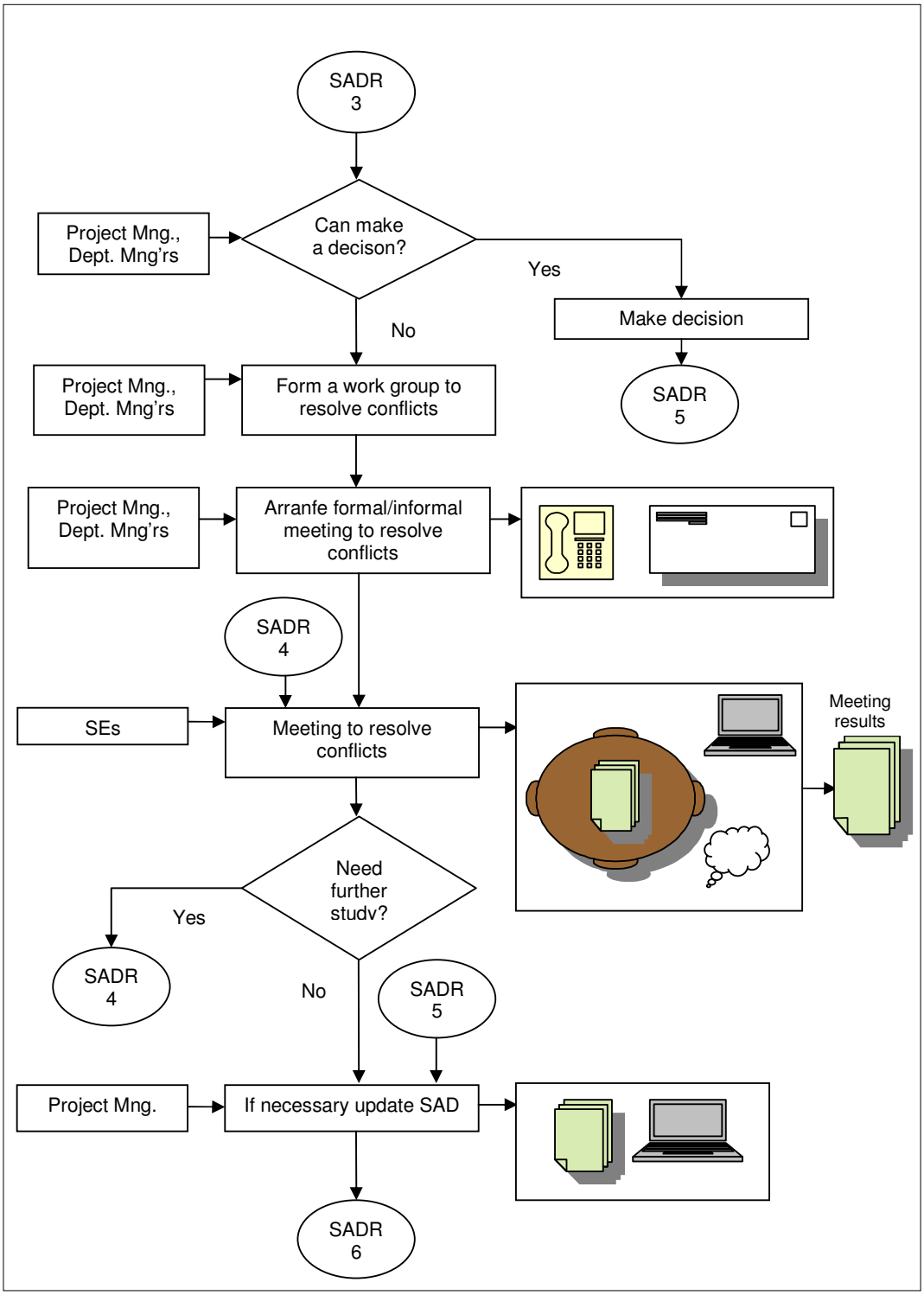

Figure 3-1 (cont'd)

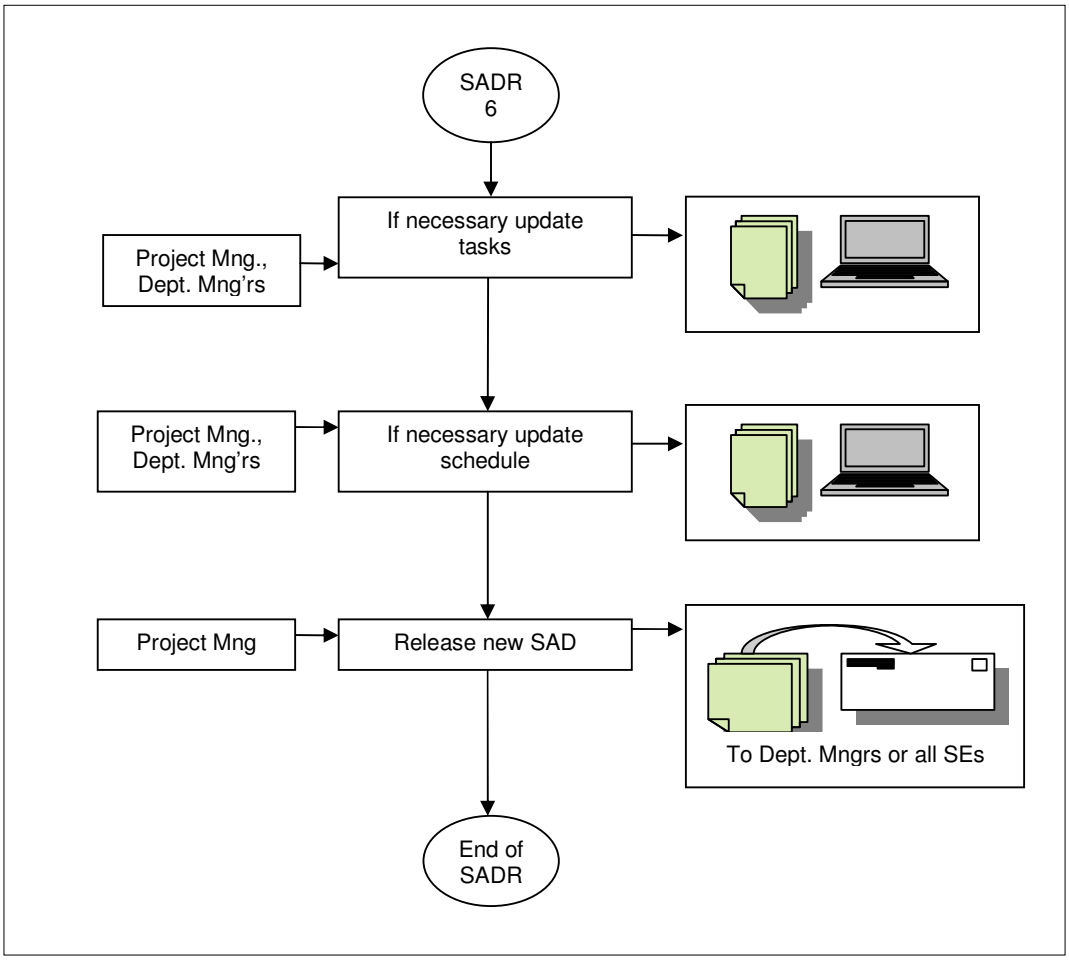

Figure 3-1 (cont'd)

3.1.9 As-Is SDDR Process Model

The as-is SDDR process is modeled as given in Figure 3-2:

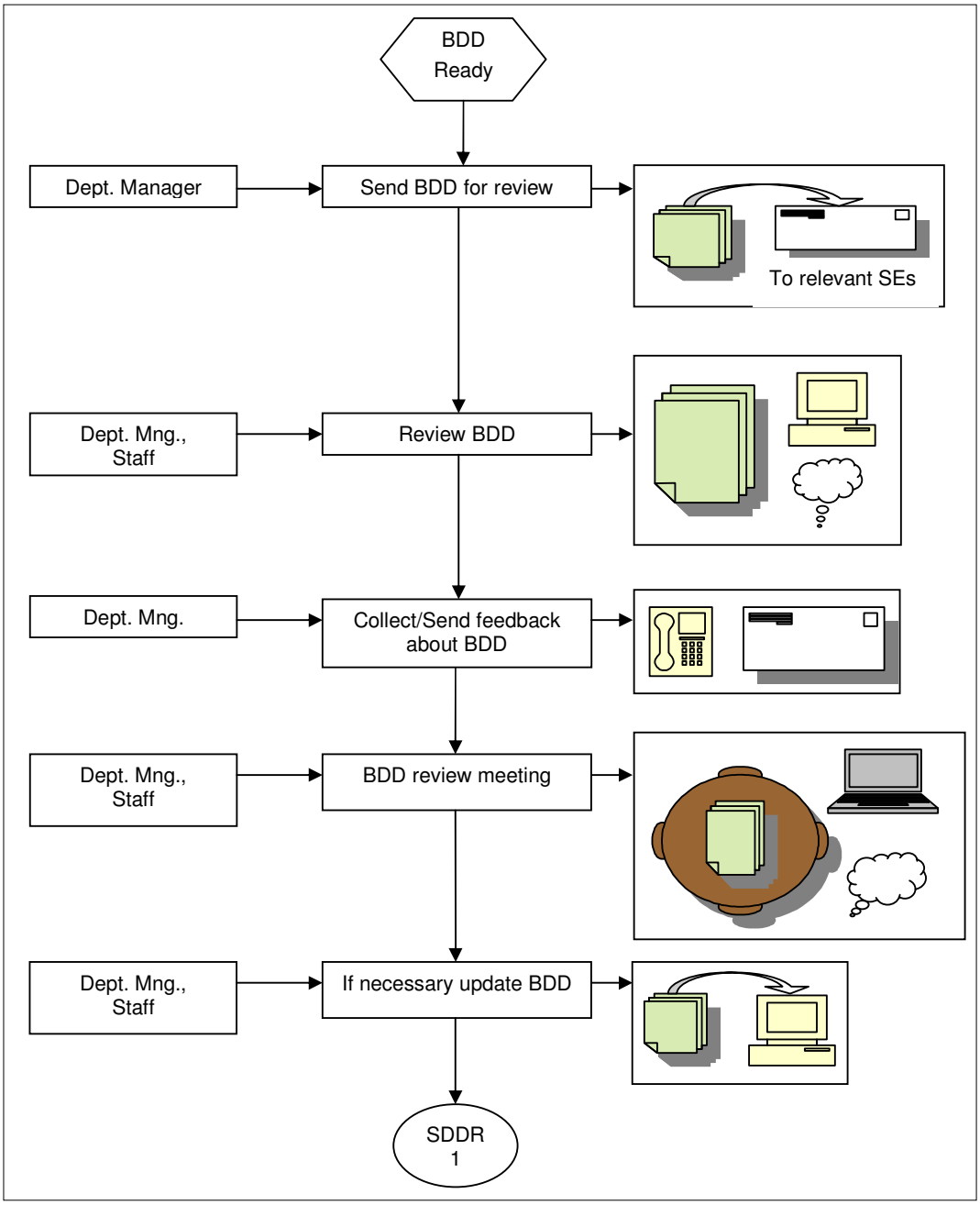

Figure 3-2 As-Is SDDR Process Model

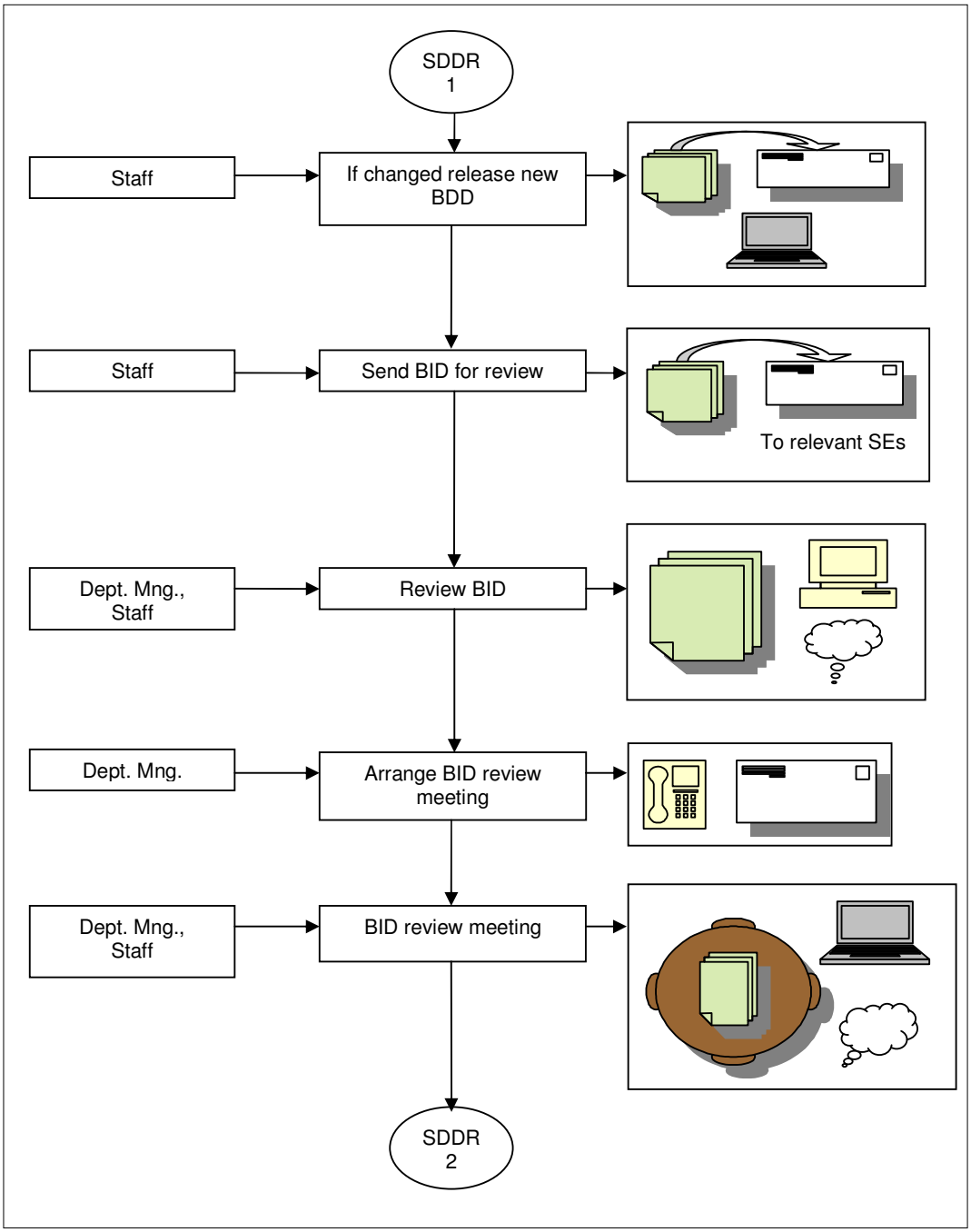

Figure 3-2 (cont'd)

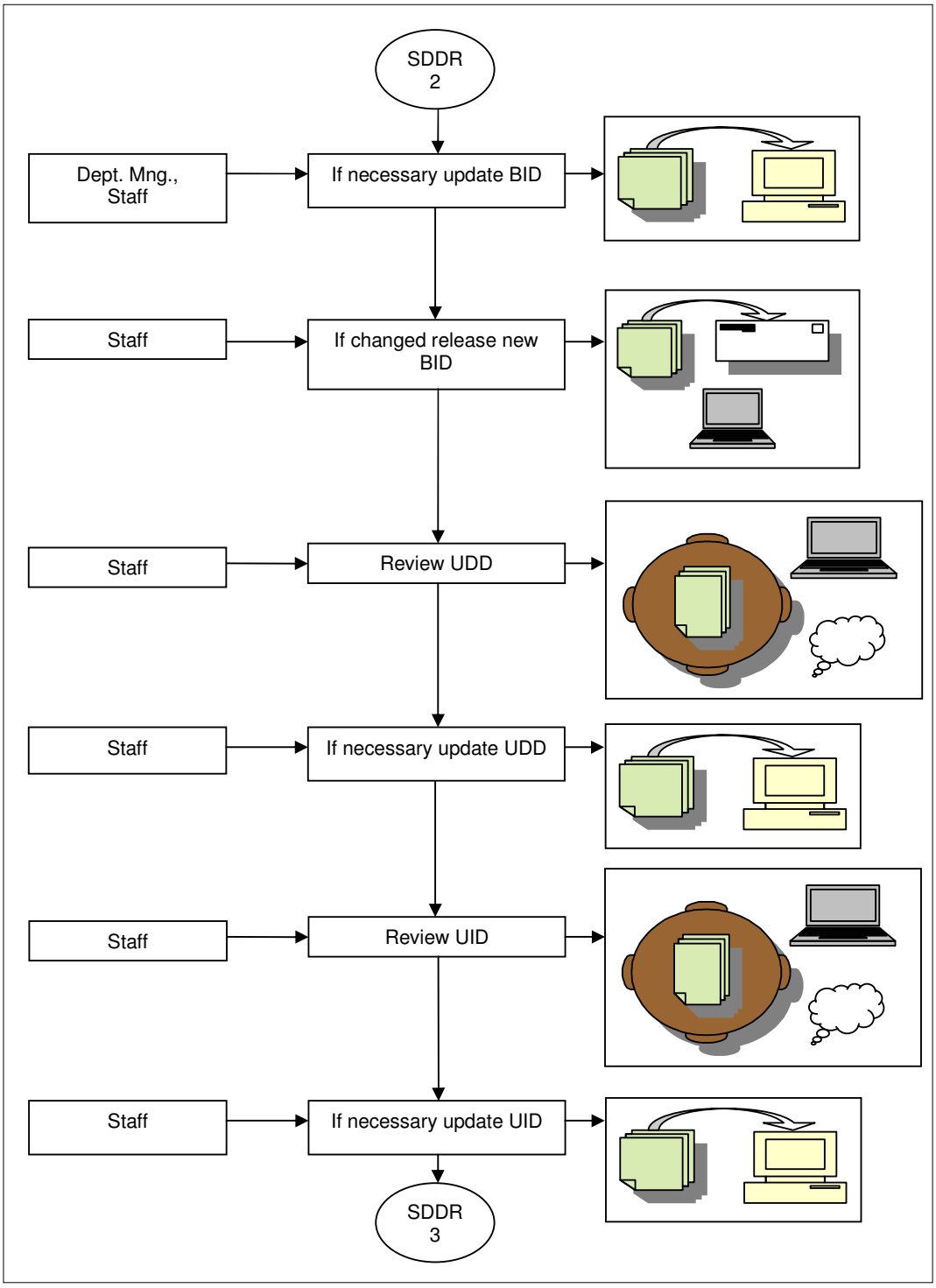

Figure 3-2 (cont'd)

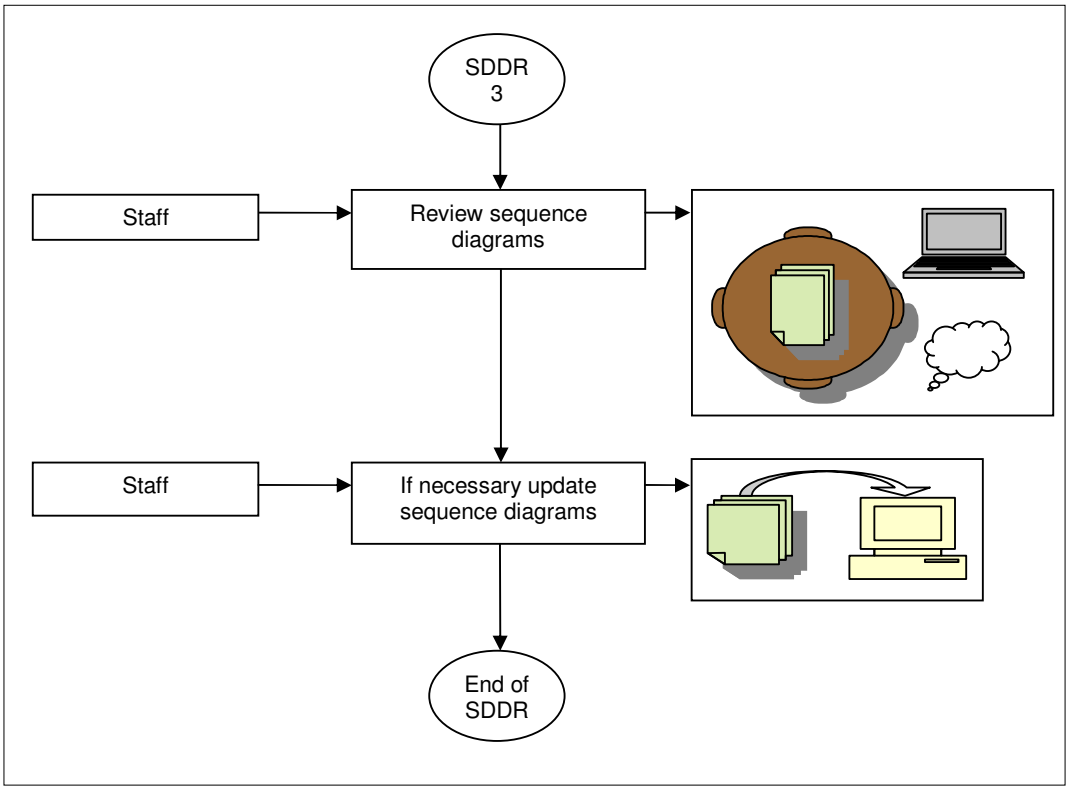

Figure 3-2 (cont'd)

3.2 Measurements

Metric values given in this section are calculated according to Selçuk Güceğlioğlu's study [14]. For this purpose the associated review activities and metrics for these steps are determined. These activities are given in Appendices A.1 and A.2 for SADR and SDDR respectively. The calculated measurements for SADR and SDDR processes are given in Table 3-5:

Table 3-5 Measurements for the SADR and SDDR processes

Metrics	SADR (Number of activity = 17)	SDDR (Number of activity = 19)
Complexity	X(1) = 1- 3/17 = 0,82 X(2) = 1- 3/17 = 0,82 X(3) = 1- 2/17 = 0,88	X(1) = 1- 5/19 = 0,74 X(2) = 1- 2/19 = 0,89 X(3) = 1- 5/19 = 0,74
Coupling	X = 1- 5/17 = 0,7	X = 1- 7/19 = 0,63
Failure Avoidance	X = 7 / 17 = 0,4	X = 12 / 19 = 0,63
Restorability	X = 9 / 17 = 0,53	X = 9 / 19 = 0,47
Restoration Effectiveness	X = 9 / 17 = 0,53	X = 9 / 19 = 0,47
Functional Adequacy	X = 5 / 17 = 0,29	X = 7 / 19 = 0,37
Functional Completeness	X = 1 – 2/3 = 0,33 [1]	X = 1 - 4 / 5 = 0,20[1]
IT Usage	X = 6 / 17 = 0,35	X = 9 / 19 = 0,47
IT Density	X = 3 / 4 = 0,75 [2]	X = 7 / 7 = 1
Computational Accuracy	Not Applicable	Not Applicable
Data Exchangeability	No interaction	No interaction
Access Auditability	X = 3 / 3 = 1	X = 4 / 4 = 1
Functional Understandability	X = 17 / 17 = 1	X = 19 / 19 = 1
Existence in Documents	X = 0	X = 0
Input Validity Checking	Not Applicable	Not Applicable
Undoability	X = 17 / 17 = 1	X = 19 / 19 = 1
Attractive Interaction	X = 3 / 4 = 0,75	X = 5 / 7 = 0,71

[1] This metric is calculated due the number of activities defined in the documents.

[2] The documents are: SAD, the outline(s) for the meeting(s), the task assignments and the schedule, and finally the project plan. The usually missing one is the outline document.

3.3 Improvement Suggestions Based on Assessment of the As-Is Process

SED has a good reputation in company X as being a department that successfully accomplishes all the tasks they undertake. However, in this department a complete and an effective Software Development Process (SDP) has not been applied until now. Usually, the as-is process does not comply with the process on paper. Without completing the detailed design, and design review processes the implementation process is started. Verification and validation processes are neither rigorous, nor sufficient. When comes to documentation, as one of our design leaders points out, because of aiming to bring up the product as soon as possible there are many missing parts. Because there are absences or insufficiencies in the design, review, and problem tracking documents, the same problems are encountered in the life-cycle of the project and work steps are repeated unnecessarily. These cause loss of time and effort.

As a result, the experiences comply with the literature in that, since the defects cannot be discovered early in the project life-cycle it takes more time to fix them in the later phases yielding late delivery of the product and higher development and maintenance costs. Although there is consensus among the staff including the department manager, design leaders, and senior engineers as that the process should not be like this, it has not yet been able to implement a complete process.

One may well ask after presenting such a negative scene how can such a department have a good reputation and be successful? The answer lies in the only most important asset in the software engineering discipline as Pressman points out: staff. The SED department has around 10 software engineers. The mean experience of the department staff is more than 7 years, all in embedded projects, and almost all of the staff have gained this experience in house.

So experienced but cannot deploy a process, why? The basic reasons why a complete process cannot be followed are [1]

1. Reasons based on project planning and application of the plan
   Sometimes to create better budget figures, sometimes by not properly evaluating the effect of the current works on the new project, the new project plan cannot be formed accurately. So, schedule and task force cannot be planned accurately and all the process gets affected. The project progresses although the design, the design review, and code review processes are incomplete or insufficient. The verification and validation processes are not sufficient.

   Example 1: For X1 project it was planned to complete the design in 18 months and to deliver the product in 24 months. But it took 22 months to complete the design and 32 months to deliver the product. 4 problems had remained unresolved at the time of delivery and it took 6 months to correct these problems.

   Example 2: X3 project was started while the maintenance and delivery of X1 project was continuing and X2 project was incomplete. Since work force needs of these projects were not considered accurately while planning project X3, design activities of project X3 had to be hindered for 8 weeks because of the tasks related to projects X1 and X2.

---

[1] Most of the reasons/suggestions are first introduced by the author. Then two methods have been followed to take the senior staff's comment on both these items and the process and how it can be improved. The first method was to discuss with the senior staff on a one to one basis. The second method was presenting this section and asking them to add their comments and improvement suggestions to it. The items given here are a result of such discussions. The different ideas other than the generally agreed ones are given in parentheses.

Example 3: After design process of X3 project was completed it was requested to add a service that was not planned at the beginning. This delayed the delivery of the product for 2 weeks.

Example 4: Since it was not possible to verify and validate this new service satisfactorily, a problem was found after delivery. As a result of not following the project plan, a new version of the project got started which could be named as a new project, X4. So, the problem remained unresolved for 6 months with no action taken.

Example 5: X3 project was delivered with 4 problems to be resolved after delivery. 3 other problems were reported after delivery as well.

Example 6: Design process for X3 project was delayed for a total of 3 weeks because of unplanned demonstrations of the product.

2. Reasons based on inability in following the design process
   Since there is not enough work force and time, the design process and documentation is not complete. For example, the reviews are either insufficient or not held at all. Some of the design leaders and senior engineers in the department think that, by having reviews regularly, even only code reviews, there can be 30 – 40 % improvement in the time schedule and quality of the project. Since there is not enough review, verification, and validation, and there are absents in the documentation, it takes more time to complete the project. Also, there are unwanted repetitions, and total quality is reduced.

   Example 1: In X3 project there was not enough time to review the design of X3m2 module. Because of this, at later phases of the project this module had become so complicated that it took 1 and 3 months more than planned to add the required S service and to fix the R service, respectively.

Example 2: The S3 problem had appeared 3 times during the life-cycle of the X1 project.

3. Reasons based on human resources and planning of human resources
   a. Misplanning of human resources

      The tasks are not distributed evenly between the staff. One of the design leaders states that this is due to unwillingness of the personnel to share their work load as well as due to planning mistakes. Usually, a task is assigned to a staff with experience on that topic. The reason for this is lack of time. However, by not assigning new staff to such topics neither a fairer and better task distribution nor staff back-up can be achieved.

      Example: The design of K module at X3 project was assigned to the personnel who participated both in X1 and X2 projects. As a result, the design process of K module could only start 1,5 months after the planned date.

   b. Insufficient human resources

      The number of personnel in SED department is not enough to apply the complete software development process with a high grade of quality.

      Example: Software engineers sometimes take on many roles during the course of a project. In all projects X1, X2, and X3, the software engineers undertook the roles of test engineer, field engineer, and even sometimes system engineer.

4. Reasons based on organization
   a. The number of system engineers is not enough or even worse, sometimes there is no system engineer at all.
   b. There is no test engineer in the organization (one of the design leaders thinks that this is a high priority topic to be fixed as soon

as possible). As a result, sufficient and satisfactory tests are not performed.

5. There is not enough hardware resources

   Example 1: In project X3, although the implementation of module M2 has been completed, integration and testing of the module had to wait for 1 month.

   Example 2: In project X3, although the implementation of module M5 has been completed, integration and testing of the module had to be postponed for a total of 8 weeks.

   Example 3: During project X3, occasionally the devices dedicated to design teams were taken for demonstration purposes. This hindered design and integration activities for a total of 4 weeks.

6. Applying immediate solutions to problems and then leaving these solutions as final in order not to miss deadlines

   Actually this item is related to items 1 and 2. It is very normal to face some problems during a project. It is understandable sometimes to have quick solutions to these problems. But leaving such immediate solutions as is without further analyzing causes the same problems to appear or new ones to arise because of the solution applied.

7. Ineffective utilization or misuse of CASE Tools

   a. Inefficient utilization and misuse of Configuration Management (CM) tools

      Example 1: Because of the improper use of the CM tool in projects X1 and X3, the code had to be re-built 18 and 12 times respectively.

      Example 2: In project X1, although 2 problems were fixed earlier, they had appeared again in the further phases.

      Example 3: In project X3, for problems P and D, a total of 10 days has been spent to find the problems and then it is

understood that it is again due to misuse of CM tool. The code had to be built 4 times again because of these problems.

b. Ineffective usage of problem follow-up tools.

8. Not considering vacations in the project plan and inefficient use of vacations

   Usually, the staff is requested to postpone their holidays to meet the schedule. This directly affects the efficiency and motivation of the staff.

There is motivation both in the management and the staff to solve at least some of these problems. It is accepted that the software development process should not be applied as it has been up to now.

Software development process improvement studies have started. As a first step, the defined process document is being revised by a process action group consisting of the members of the department staff. There are some organizational changes/improvements under way as well. A test engineering department is being formed. There are also new software engineers being recruited to the design departments.

However, most of the staff in the department think it will take some time to have things changed dramatically because of lack of human resources and ongoing projects.

In spite of this, the author believes that by taking some measures such as including staff in the improvement studies; informing staff more about the process and expected outcomes; applying procedures strictly; distributing the tasks more evenly; a better software development process can still be achieved.

The fundamental suggestions for a better software design, compiled by interviewing the senior staff including the author are as given below:

1. Suggestions for better project planning:
   a. More time should be allocated to the design process

Considering the weaknesses and absent parts in the design, design verification/review, and documentation of the projects until now, more time should be given to avoid discrepancies in the process and the project.

b. The possible effect of current work to the project should be considered more elaborately.
c. The time needed to learn and apply new topics should be explicitly accounted for.
d. Customer requirements should be understood more accurately and hence, system requirements supplied to the design teams should be more precise and more comprehensive.

2. Suggestions regarding the design process and its effective application:
   a. Especially by reviewing all designs, by well documenting the designs and the reviews a significant improvement can be achieved in the design quality.
   b. Using templates for the design reviews can help to improve design quality. Since usually similar things are done in the department, such templates can help to shorten the design time. Also by using design templates a design library can be formed and newcomers to the team can be integrated more quickly.
3. Suggestions for handling the change requests and updates:
   a. All change and update requests should be analyzed in detail considering the effects to the design.
   b. How such change or new feature requests will affect the schedule should be elaborately analyzed and planned. If such requests impose too much change or effort either they should be refused, or, they should be considered as a new project.
   c. The improved process should apply to updates as well.
4. Suggestions regarding the use of CASE tools more widely and frequently:

CASE tools have been used in the department for a long time. But they should be used more widely and frequently (in all designs where possible) during the design and verification processes.

Most of the CASE tools come with features such as simulation, scenario testing, regression testing, etc. Verification of the design can be done more comprehensively by using such features. By doing so, possible defects in the design can be found earlier. Also, the effect of possible updates can be evaluated more accurately.

The investments have been budgeted for this purpose. At the time the present study was being conducted, new tools have been ordered.

3.4 Metrics Based Improvement Suggestions

In this section the measurements obtained by applying Selçuk Güceğlioğlu's approach to SED are investigated. Improvement suggestions are made where appropriate based on these measurements.

The first section below covers suggestions for the SADR Process; second section covers suggestions for the SDDR Process.

3.4.1 Suggestions for the SAD Review Process

Complexity metric (X(1) = 0,82; X(2) = 0,82; X(3) = 0,88)

The interpretation of measured value (0 <= X <= 1): Higher the value of X (1), X (2), X (3), better is the analyzability [14].

The complexity metric values measured for all groups (X1, X2, X3) are close to 1, which is desirable. There is no need to further improve these metric values. However, these results do not reflect the actual complexity of the whole process. Because, a specific activity, namely, review of SAD, has a much more significant effect on complexity than the other activities, and this effect should be reflected to the measurements by weighing the activities. Hence, the author suggests, the given definition of the metric should be changed to cover the weight of an activity on the whole process.

Coupling metric (X = 0,7)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the analyzability [14].

The coupling metric value measured is close to 1. However, it is not possible to discuss this value due to the given definition of this metric. Because, since all the activities do not have equal contribution to the whole process. The coupling is observed during the review of SAD and this has a significant effect on the whole process. However, this result can not be interpreted from the measured value. Hence, the author suggests, the given definition of the metric should be changed to cover the weigh of an activity on the whole process.

Failure Avoidance metric (X = 0,4)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the failure avoidance [14].

Failure avoidance metric value measured is low. However the result is normal due to the nature of activities. Most of the activities identified with “No review, inspection, checkpoint or similar techniques” attribute are so trivial that further improvement to avoid failure avoidance is not practical.

Restorability and Restoration Effectiveness metrics (X = 0,53)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the restorability and the restoration effectiveness [14].

The measured value is quite far from the ideal one. When we inspect the process model, the first thing that we notice is that there are no formal review reports. This should be compulsory in any organization and strictly followed. There must be a review report and it should be documented.

Also, reviews should be planned at the beginning in any project. This way, people can take necessary actions/measures to avoid any delays in the projects or project plan can be revised if necessary. Planning reviews is also a quality requirement. So, arrangement of review meetings should be recorded.

Applying these precautions can help to improve the measured value of this metric.

Functional Adequacy metric (X = 0,29)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the functional adequacy.

It is understood that most of the activities that take place in practice are missing in the regulatory documents. This causes a weakness in the process. Because the activities are not present in the documents they cannot be planned, questioned, reviewed. To avoid the discrepancies caused by this situation at the software life-cycle process the regulatory documents should be updated.

Functional Completeness metric (X = 0,33)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the functional completeness.

The measured value is far away from the ideal one. Most of the activities are missing in the regulatory documents. The suggestion in this case is obvious: as suggested in the previous metric, update the regulatory documents to cover all the activities as necessary.

IT Usage metric (X = 0,35)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, more is the IT usage.

Although the measured value for this metric is not satisfactory, an improvement suggestion can not be made because of the intrinsic characteristics of the activities where IT is not used. i.e. review meetings.

However, the author suggests using design templates in the design processes. If this can be accomplished, perhaps IT usage can be involved in review meetings as well to compare the design with the template.

IT Density metric (X = 0,75)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, more is the IT density.

An IT based document should be prepared describing the outline for each meeting.

Existence in Documentation metric (X = 0)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, the more complete is the documentation.

The measured value for this metric is really stunning. Unfortunately there is no document describing the activities. As a result, the activities are performed according to the understanding and experience of the staff involved. The activities should be well defined and clearly explained with examples where necessary/applicable. This metric points towards one of the basic problems in SED.

Undoability metric (X = 1)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the undo ability.

The value measured for undo ability metric is 1, which is the ideal value. It is measured considering all activities, whether recorded or not.

3.4.2 Suggestions for the SDD Review Process

Complexity metric (X(1) = 0,74; X(2) = 0,89; X(3) = 0,74)

The interpretation of measured value (0 <= X <= 1): Higher the value of X (1), X (2), X (3), better is the analyzability [14].

When we look at the complexity metric values we see that they are close to 1, which is desirable. There is no need to further improve these metric values. However, the discussion presented in the section 3.4.1, for the SADR process, is valid here also.

Coupling metric (X = 0,63)

The interpretation of measured value (0 <= X <= 1):

Higher the value of X, better is the analyzability [14]. As with SADR it is not possible to discuss the measured value due to the given definition of this metric.

Failure Avoidance metric (X = 0,63)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the failure avoidance [14].

Failure avoidance value measured for SDDR is better than the value for SADR (0,4). This is due to the given definition of this metric.

Actually, the author thinks the quality of the reviews should be considered as well as whether they exist or not. Considered this way, by increasing the number of reviews held and applying them wherever they are called for, without exception, the value of this metric could be improved. Also, by using design and review templates, by following review results, not only will the measured value be higher, but also the quality achieved will be better.

However, as with SADR, there can not be improvements for the activities identified with “No review, inspection, checkpoint or similar techniques” attribute. Those activities are so trivial that further improvement to avoid failure avoidance is not practical.

Restorability and Restoration Effectiveness metrics (X = 0,47)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the restorability and the restoration effectiveness [14].

The measured value is far from the ideal one. As with SADR, here again, the first thing that is noticed is that there are no formal review reports. So the improvement suggestion follows the one for SADR: There must be a review report and it should be documented.

Functional Adequacy metric (X = 0,37)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the functional adequacy [14].

It is understood that most of the activities that take place in practice are missing in the regulatory documents. This causes a weakness in the process. Because the activities are not present in the documents they cannot be planned, questioned and reviewed. To avoid the discrepancies caused by this situation, the regulatory documents should be updated to cover the activities that take place in practice.

Functional Completeness metric (X = 0,20)

The interpretation of measured value (0 <= X <= 1):

Higher the value of X, better is the functional completeness [14]. The measured value is far away from the ideal one. Most of the activities are missing in the regulatory documents. The suggestion in this case is obvious: as suggested in the previous metric, update the regulatory documents to cover all the activities as necessary.

IT Usage metric (X = 0,47)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, more is the IT usage [14].

As with SADR, although the measured value for this metric is not satisfactory, an improvement suggestion can not be made because of the intrinsic characteristics of the activities where IT is not used. i.e. review meetings.

However, the author suggests using design templates in the design processes. If this can be accomplished, perhaps IT usage can be involved in review meetings as well to compare the design with the template.

Existence in Documentation metric (X = 0)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, the more complete is the documentation [14].

As stated before for SADR, the measured value for this metric is really stunning. Unfortunately there is no document describing the activities. As a result the activities are performed to staff's understanding of it and experience. The activities should be well defined and clearly explained with examples where necessary/applicable. This metric points towards one of the basic problems in the department.

Undoability metric (X = 1)

The interpretation of measured value (0 <= X <= 1): Higher the value of X, better is the undo ability [14].

The value measured for undo ability metric is 1, which is the ideal value. It is measured considering all activities, whether recorded or not.

3.5 Discussion

When we look at the improvement suggestions gathered from the staff and compare them with the metric based suggestions based on Selçuk Güceğlioğlu's study [14], we note that there are matching areas such as Failure Avoidance metric, Functional adequacy metric and Existence in Documentation metric.

There are also some items which are introduced by the staff but do not have a counterpart in the metrics. E.g. project planning. Similarly, there are

metrics which can form a basis for improvement studies but not mentioned by the staff. i.e. Complexity metric.

Selçuk Güceğlioğlu's study [14] actually is based on an assumption that in an organization, the regulations, documentation and process should be complete. However, at SED we see that regulations are not complete and clear and there are discrepancies with the process. Hence on-paper process is different than as-is process. Also, there are missing parts in documentation.

These defects describe some of the mismatches. But, the author thinks that there is still another important reason describing why some of the suggestions from the staff and suggestions based on metrics point towards different directions. That is, applying the metrics with their given definitions without considering their contribution to the whole process does not give a correct reflection of the metric in question. One should weigh the activities in relation to the whole process and possibly in context of the organization in which the process is realized, before calculating the metrics.

As an example consider the complexity metric. The complexity of activities such as "Send SAD for Review" has the same effect as the "Review SAD" activity. However, review of SAD is really a complex activity and has a major effect on the whole process. Much more time and effort is spent for it.

"Failure avoidance" (FA) metric is another good example of why some of the metrics should be weighted. Again consider the "Send SAD for Review" activity. Attributing equal weights to this activity and the "Review SAD" activity does not give a correct idea about the quality of the failure avoidance of the whole process. As not all the activities have the same effect on the process, they should either be weighted or the definition of some metrics should be changed.

Activities should be considered from the viewpoint of their contribution to the process. Analyzing how the process is affected by the result of an

activity can help us weight the activities. A new definition of the FA metric could be:

X = A / B

A = Number of activities in which review, inspection, checkpoint or similar techniques are applied

B = Number of activities in which review, inspection, checkpoint or similar techniques are required

Furthermore, this definition by itself is not sufficient. We should somehow measure the quality, that is, measure the completeness of the FA techniques applied. We can name this metric “Failure Avoidance Completeness” metric.

Here is how this new metric can be applied:

In general, applying the measures these metrics intuitively suggests will not be sufficient, because, in an organization there is a need to know how to apply them. Saying “Put reviews in the project plan, then document the reviews and the reports” will not help more than having documents of something called review and having accomplished the rules on paper. The real aim is to have an effective review: to have the design reviewed effectively so that the defects are discovered and solutions are proposed. To do so, reviews should be held against a template. There should be review templates as well as design templates. There should be design and review libraries where staff can find the rules of thumb for a project. Then reviews would be certainly beneficial. Otherwise review meetings will mostly be like a social event.

As a result, ISO 9126-3 and Selçuk Güceğlioğlu’s study [14] which is based on use of metrics to improve the process at an organization can be used in process improvement studies. However, taking the standard and the mentioned study as rigid templates, without adapting them to the specific organization needs may not yield the desired improvement.

As a starting point, an organization should analyze itself very frankly. Then, it should weight the activities according to their contribution to the

process. After activities are weighted elaborately, metrics should be calculated. Then an improvement study based on the values of the metrics can be planned.

Chapter 4

IMPROVED PROCESS MODELS

4.1 Improved SAD and SAD Review Processes

In this section suggested modifications to the SAD Process are presented. *The modifications are given in italic font to note easily*. The rationale for every modification is presented briefly.

4.1.1 Improved Software Architecture Design Process

Table 4-1 Improved SAD Process

Step	Original	Suggested	Change Rationale
Inputs	1. Software Requirements Specification (SRS).	No change	N/A
	2. Hardware Architecture Plan.		
	Does not exist	*3. If exists, Design Template*	*An approved, validated design can help to avoid design defects and to shorten design time dramatically.*
		4. Software Development plaN (SDN)	*To assess/measure how we are doing with the plan.*
		5. Measurement Guideline	*To describe how to measure the metrics.*
		6. Metrics Document	*To give metrics to be measured at this stage.*
To Do	1. Requirements are analyzed.	No change	N/A
	2. The software blocks required for the equipment/system, their functions and place on the hardware are determined.		

Table 4-1 (cont'd)

Step	Original	Suggested	Change Rationale
To – Do cont.	*Does not exist*	*3. State the design decisions explicitly and document them.*	*To avoid possible confusions about the design. The rationale behind a decision will help while reviewing, and while maintaining/upgrading as well. The staff will not doubt why they have done it that way rather than other if they state it explicitly here. Even the very obvious reasons must be given.*
		4. Form/Update Requirements Traceability Matrix (RTM) to represent which requirement is met at which block.	*It is necessary to be able to follow where the requirements are met. This will also help while analyzing the possible effects of a requirement change.*
	5. Work load and timing information is updated if necessary.	No change	N/A
	6. Tasks are updated if necessary.		
	7. Units and their functions are determined.		
	Does not exist	*8. Update RTM to represent which requirement is met at which unit.*	*A more detailed RTM allows the organization to be more accurate while analyzing the effects of a possible change, or during maintenance.*

Table 4-1 (cont'd)

Step	Original	Suggested	Change Rationale
To – Do cont.	9. The block and unit interfaces are determined at high level.	No change	N/A
	Does not exist	*10. Update RTM to represent traceability to an interface if it is dependent to a requirement.*	*To be able to analyze the effects of a possible change.*
		11. Measure how compatible we are with the SDN.	*To be able to take action if necessary.*
		12. Calculate the defined metrics.	*To assess how well/bad the staff are doing and take action if necessary.(*)*
		13. Update the SDN if necessary.	-
Roles	1. Software Project Manager.	No change	N/A
	2. Software Engineers.	*If "Software Engineers" are carrying other roles as well, consider how these can affect the SDN.*	*More accurate the SDN, better will be the process and the quality of the product.*
	3. Software Test Engineers.	No change	N/A

(*) The author is aware that this assessment technique, together with target and reference values and the actions to be taken under specified conditions, needs to be precisely specified for a mature process. This, however, is beyond the scope of this thesis study, and must be considered as a necessary extension towards higher maturity of the overall software development process.

Table 4-1 (cont'd)

Step	Original	Suggested	Change Rationale
Outputs	1. Software Architecture Design Document (SADD).	No change	N/A
	Does not exist	*2. Updated SDN if applicable.*	-
		3. Metrics document updated with the measurements.	*Records will guide the staff both in the current and the future projects.*

4.1.2 Improved Software Architecture Design Review Process

Table 4-2 Improved SAD Review Process

Step	Original	Suggested	Change Rationale
Inputs	1. Software Requirements Document.	No change	N/A
	2. SADD.		
	Does not exist	*3. Software Architecture Design Review Guideline.*	*To avoid loss of precious time the staff need to know how they should review the design and how they should be prepared for the review meeting.*
		4. Software Architecture Design Review / Inspection Checklist.	*To be sure that the design is reviewed completely the staff must use a checklist.*
		5. RTM	*To be sure they have the final reflection of the requirements the staff must check and update RTM if necessary*
		6. If exists, Design Template	*An approved, validated design can help to avoid design defects and to shorten design time dramatically.*
		7. Software Development plaN (SDN)	*To assess/measure how compliant the staff are with the plan.*
		8. Measurement Guideline	*To describe how to measure the metrics.*

Table 4-2 (cont'd)

Step	Original	Suggested	Change Rationale
Inputs cont.	*Does not exist*	*9. Metrics Document*	*To give metrics to be measured at this stage (*)*
To – Do	Software architecture design is reviewed and updated if necessary.	*Removed*	*To explicitly give the review steps.*
	Does not exist	*1. Send SAD to the staff and ask to get prepared for the design review meeting.*	*To force the staff to review the SAD and hence to have real benefit from the review meeting.*
		2. Collect feedback about SAD.	*To force the staff to review the SAD.*
		3. Verify that architecture meets all requirements.	*To be sure that the right product is being designed.*
		4. Verify that architecture is testable and/or verifiable	*To see if the staff has done what they should, they have to test what they did. For this purpose, they have to plan the tests, test methods, and the test schedule as early as possible. Otherwise it will be too late and more costly to make change for the tests. The staff should consider how they can test their designs and invest if necessary.*

(*) The author is aware that this assessment technique, together with target and reference values and the actions to be taken under specified conditions, needs to be precisely specified for a mature process. This, however, is beyond the scope of this thesis study, and must be considered as a necessary extension towards higher maturity of the overall software development process.

Table 4-2 (cont'd)

Step	Original	Suggested	Change Rationale
To – Do cont.	*Does not exist*	*5. Verify that design decisions are correct or appropriate.*	*Check that the design is appropriate for its intend of use and also it is easy to implement, manage, maintain, and update.*
		6. Update RTM if necessary.	*A more detailed RTM allows the organization to be more accurate while analyzing the effects of a possible change, or during maintenance.*
		7. Update metrics document if necessary.	*The changes in SAD may force to update metrics.*
		8. Measure how compatible we are with the SDN	*To be able to take action if necessary.*
		9. Update the SDN if necessary	-
		10. Calculate the defined metrics	*To assess how well/bad the staff are doing and take action if necessary.*
Roles	Same as SAD	No change	N/A
Outputs	1. SAD review report	No change	N/A
	2. Updated SADD		
	Does not exist	*3. Updated SDN if applicable.*	-
		4. Updated RTM if applicable.	-
		5. Metrics document updated with the measurements.	*Records will guide the staff both at the current and the future projects.*

4.2 Improved SDD and SDD Review Processes

In this section modification suggestions to the SDD Process are presented. *The modifications are given in italic font to note easily*. The rationale for every modification is presented briefly.

4.2.1 Improved Software Detailed Design Process

Table 4-3 Improved SDD Process

Step	Original	Suggested	Change Rationale
Inputs	1. Software Requirements Document.	No change	N/A
	2. SADD.		
	Does not exist	*3. If exists, Design Template*	*An approved, validated design can help to avoid design defects and to shorten design time dramatically.*
		4. Software Development plaN (SDN)	*To assess/measure how the staff are doing with the plan.*
		5. Measurement Guideline	*To describe how to measure the metrics.*
		6. Metrics Document	*To give metrics to be measured at this stage.*
		7. RTM	*The staff must check and update RTM if necessary.*
To Do	1. Block interfaces are defined.	No change	N/A

Table 4-3 (cont'd)

Step	Original	Suggested	Change Rationale
	Does not exist	*2. State the design decisions explicitly and document them.*	*To avoid possible confusions about the design. The rationale behind a decision will help while reviewing, and while maintaining/upgrading as well. The staff will not doubt why they have done it that way rather than other if they state it explicitly here. Even the very obvious reasons must be given.*
	3. Communication mechanism between blocks is determined (hardware and software).	No change	N/A
To – Do cont.	*Does not exist*	*4. State the design decisions about a communication mechanism explicitly and document them.*	*Please refer to item #2 above.*
	5. Units are detailed. Functions are determined. If necessary, a unit can involve other unit(s).	No change	N/A
	6. Unit interfaces are defined.		
	Does not exist	*7. State the design decisions about an interface explicitly and document them.*	*Please refer to item #2 above.*

Table 4-3 (cont'd)

Step	Original	Suggested	Change Rationale
To – Do cont.	8. Communication mechanism between units is determined (i.e. Message, function call, semaphore).	No change	N/A
	Does not exist	*9. State the design decisions about a communication mechanism explicitly and document them.*	*Please refer to item #2 above.*
	10. For every process that will be carried out by the software, sequence diagrams are prepared per block and/or unit basis.	No change	N/A
	11. Data structures are formed.		
	Does not exist	*12. Measure how compatible we are with the SDN.*	*To be able to take action if necessary.*
	Does not exist	13. Calculate the defined metrics.	*To assess how well/bad they are doing and take action if necessary.*
		14. Update the SDN if necessary.	-
		15. Update RTM if necessary.	-
Roles	1. Software Project Manager.	No change	N/A
	2. Software Engineers.	*If "Software Engineers" are carrying other roles as well, consider how these can affect the SDN.*	*More accurate the SDN better will be the process and the quality of the product.*

Table 4-3 (cont'd)

Step	Original	Suggested	Change Rationale
Outputs	1. Block Design Document (BDD).	No change	N/A
	2. Block Interface Document (BID)		
	3. If necessary, Unit Design Document (UDD)		
	4. Unit Detailed Interface Document (UDID)		
	5. Sequence diagrams per block and/or unit basis.		
	6. The header files.		
	Does not exist	*7. Updated RTM, if applicable.*	
		8. Updated SDN if applicable.	
		9. Metrics document updated with the measurements.	

4.2.2 Improved Software Detailed Design Review Process

Table 4-4 Improved SDD Review Process

Step	Original	Suggested	Change Rationale
Inputs	The outputs of the SDD section are used as inputs to this section	*Removed*	To explicitly list all the inputs
	Does not exist	*1. SRS*	*To follow the requirements*
		2. SADD	*To follow the Architecture design*
		3. All the related BDD, BID, UDD, UDID, Sequence diagram, Header files.	*Material that will help to understand the design and also the material to review as well.*
		4. If exists, Design Template.	*As stated before.*
		5. SDN	*As stated before.*
		6. Software Detailed Design Review Guideline.	*To avoid loss of precious time the staff need to know how they should review the design, how they should be prepared for the review meeting.*
		7. Software Detailed Design Review/Inspection Checklist.	*To be sure that the design is reviewed completely the staff must use a checklist.*
		8. Measurement Guideline.	*To describe how to measure the metrics of this stage.*
		9. Metrics Document.	*To give metrics to be measured at this stage.*

Table 4-4 (cont'd)

Step	Original	Suggested	Change Rationale
Inputs cont.	*Does not exist*	*10. RTM*	*The staff must check and update RTM if necessary.*
To Do	Software architecture design is reviewed and updated if necessary.	*Removed*	*To explicitly list all the inputs*
	Does not exist	*1. Arrange review meetings and collect feedback about the designs.*	*To force the staff to review the SDD and hence to have real benefit from the review meeting.*
		2. Review block interface design by the participation of the relevant designers.	-
		3. Verify that the design decisions for a block interface are correct. Update interface documents if necessary.	*Check that the design is appropriate for its intend of use and also it is easy to implement, manage, maintain, and update.*
		4. Review block design with the relevant designers.	-
		5. Verify that the design decisions for a block are correct. Update if necessary.	*Check that the design is appropriate for its intend of use and also it is easy to implement, manage, maintain, and update.*
		6. Review unit design and unit interface with the relevant designers.	-

Table 4-4 (cont'd)

Step	Original	Suggested	Change Rationale
To – Do cont.	*Does not exist*	*7. Verify that the design decisions for a unit and unit interface are correct. Update if necessary.*	*Check that the design is appropriate for its intend of use and also it is easy to implement, manage, maintain, and update.*
		8. Review sequence diagrams with the relevant designers.	-
		9. Verify that the sequences of events are correct and compatible with the design. Update if necessary.	*Check that the design is appropriate for its intend of use and also check that all the possible scenarios are covered.*
		10. Review header files with the relevant designers. Update if necessary	-
		11. Verify that all the interfaces (block/unit) are testable/verifiable. Update if necessary.	*The staff have to be sure that they have done what they should. To see that, they have to test what they did. For this purpose, they have to plan the tests, test methods, and the test schedule as early as possible. Otherwise it will be too late and more costly to make change for the tests. The staff should consider how they can test their designs and invest if necessary.*
		12. Verify that the block/unit design is testable/verifiable. Update if necessary.	
		13. Update RTM if necessary.	-
		14. Verify that all the requirements are covered and met by the design.	-

Table 4-4 (cont'd)

Step	Original	Suggested	Change Rationale
To – Do cont.	*Does not exist*	*15. Measure how compatible we are with the SDN.*	*To be able to take action if necessary.*
		16. Update the SDN if necessary.	*To assess how well/bad they are doing and take action if necessary.*
		17. Calculate the defined metrics.	-
Roles	Same as SDD.	No change	N/A
Outputs	1. SDD Review Report	No change	N/A
	2. If there are any changes, updated versions of the documents in the "SDD Outputs" section.		
	Does not exist	*3. Updated RTM if applicable.*	-
		4. Updated SDN if applicable.	-
		5. Metrics document updated with the measurements.	-

4.3 Measurements

The calculated measurements for the improved SADR and SDDR processes are given in Table 4-5:

Table 4-5 Measurements for the Improved Process Models

Metrics	**Improved SADR (# of activity = 11[1])**	**Improved SDDR (# of activity = 17[1])**
Complexity	X(1) = 1 - 1/11 = 0,91 X(2) = 1 - 3/11 = 0,73 X(3) = 1 - 1/11 = 0,91	X(1) = 1 - 1/17 = 0,94 X(2) = 1 - 11/17 = 0,35 X(3) = 1 - 1/17 = 0,94
Coupling	X = 1 - 3/11 = 0,73	X = 1 - 10/17 = 0,41
Failure Avoidance	X = 6 / 11 = 0,54	X = 12 / 17 = 0,70
Restorability	X = 10 / 11 = 0,90	X = 17 / 17 = 1
Restoration Effectiveness	X = 7 / 11 = 0,63	X = 10 / 17 = 0,59
Functional Adequacy	X = 10 / 11 = 0,90	X = 17 / 17 = 1
Functional Completeness	Not Applicable	Not Applicable
IT Usage	X = 7 / 11 = 0,63	X = 8 / 17 = 0,47
IT Density	X = 10 / 10 = 1[2]	X = 11 / 11 = 1
Computational Accuracy	Not Applicable	Not Applicable
Data Exchangeability	No interaction	No interaction
Access Auditability	X = 10 / 10 = 1	X = 11 / 11 = 1
Functional Understandability	X = 8 / 11 = 0,72	X = 15 / 17 = 0,88
Existence in Documents	X = 8 / 11 = 0,72	X = 16 / 17 = 0,94
Input Validity Checking	Not Applicable	Not Applicable
Undoability	X = 10 / 11 = 0,90	X = 17 / 17 = 1
Attractive Interaction	Not Applicable	Not Applicable

[1] Number of activities is derived from the relevant To-Do section.
[2] These numbers are derived from the relevant input and output sections.

4.4 Comparison of As-Is and To-Be Measurements

The measured values of the "Failure Avoidance", "Restorability", "Restoration Effectiveness", "Functional Adequacy", "IT Usage" of SADR, "Existence in Documents" metrics for the To-Be model are better than the values for the As-Is model as expected.

Overall complexity figures for the improved model (SADR: 0,85; SDDR:0,74) and As-Is model (0,84; SDDR: 0,79) are almost the same. This is due to decreased number of activities in the To-Be model. The activities such as "arrange review meeting" which do not have a major contribution to the process have been removed.

Although the measured complexity values are similar it is worth noting here that, the To-Be model presents a better reflection of the process now. From this point of view, it is now more possible to account for the real complexity of the process and make software development plan accordingly.

Similar comments are valid for the coupling metric. The measured value of the coupling metric for the SADR is almost the same for the As-Is and To-Be models. When it comes to SDDR, the As-Is model has a better (0,63) coupling value then the To-Be model (0,41). As with complexity metric, this is normal due to the characteristics of the activities that are forming the As-Is and To-Be processes. The To-Be model presents a better process covering almost all the activities that should be present to have a better software development process. Since the new model reflects the inside of the process better, it is easier to account for each activity, hence to make up the SDN more accurately. Also, if things go wrong somewhere during the development process, it is easier to find the possible reason of defect by looking at what has been missed, where more iterations have been done, etc.

The measured value of the "Functional Understandability" metric for the To-Be model is lower then the measured value for the As-Is model. As with the above metrics, this is again due to the internal characteristics of the newly introduced activities. Some of the new activities need clarification

before being practically applicable, such as "verifying a design", "making measurements", "calculating metrics", etc. These are new topics for the staff. However, a software development process can not be without these activities. Again, the possible but necessary overheads such as training and human resources needs should be considered; the timing of these activities during the life-cycle of the project should be planned. Also, the SDN should be made up accordingly. But, as with the complexity and coupling metrics, all these new activities not only bring better predictability from the SDN point of view and analyzability, but also the quality of the software produced will be higher.

As a result, the assessment of the new model based on Güceğlioğlu's study also supports To-Be model. The measured values for the To-Be model are promising. The measurements point that new model overcomes most of the major problems associated with the current software development process.

4.5 Improved SADR Process Model

In this section, improved process model for SADR is given. The model represents the flow of activities during the review process. There are a few steps that are not present on-paper, like "Review SAD". Because, it is clear that SAD should be reviewed by the staff before the meeting.

Steps that are present on-paper are given with a "Step On-Paper (SOP)" sign above the related item. Improved SADR Model is given in Figure 4-1:

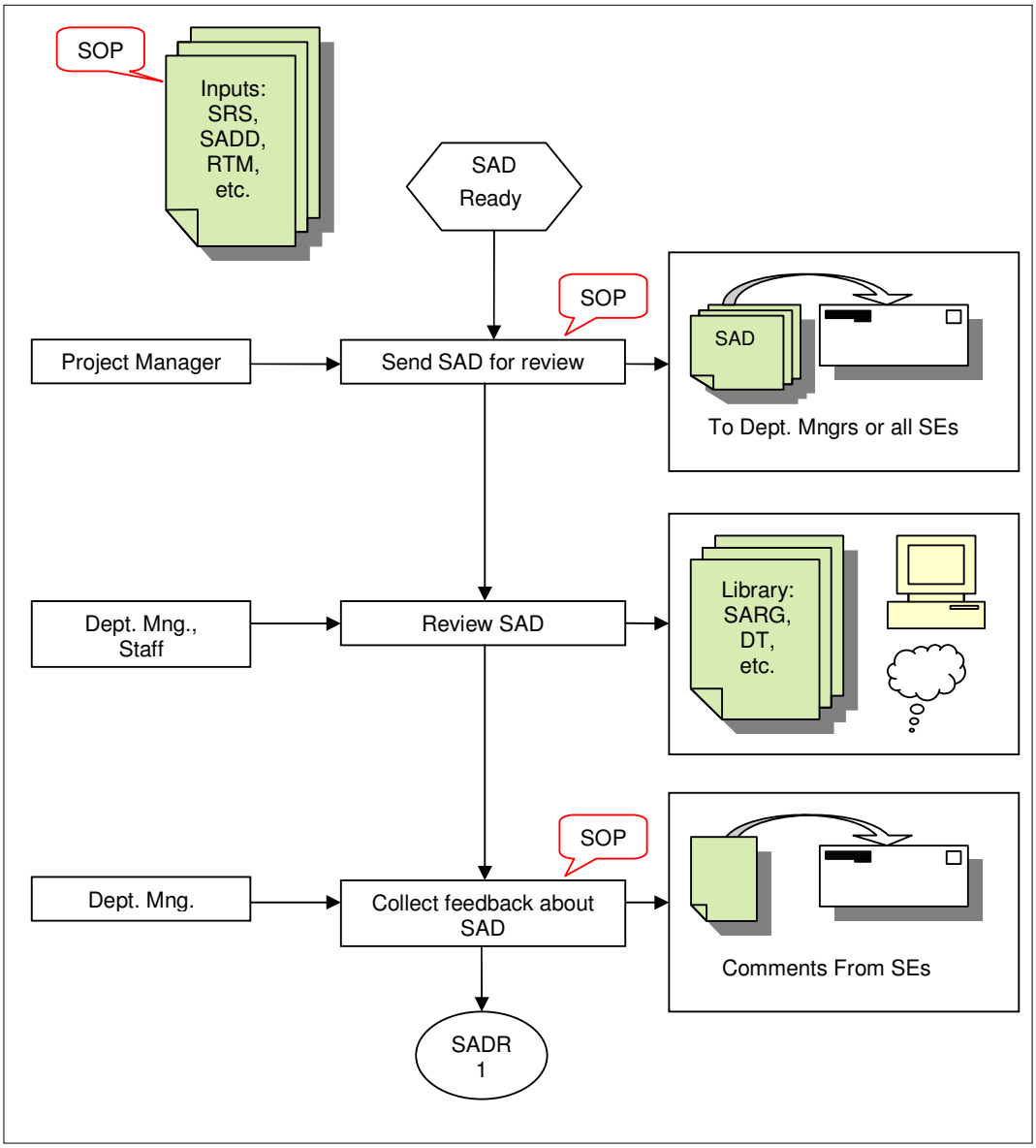

Figure 4-1 Improved SADR Process Model

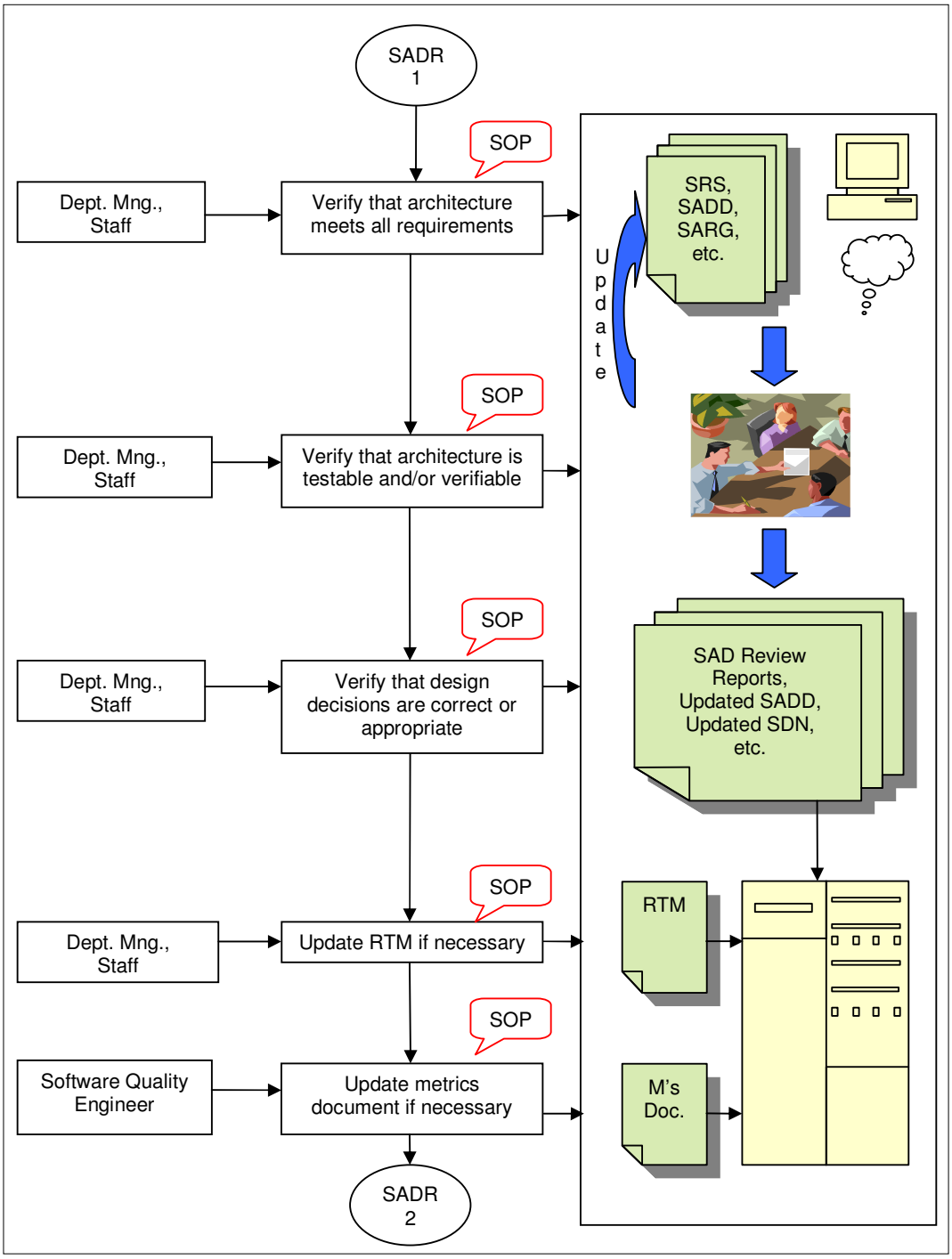

Figure 4-1 (cont'd)

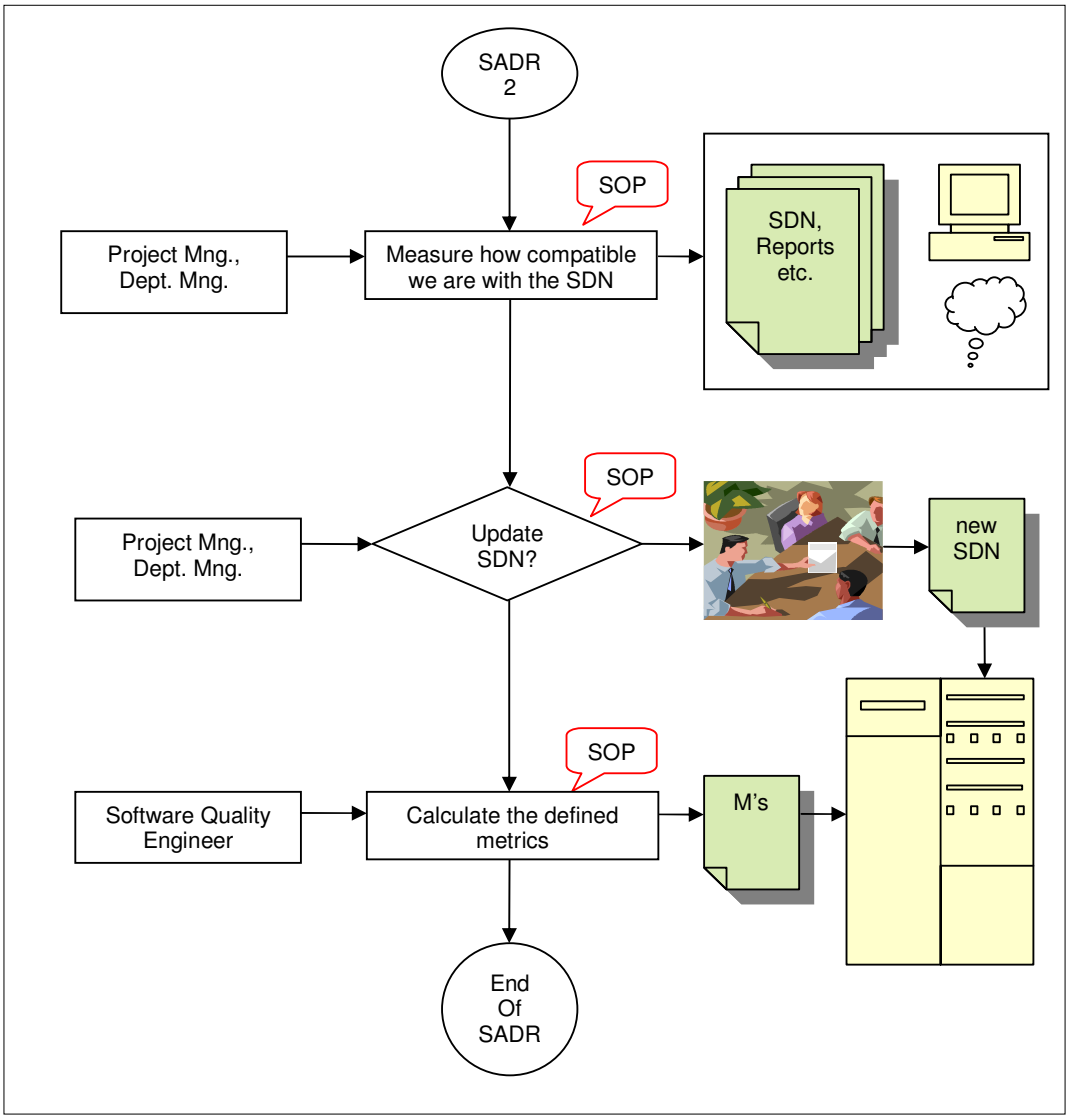

Figure 4-1 (cont'd)

4.6 Improved SDDR Process Model

In this section, improved process model for SDDR is given. The model represents the flow of activities during the review process. As with SADR model, steps that are present on-paper are given with a "Step On-Paper (SOP)" sign above the related item. Improved SDDR Model is given in Figure 4-2:

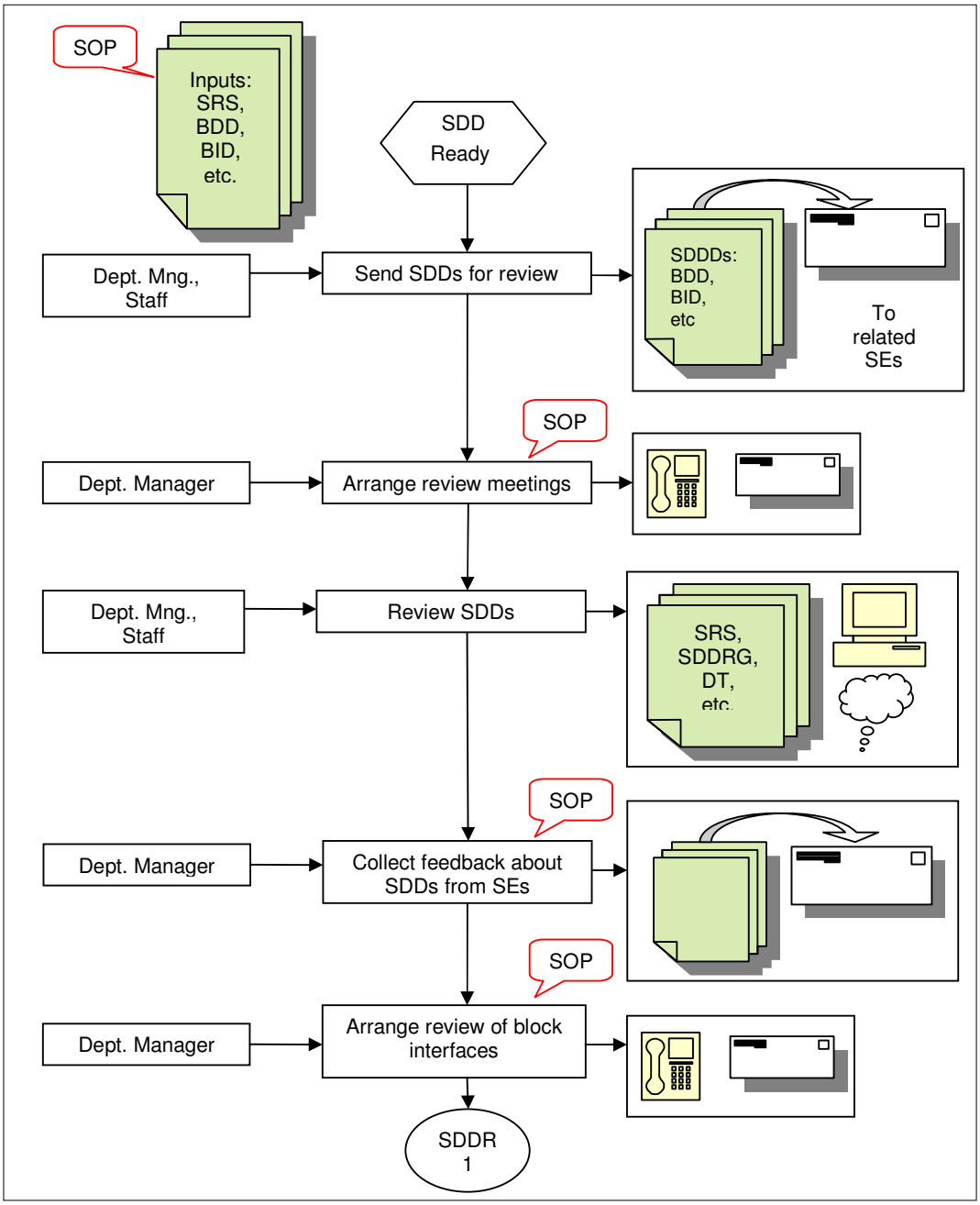

Figure 4-2 Improved SDDR Process Model

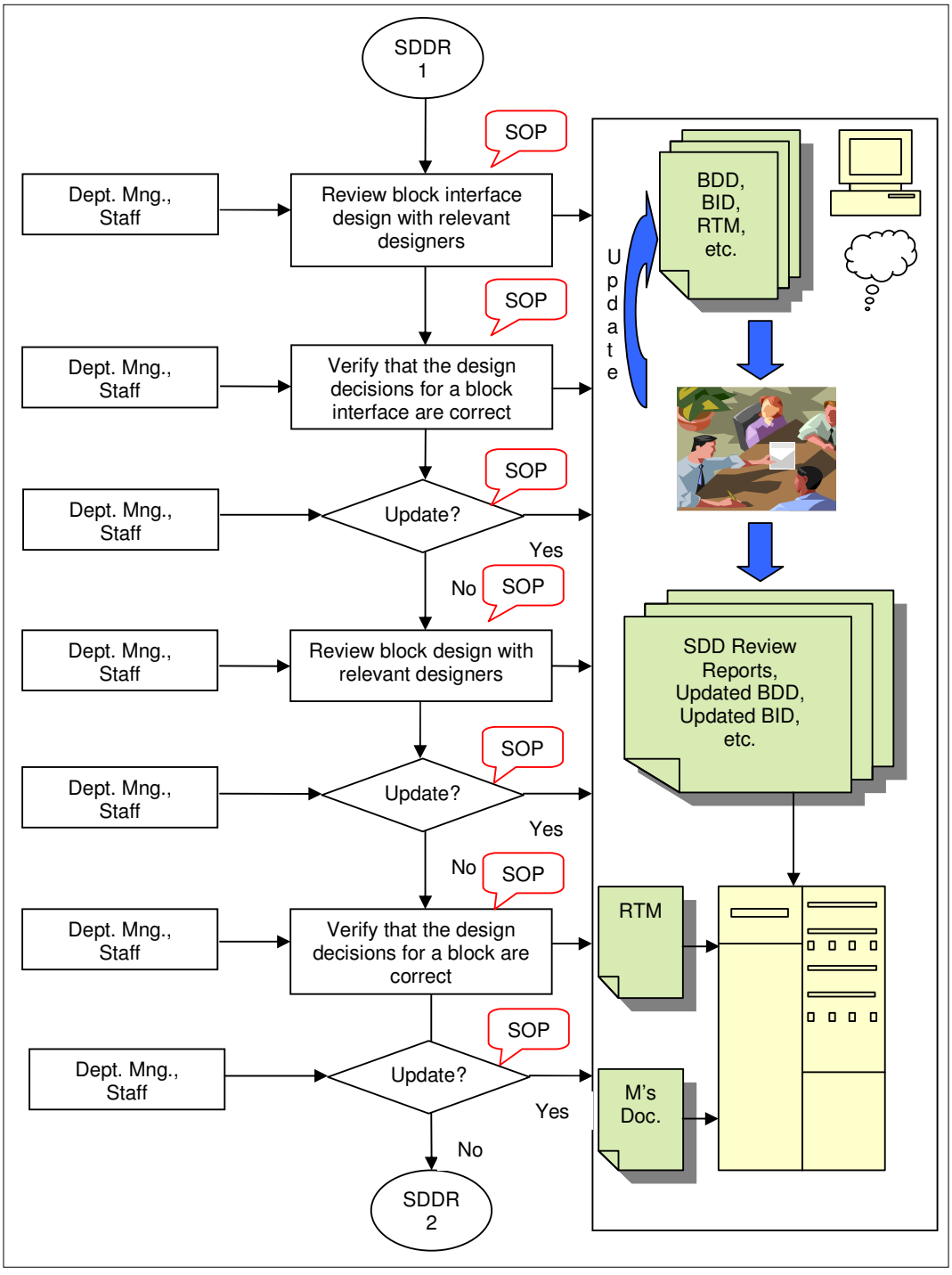

Figure 4-2 (cont'd)

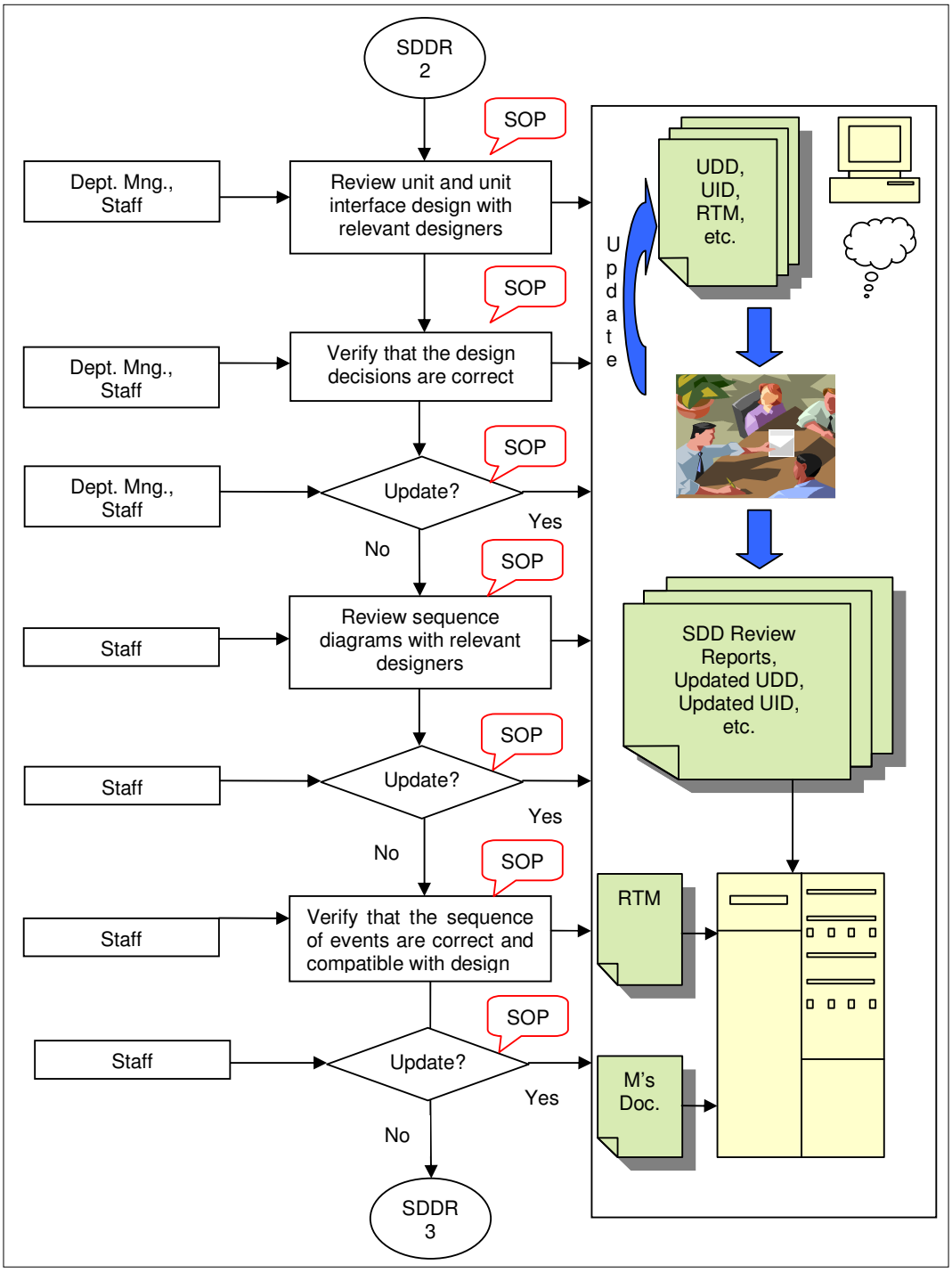

Figure 4-2 (cont'd)

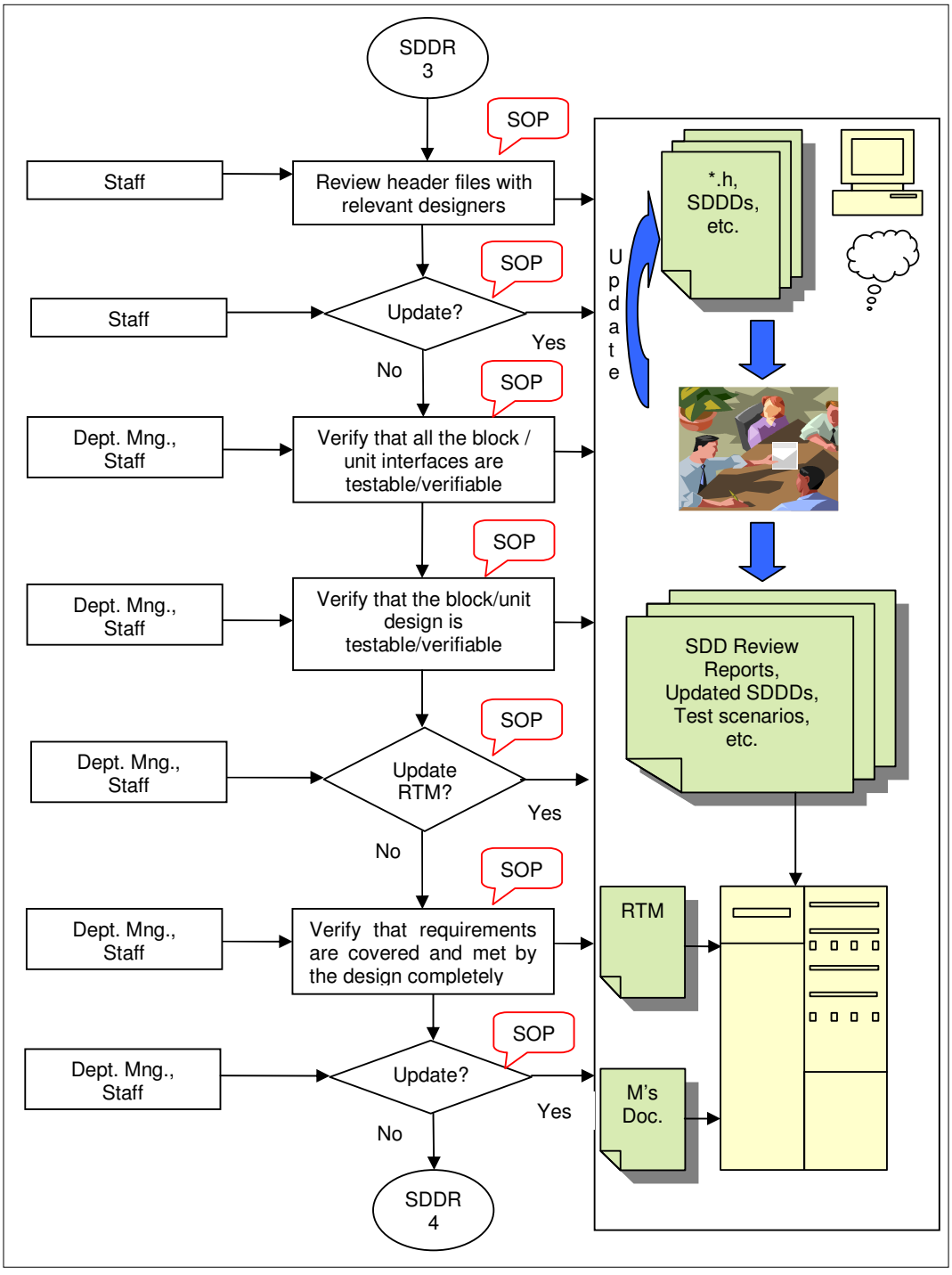

Figure 4-2 (cont'd)

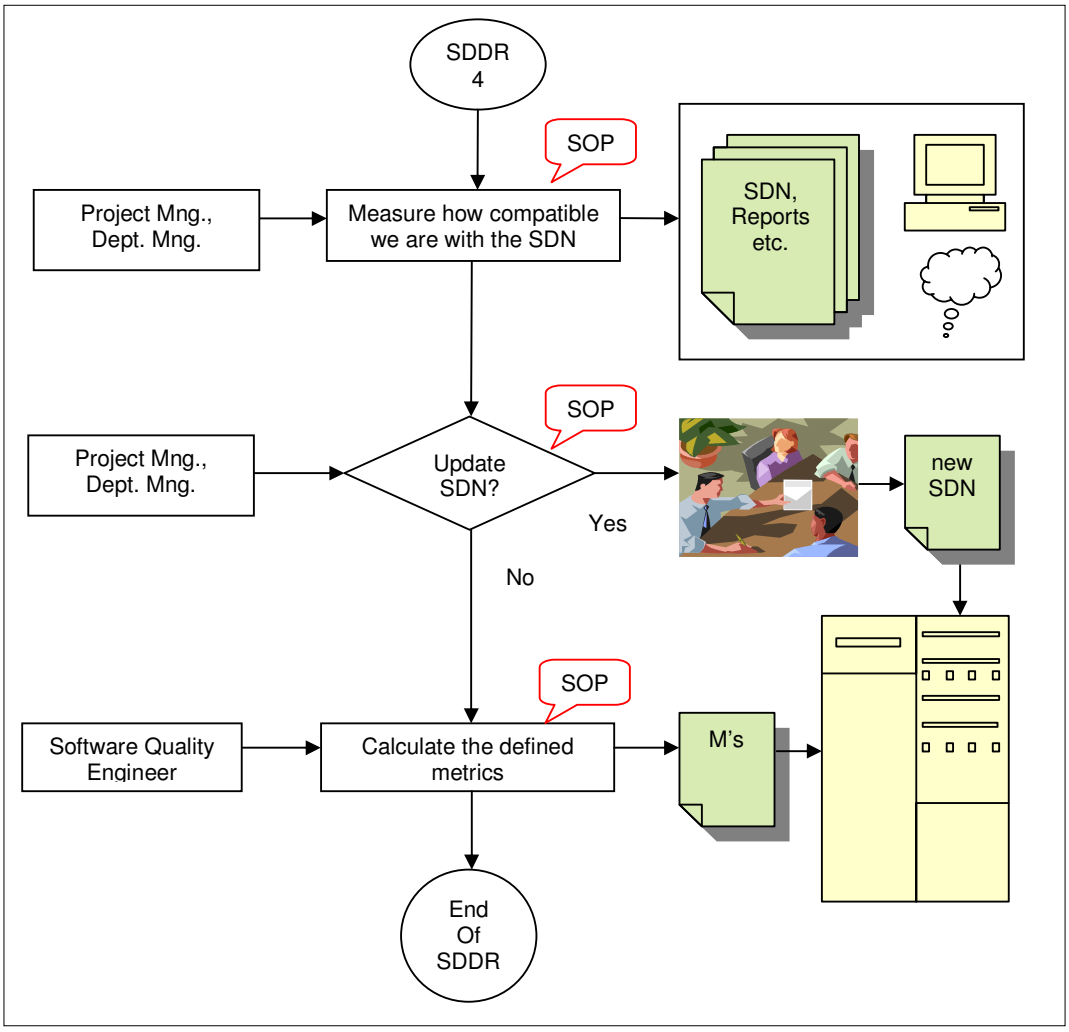

Figure 4-2 (cont'd)

Chapter 5

EVALUATION AND CONCLUSION

5.1 Evaluation

The improvement suggestions given in Chapter 4 have been presented to the experienced staff (with a mean experience of 9,5 years in house). The staff were asked to evaluate the items according to the questionnaire given in Appendix C. A total of 7 senior design engineers, team supervisors have contributed to this evaluation. The responses are encouraging about the acceptance of the suggestions by the staff.

The improvement suggestions can be gathered in two main groups: Suggestions regarding use of documents and documentation. There are 7 such suggestions. Use of a design template, a design review guideline, or a RTM are examples of these group. Second group of suggestions are for the “to-do” part of the processes. There are 14 such suggestions in this group. Explicit statement of a design decision, verification of a design or a design decision are a few examples for this group. From this main group of suggestions, a total of 35 and 45 suggestions are derived for SAD/SADR, and SDD/SDDR processes respectively.

24* out of the 35 suggestions for the SAD and SADR processes, and 44** out of the 45 suggestions for the SDD and SDD Review (SDDR) processes are considered as necessary.

---

[*] 16/24 unanimously, 8/24 with majority

[**] 21/44 unanimously, 23/44 with majority

19* out of 24 of the suggestions for the SAD and SADR processes, and 35** out of the 45 of the suggestions for the SDD and SDDR processes, which have been indicated as necessary are also found to be applicable.

Almost all of the engineers who do not agree with an item stated that they disagree because they lack both theoretical and practical knowledge about the relevant item. Some of the disagreements are based on lack of human resources. For example, some of the staff who thinks measurements are necessary but cannot be applied states that we do not have enough people to do this work.

The necessity and applicability measurements are recalculated after the staff is informed about measurements and metrics. This time we see that 34 out of 35 suggestions for the SAD and SADR processes are considered as necessary and 25 of them are noted as applicable. Similarly, for the SDD and SDDR processes, all of the suggestions are considered as necessary and 39 out of 45 suggestions are noted as applicable.

The suggestions given below are considered as the most important among others by the staff:

1. State the design decisions explicitly and document these decisions
2. Review and verify the designs, and document these activities
3. Consider all roles carried out by the staff while making up the SDN
4. Use and update a RTM

---

[*] 7/19 unanimous, 12/19 with majority

[**] 12/35 unanimous, 23/35 with majority

Although there was no chance to apply the new process completely and assess the outcomes, partial implementation of some of the suggestions has been possible. Design reviews and documentation of these activities, preparation of meeting reports, use of a RTM are some of the suggested steps that have found a chance of application by the time this thesis was being brought to completion. The first impressions gathered from the staff about the outcome of these activities are very positive. The new model is expected to overcome most of the major problems encountered in the SED. A significant decrease in the number of all types of defects dependent on software design is the first expected improvement. Decrease in overdue deadlines, reduction of costs, ease of maintainability of the software developed, and increased robustness and quality are the other expected improvements to follow if the new process model is applied completely.

The improved process model is thought to bring a better consideration of requirements by introducing the RTM and by emphasizing update of the RTM as needed during the course of project.

Multiple roles carried out by an individual can directly effect the staff's performance and hence the schedule. This point is emphasized and urged to be considered while making up the plan to help to make the plan more accurate.

Furthermore, SDN is introduced as an input to every stage in order to follow how far a team is from the plan and take action if necessary. The problems based on the project planning and implementation of the plan are tried to be addressed by these suggestions.

However, there are still uncovered concerns about the plan such as change of agenda. Of course it is the best practices to keep up with the plan and not to change the agenda. But, if it is unavoidable, the author suggests at least defining a stop condition such that bringing the work to a state that will be easy to restart and take necessary notes about the point the work is left. It is for sure that this interruption will cost some time but this way, may

be, staff can leave the work at a proper state and avoid extra lost of time by trying to remember what they were doing, what was the problem, etc.

Other uncovered points that affect the whole life cycle are interruption of work because of demonstration purposes. Since this is out of concern of any software process, the author suggests supplying enough hardware resources to avoid this situation.

5.2 Conclusion

In this thesis study it is seen that it is possible to predict the quality of a software process before it is put into practice using Güceğlioğlu's model which is based on ISO/IEC 9126. The verification of this prediction was not possible by the time this thesis is completed.

The model can first be used to identify the discrepancies in the current process using "Functional Adequacy" metric along with the "Functional Completeness" and "Existence in Documents" metrics. If the discrepancies are not identified prior to measurements, one can misinterpret the measured values. For example, consider the measured values of the unstructured complexity and coupling metrics for the As-Is and To-Be SDDR process models. The unstructured complexity value of the As-Is process is close to the ideal value. Similarly, the coupling value measured for the As-Is process is better then the value for To-Be model. These are both a result of equally weighing all the activities and functional inadequacy of the As-Is process.

It will be better to calculate the complexity and coupling values for a process where either the process is functionally adequate and complete totally (measured values are close to 1), or for only activities which are functionally adequate and complete by giving a new definition for these metrics.

However, it is also discovered that, there is a need for weighing the activities in the process according to their contribution or influence. Without this adjustment, assessment of the current process or the improved one may

result in wrong conclusions. How to weigh the activities is beyond the scope of this thesis and needs further study both theoretically and practically.

The staff is explicitly considered neither in Güceğlioğlu's study nor in ISO/IEC 9126. However, staff has a direct influence on the process and product quality. No matter how complete and well defined the software process is, if the staff is not competent enough to apply software development and process practices, the process, and hence the product quality will be poor.

The author thinks, most of the people carrying the "software engineer" title lack the proficiency required to fulfill the tasks associated with this title. Usually, the external reasons such as management pressure, project deadline, process implementation practices and similar are given as excuses for the incompetence and these are right to a point. But there are also internal reasons for this incompetence such as lack of knowledge about the principles of software engineering. This is usually due to insufficient training both theoretically and practically on the subject.

In all engineering disciplines, a student is first trained about the fundamentals of the engineering and given all the supporting fundamental information that will be needed like mathematics, physics, etc. When we look at the software industry from human resources point of view, are the knowledge base and fundamental skills of people developing software, enough to undertake the responsibilities/tasks they are given?

The author believes that the arguments on why we can not achieve the quality desired in the software products we develop miss this very fundamental point. Staff should be the very starting point for every software process improvement attempt.

When one investigates the capabilities and work habit of staff from this point, it is easy to note the weaknesses about fundamentals of software engineering. For example, reviews or inspections are not held, documentation is not complete or insufficient, and configuration and version management is not done properly, etc.

A close look at the problems at the SED supports the above opinion as well. The reviews were missing, documentation was poor, and there was misuse of CM/VM tools, etc. Responses gathered from the staff support the above theory as well. For example, some of the engineers disagreed with the improvement items pointing to the need for verifying design testability. Almost all of the engineers who did not agree with this item also disagreed with the measurement and metrics items. After reviewing the questionnaire results, the respondents stated that they disagree because they lack both theoretical and practical knowledge about these fundamental software engineering practices.

The above examples and discussions imply that, an organization should fit the standards to its needs. Since an organization cannot exchange its staff with a staff capable of implementing better software process practices it has to adopt the software processes and process improvement studies to itself.

The author recommends training of recruited staff, first about the basics of software engineering, and then about the organization's internal process model. Usually it is more difficult to modify established opinions and behavior patterns rather than forming them initially with a new recruit.

As a result, the staff should be considered in these models from both fundamentals and experience point of view. A new category named "Staff Competence" can be added to the Güceğlioğlu's model for this purpose.

Management commitment is one of the musts for any process improvement attempt to be successful. However, the four basic suggestions and their related counterparts introduced in the previous chapter seem to be easily applicable without too much investment. Acceptance of the model by the staff is also worth noting here. These are very encouraging from the management point of view.

A process resembles a chain. It cannot be stronger than the weakest link. This thesis study only covers the design and design verification parts of

a software development process. However, a SPI program should handle all parts of a development process.

Organizational structure is another aspect to consider to have SPI attempt be successful. The findings of this thesis study also support this comment. The design engineers are not willing to carry out other roles such as test engineer, quality engineer, etc. Also, effects of multiple roles on the SDN cannot be taken into account accurately.

Furthermore, the relations between software process improvement and the target environment (i.e. real-time embedded) for which the software is being developed may be worth further investigation. Although it is expected that main steps shall apply independently from the target environment, the implementation of these steps can differ according to various characteristics of the system being developed.

REFERENCES

[1] Forrest Shull, Carolyn Seaman, and Marvin Zelkowitz, “Victor R. Basili's Contributions to Software Quality”, IEEE Software Jnl, pp. 16-18, Jun 2006.

[2] Victor R. Basili, “Software Development: A Paradigm for the Future”, IEEE Software Jnl, pp. 471-485, Sep 1989.

[3] M. Hayes, “Precious Connection”, InformationWeek, pp. 34-50, Oct 2003.

[4] P. Thibodeau and L. Rosencrance, “Users Losing Billions Due to Bugs”, Computerworld, Vol. 36, no. 27, pp. 1-2, 2002.

[5] V. R. Basili, "Data Collection, Validation, and Analysis", in Tutorial on Models and Metrics for Software Management and Engineering, IEEE Catalog No. EHO-167-7, pp. 310-313, 1981.

[6] Ivan Aaen, “Software Process Improvement: Blueprints Versus Recipes”, IEEE Software Jnl, pp. 86 – 93, Sep/Oct 2003.

[7] CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI – SE/SW/IPPD/SS) Version 1.1, Software Engineering Institute, Carnegie Mellon University, 2002.

[8] Ilmasri Saastamoinen, and Markku Tukiainen, “SPI in Small and Medium Sized Software Enterprises in Eastern Finland: A State-of-the-Practice Study”, SPI 11th European Conference Proceedings, pp. 69 – 78, Aug 2004.

[9] ISO/IEC IS 15504 – 2: Software Engineering – Process Assesment – Part 2: Performing An Assesment, 2003.

[10] P. Kuvaja, J. Simila, L. Krzanik, A. Bicego, G. Saukkonen, “Software Process Assessment and Improvement”, The BOOTSTRAP Approach. Blackwell Publishers, 1994.

[11] ISO/IEC TR 9126-3: Software Engineering – Software Product Quality – Part 3: Internal Metrcis, 2000.

[12] Felix Garcia, Francisco Ruiz, and Mario Piattini, “An Experimental Replica to Validate a Set of Metrics for Software Process Models”, SPI 11th European Conference Proceedings, pp. 69 – 78, Aug 2004.

[13] Software Development Process Document At SED, X Company.

[14] Selçuk Güceğlioğlu, “A Pre-Enactment Model for Measuring Process Quality”, Ph. D Thesis, Middle East Technical Univ., Informatics Institute, 2006.

[15] David F. Rico, “ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers”, J. Ross Publishing, 2004.

[16] Dean Leffingwell, and Don Widrig, “Managing Software Requirements” foreword by Ed Yourdon, 2nd Edition, Addison-Wesley Publ., 2003.

[17] Roger S. Pressman, “Software Engineering: A Practitoner’s Approach”, 4th Edition, McGraw Hill Inter. Editions, 1997.

[18] Davis, M.J., “Process and Product: Dichotomy or Duality”, Software Engineering Notes, ACM Press, Vol. 20, No. 2, pp. 17-18, Apr 1995.

[19] Lars Mathiassen, Ojelanki K. Ngwenyama, and Ivan Aaen, “Managing change in software process improvement”, IEEE Software Jnl, pp. 84-91, Nov/Dec 2005.

[20] Reidar Conradi, Alfonso Fuggetta, “Improving software process improvement” , IEEE Software Jnl, pp. 92-99, Jul/Aug 2002.

[21] Ed Yourdon, “Software Metrics”, Application Development Strategies, Nov 1994.

[22] S. Brady, and Tom DeMarco, “Management-Aided Software Engineering”, IEEE Software, Vol. 11, No. 6, pp. 25-32, Nov 1994.

[23] Lawrence H. Putnam, and, Ware Myers, “Executive Briefing: Controlling Software Development”, IEEE Software Society Press, 1996

[24] Watts S. Humphrey, “PSP: A Self-Improvement Process for Software Engineers”, SEI Series in Software Engineering, Addison-Wesley, 2005

[25] Systems and Software Consortium, RTCA DO-178B, http://www.software.org/quagmire/descriptions/rtcado-178b.asp, April 2006

[26] NASA, Process Based Mission Assurance (PBMA) Knowledge Management System (KMS), Software Design Verification and Testing, http://pbma.hq.nasa.gov/framework_content_cid_458, April 2006

[27] QA Systems Inc., Concepts – ISO 9126, http://www.qa-systems.com/concepts/iso9126.html, April 2006

[28] H. Seçkin, “Software Process Improvement Based On Static Process Evaluation”, M.S. Thesis, Middle East Technical Univ., Elec. and Electronics Eng., 2006

APPENDIX A

REVIEW ACTIVITIES
AND
METRICS

In this section, first the activities of As-Is SAD Review and SDD Review processes are given respectively. Then, the metrics are defined for each activity.

The activities along with this metrics given in the following two sections form the basis for the measurements given in Section 3.2 Measurements.

There are 17 activities in the SADR process, and, 19 activities in the SDDR process.

A.1 Review Activities

A.1.1 SAD Review Activities

Activities of the As-Is SAD Review process are given in Table A-1-1.

Table A-1-1 SAD Review Activities

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
1	Send SAD for review	The SAD is sent to all departments for review by project manager.	Project Manager, Department Managers	e-mail/ word document
2	Review SAD	The SAD is reviewed by software engineering staff at each department. If there are any points of concerns, objections, unclear parts it is noted. If necessary, staff conducts research.	Department Managers, Software Engineers	Blackboard/white paper/word document/notebook/ conversation/ internet
3	Arrange internal department review meeting	The department staff is called to attend internal SAD review meeting.	Department Manager	e-mail/ Conversation/ phone
4	Internal department review meeting	If there are any points of concerns, objections, unclear parts it is pointed and discussed first in the department. If concerns cannot be clarified internally or these points need participation of other departments' staff a meeting will be requested.	Department Manager, Software Engineers	Blackboard/ white paper/ word document/ notebook/ conversation

Table A-1-1 (cont'd)

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
5	If necessary, arrange review meeting with other departments	Prior to a broader meeting with all departments, the relevant departments with points of concerns, objections request a meeting with limited participation.	Department Manager	e-mail/ conversation/ phone
6	If necessary, discuss SAD with limited participation	The points of concerns, objections, or unclear parts are discussed by software engineering staff from relevant departments only. The points of concerns, objections, or unclear parts are tried to be clarified.	Department Managers, Software Engineers	Blackboard/ white paper/ word document/ notebook/ conversation
7	Collect/Send feedback about SAD	Departments are asked to return their opinions about SAD.	Project Manager, Department Managers	e-mail/ Conversation/ phone
8	Determine the outline for the meeting	Determine the outline for the SAD review meeting due to the feedbacks. Prioritize subjects. The SAD document will be totally reviewed but more emphasize will be given to points of concern or conflicts (if exists any).	Project Manager, Department Managers	e-mail/ word document/ phone

Table A-1-1 (cont'd)

No	Activity Name	Activity Definition	Staff	**Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias**
9	Arrange a final review meeting	For a final review of SAD a meeting is arranged by the project manager.	Project Manager	A meeting request via Outlook is sent to each required participant individually or to the department managers generally.
10	Final review of SAD	Review the SAD finally before proceeding to the detailed design stage with participation of all departments. It is make sure that there is no conflict between departments' understanding the SAD.	Project Manager, Department Managers, Software Engineers	Conversation/ blackboard/ word documents/ notebook/ Microsoft Project
11	If there are conflicts, form a work group to resolve the conflicts	If there exists any conflicts that needs further study, form a work group from related departments to resolve the issue.	Project Manager, Department Managers, Software Engineers	Conversation
12	If necessary, arrange meeting to resolve conflicts	Arrange a meeting with limited participation to come over the conflicts and reach a consensus.	Department Managers, Software Engineers	Conversation/ phone

Table A-1-1 (cont'd)

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
13	Meet to resolve conflicts	Work together with other departments' engineers and try to reach a solution. If an agreement can not be made, then inform the project manager and make a decision with project manager.	Department Managers, Software Engineers	Conversation/ blackboard/ word documents/ notebook
14	If necessary, update SAD	Due to the decisions made in the review meeting the architecture is updated if necessary.	Project Manager, Department Managers, Software Engineers	word document/ notebook
15	If necessary, update tasks	Due to the decisions made in the review meeting the tasks are updated.	Project Manager, Department Managers	Conversation/ Microsoft Project/ e-mail
16	If necessary, update task schedule.	Due to the decisions made in the review meeting the schedule is updated.	Project Manager, Department Managers	Conversation/ Microsoft Project/ e-mail
17	Release new SAD	Updated SAD is released.	Project Manager	e-mail/ word document

A.1.2 SDD Review Activities

Activities of the As-Is SDD Review process are given in Table A-1-2.

Table A-1-2 SDD Review Activities

No	Activity Name	Activity Definition	Staff	**Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias**
1	Send BDD for review	The BDD is sent to all relevant staff for review.	Department Manager	e-mail/ word document
2	Review BDD	The BDD is reviewed by software engineering staff. If there are any points of concerns, objections, unclear parts it is noted.	Department Manager, Software Engineers	Blackboard/ white paper/ word document/ notebook/ conversation/ CASE Tools
3	Collect/Send feedback about BDD	Either the department manager requests feedbacks about BDD or the staff sends or tells any concerns regarding BDD, usually informally.	Department Manager, Software Engineers	Conversation/ phone/ e-mail
4	If necessary, arrange a meeting to review BDD together	If there are concerns, objections, unclear parts regarding BDD a meeting is arranged with relevant staff. If needed, staff from other departments can be requested to attend the meeting. Such meetings are usually informal.	Department Manager, Software Engineers, Project Manager	Conversation/ e-mail/ phone

Table A-1-2 (cont'd)

No	Activity Name	Activity Definition	Staff	Forms/ Documents/ Archival Records/ Tools/ Applications/ Other Medias
5	If necessary, review BDD with limited participation	Points of concerns, objections, or unclear parts are discussed tried to get resolved. If such worries can not be resolved or understood project manager is conducted and process is repeated from step 4.	Department Manager, Software Engineers, Project Manager	Blackboard/ white paper/ word document/ notebook/ conversation
6	If necessary, update BDD	If any changes are decided to make to the block design, the BDD is updated. Then, BDD is placed to the backed-up network storage area or checked-in to the version management tool.	Software Engineers	word document/ CASE Tools
7	If necessary, release new BDD	New BDD is released.	Software Engineers	word document/ CASE Tools
8	Send BIDs for review to all relevant staff	The BIDs are sent to all relevant staff for review including other departments. Usually the software engineer who puts the interface into a document form sends the BID.	Software Engineers	e-mail/ word document
9	Review BIDs	The BIDs are reviewed by software engineering staff. If there are any points of concerns, objections, unclear parts it is noted.	Department Manager/ Software Engineers	white paper/ word document/ notebook/ conversation/ CASE Tools

Table A-1-2 (cont'd)

No	Activity Name	Activity Definition	Staff	Forms/ Doc's/ Archival Records/ Tools/ App's/ Other Medias
10	If necessary, arrange BID review meeting with other departments	The relevant department staff is called to attend BID review meeting. Sometimes this call is done by staff engineers, sometimes by the department manager.	Department Manager/ Software Engineers	e-mail/ conversation/ phone
11	If necessary, discuss block interfaces with limited participation	The points of concerns, objections, or unclear parts are discussed by software engineering staff from relevant departments only. The points of concerns, objections, or unclear parts are tried to be clarified. Sometimes department managers also attend these meetings.	Department Managers, Software Engineers	Blackboard/ white paper/ word document/ notebook/ conversation
12	If necessary update BID	If any changes are decided to make to block interfaces the BID is updated. BID is placed to the backed-up network storage area or checked-in to the version management tool.	Software Engineers	word document/ CASE Tools
13	Release new BID	Send new BID or a notification telling that new BID is ready to all relevant staff.	Software Engineers	e-mail/ word document
14	Review UDD	If exists, the unit design document is reviewed by relevant software staff or the person himself. Sometimes the department manager also attends this reviews.	Department Manager, Software Engineers	Blackboard/ white paper/ word document/ notebook/ conversation/ CASE Tools

Table A-1-2 (cont'd)

No	Activity Name	Activity Definition	Staff	Forms/ Doc's/ Archival Records/ Tools/ App's/ Other Medias
15	If necessary update UDD	After conducting a review UDD is updated if necessary. UDD is then stored either in a file in hard or soft copy, or placed in a backed-up network storage area, or checked-in to the version management tool.	Software Engineers	word document/ notebook/ CASE Tools
16	Review UID	If exists, Unit Interface Document is reviewed by the participation of relevant staff.	Software Engineers	e-mail/ word document/ white paper/ conversation/ phone/ CASE Tools
17	If necessary update UID	After conducting a review unit interfaces are updated if necessary. New interface document is then stored either in a backed-up network storage area, or checked-in to the version management tool.	Software Engineers	word document/ CASE Tools
18	Review sequence diagrams	If exists, the sequence diagrams are reviewed either by relevant staff or personally. Sometimes department manager also attends to these reviews.	Department Manager, Software Engineers	conversation/ CASE Tools
19	If necessary update sequence diagrams	After conducting a review sequence diagrams are updated if necessary. Sequence diagrams are then stored either in a backed-up network storage area, or checked-in to the version management tool.	Software Engineers	CASE Tools

A.2 Metrics calculated according to the Activities

Metrics are calculated using the model presented in Selçuk Güceğlioğlu's study [14].

A.2.1 SAD Review Process Metrics

Metrics applied to the SAD Review Process are given in Table A-2-1 through Table A-2-5:

Table A-2-1 SAD Review Process Metrics *(from 1 to 3)*

Activity Number	Complexity (1)	Coupling(2)	Failure Avoidance (3)
1, 3, 5, 7, 9, 11, 12, 17	No decision	No interaction	No review, inspection, checkpoint or similar techniques
2	Semi-structured decision. Department manager and software engineers analyzes requirements and SAD. Then make judgment about whether the design is appropriate or not. there is no rule or document followed which determines/guides "when a request is approved" or "under which conditions a request is refused". There are sometimes complex decisions.	Interaction with other software blocks	The SAD is reviewed. But there is no formal method followed to avoid failure. Failure avoidance is dependent to the experience and talent of staff.

Table A-2-1 (cont'd)

Activity Number	Complexity (1)	Coupling(2)	Failure Avoidance (3)
4	Unstructured decision. Department manager and software engineers state their concerns, objections, or point out unclear parts. If these points can not be clarified at this meeting a broader meeting is planned with all relevant departments only. There are sometimes complex decisions.	Interaction with other software blocks	The SAD is reviewed. But there is no formal method followed to avoid failure. Failure avoidance is dependent to the experience and talent of staff.
6	Unstructured decision. Points of concerns, objections, or unclear points are discussed. Then a judgment is made whether these concerns are appropriate or not. If these points can not be clarified at this meeting usually project manager is included in the process and he tailors a decision. There are sometimes complex decisions.	Interaction with other software blocks	The SAD is reviewed. But there is no formal method followed to avoid failure. Failure avoidance is dependent to the experience and talent of staff.
8	Simple decision as what to discuss at the final review meeting	No interaction	Not applicable
10	Semi-structured. Points of concerns, objections, or unclear points are discussed. A substantial agreement is tried to be reached. A judgment is made whether these concerns are appropriate or not. If these points can not be clarified at this meeting usually project manager is included in the process and he tailors a decision. There are sometimes complex decisions.	Interaction with all software and may be hardware blocks	The SAD is reviewed with all the relevant staff. But there is no formal method followed to avoid failure. Mostly failure avoidance is dependent to the experience and talent of staff.

Table A-2-1 (cont'd)

Activity Number	Complexity (1)	Coupling(2)	Failure Avoidance (3)
13	Unstructured. Points of concerns, objections, or unclear points are discussed. A substantial agreement is tried to be reached. A judgment is made. There are sometimes complex decisions.	Interaction with all software and may be hardware blocks	The SAD is reviewed with all the relevant staff. But there is no formal method followed to avoid failure. Mostly failure avoidance is dependent to the experience and talent of staff.
14	Very simple decision	No interaction	The updated SAD is only reviewed to avoid typos and missing updates by the staff.
15	Very simple decision	No interaction	Updated tasks are reviewed
16	Very simple decision	No interaction	Updated schedule is reviewed

Table A-2-2 SAD Review Process Metrics *(4 and 5)*

Activity Number	Restorability (4)	Restoration Effectiveness (5)
1	The SAD is sent via e-mail. Both the SAD and e-mail are stored in a backed-up network	Can be restored
2	There is no review report. Review results are not formally recorded. But, people take notes about their sections or some general points are noted	No restoration for the review. Only from notes taken during the review meeting (notebook, word document, etc.)
3, 5, 11	Not recorded	Not applicable
4	There is no review report or minutes of meeting document. Review results are not formally recorded. But, people take notes about their sections or some general points are noted.	No restoration for the review. Only from notes taken during the review meeting (notebook, word document, etc.).
6, 10, 13	There is no review report or minutes of meeting document. Review results are not formally recorded. But, people take notes about their sections or some general points are noted. If a change is made, SAD is updated and stored	No restoration for the review. Only from notes taken during the review meeting (notebook, word document, etc.)
7	The comments are sent via e-mail, or the documents prepared for this purpose are stored in a backed-up network	Can be restored
8	Outline is recorded sometimes as a word document in a PC or it is stored in the backed-up network if it is send via e-mail	Can be restored when IT is used
9	A meeting request is sent via e-mail. Such mails are stored at the backed-up network	Can be restored
12	Sometimes such meeting requests are sent via e-mail but sometimes meetings are organized by phone calls or by face to face conversation. Mails are stored at the backed-up network	Can be restored if e-mail is used
14, 17	The SAD is stored in a backed-up network	Can be restored
15, 16	The tasks and The schedule are present in the Microsoft Project Document which is backed-up	Can be restored

Table A-2-3 SAD Review Process Metrics *(from 6 to 9)*

Activity Number	Functional Adequacy (6)	Functional Completeness (7)	IT Usage (8)	IT Density (9)
1	Not present in the regulatory documents	Not applicable	IT used	SAD is stored in backed-up network as a word document
2, 4, 6	Adequate	Complete	No IT usage	No forms, documents, archival records or other similar documents that are prepared, updated, deleted or searched
3	Not present in the regulatory documents	Not applicable	Limited IT usage	No forms, documents, archival records or other similar documents that are prepared, updated, deleted or searched
5	Adequate	Complete	Limited IT usage	No forms, documents, archival records or other similar documents that are prepared, updated, deleted or searched
7	Not present in the regulatory documents	Not applicable	IT used	Low
8	Not present in the regulatory documents	Not applicable	Limited IT usage	Outline is recorded sometimes as a word document or it is stored in the backed-up network if it is send via e-mail

Table A-2-3 (cont'd)

Activity Number	Functional Adequacy (6)	Functional Completeness (7)	IT Usage (8)	IT Density (9)
9, 12	Not present in the regulatory documents	Not applicable	Limited IT usage	Meeting requests are sent via e-mail
10, 11, 13	Not present in the regulatory documents	Not applicable	No IT usage	No forms, documents, archival records or other similar documents that are prepared, updated, deleted or searched
14, 17	Adequate	Complete	IT used	SAD is stored in backed-up network as a word document
15	Not present in the regulatory documents	Not applicable	IT used	The tasks are present in the Microsoft Project Document which is backed-up
16	Not present in the regulatory documents	Not applicable	IT used	The schedule are present in the Microsoft Project Document which is backed-up

Table A-2-4 SAD Review Process Metrics *(from 10 to 13)*

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
1 – 13	No specific accuracy requirement	No interaction	There is no access to data	No difficulties or misunderstandings
14	SAD should be correct and meet the system requirements	No interaction	There is no access to data	No difficulties or misunderstandings
15, 16	No specific accuracy requirement	No interaction	Only staff with appropriate rights can open project document	No difficulties or misunderstandings
17	No specific accuracy requirement	No interaction	Only staff with appropriate rights can open SAD document stored in the network	No difficulties or misunderstandings

Table A-2-5 SAD Review Process Metrics *(from 14 to 17)*

Activity Number	Existence in Documentation (14)	Input Validity Checking (15)	Undoability (16)	Attractive Interaction (17)
1	Not described	Not applicable	Recorded, document can be withdrawn by the project manager	Attractive interaction in preparing SAD document
2, 4, 6, 8, 10, 11, 13	Not described	Not applicable	Not recorded formally	No attractive interaction
3, 5, 9, 12	Not described	Not applicable	Meeting can be re-scheduled or cancelled completely	No attractive interaction
7	Not described	Not applicable	Only comments sent via e-mail can be withdrawn by the sender. Not applicable to other types of feedback	No attractive interaction
14	Not described	Not applicable	SAD is recorded in the backed-up network	No attractive interaction
15	Not described	Not applicable	The tasks are present in the Microsoft Project Document which is backed-up	No attractive interaction
16	Not described	Not applicable	The schedule is present in the Microsoft Project Document which is backed-up	No attractive interaction
17	Not described	Not applicable	Updated SAD is stored in the backed-up network	No attractive interaction

A.2.2 SDD Review Process Metrics

Metrics are calculated using the model presented by Güceğlioğlu's study [14].

Metrics for the SDDR Process are given in Table A-2-6 through Table A-2-10:

Table A-2-6 SDD Review Process Metrics *(from 1 to 3)*

Activity Number	**Complexity (1)**	**Coupling (2)**	**Failure Avoidance (3)**
1, 3, 4, 7, 8, 10, 13	No decision	No interaction	No review, inspection, checkpoint or similar techniques
2, 9, 14, 16, 18	Semi-structured decision while reviewing the design/interface documents. Department manager and software engineers analyzes requirements and design/interface documents. Then make judgement about whether the design and interfaces are appropriate or not. There is no rule or template to follow. There are sometimes complex decisions.	Interaction with other software blocks	The design/interface documents are reviewed But there is no formal method followed to avoid failure. Mostly failure avoidance is dependent to the experience and talent of staff.

Table A-2-6 (cont'd)

Activity Number	Complexity (1)	Coupling (2)	Failure Avoidance (3)
5 – BDD 11 – BID	Unstructured decision while reviewing the BDD/BID. Points of concerns, objections, or unclear points are discussed. Then a judgment is made whether these concerns are appropriate or not. A substantial agreement is tried to be reached. If these points can not be clarified at this meeting usually project manager is included in the process and he tailors a decision. There are sometimes complex decisions.	Interaction with other software blocks	The BDD/BID is reviewed with all the relevant staff. But there is no formal method followed to avoid failure. Mostly failure avoidance is dependent to the experience and talent of staff.
6, 12, 15, 17, 19	Very simple decision	No interaction	The updated document is only reviewed to avoid typos and missing updates by the staff.

Table A-2-7 SDD Review Process Metrics *(4 and 5)*

Activity Number	Restorability (4)	Restoration Effectiveness (5)
1, 8	The BDD/BID is sent via e-mail. The BDD/BID and e-mails are stored in a backed-up network	Can be restored
2, 5, 9, 11, 14, 16, 18	There is no review report. Review results are not formally recorded. But, people take notes about their sections or some general points are noted	No restoration for the review. Only from notes taken during the review meeting (notebook, word document, etc.)
3	Sometimes e-mail is used to send feedback. E-mails are stored in a backed-up network.	If send by e-mail can be restored.
4, 10	Not recorded	Not applicable
6, 7, 12, 13, 15, 17, 19	Updated/New documents are stored in the backed-up network	Can be restored

Table A-2-8 SDD Review Process Metrics *(from 6 to 9)*

Activity Number	Functional Adequacy (6)	Functional Completeness (7)	IT Usage (8)	IT Density (9)
1, 6, 7, 8, 13, 15, 17, 19	Not present in the regulatory documents	Not applicable	IT used	Documents are stored both in backed-up network and word document form
2, 5, 14, 16, 18	Not present in the regulatory documents	Not applicable	No IT usage	No forms, documents, archival records or other similar documents that are prepared, updated, deleted or searched
3, 4, 10	Not present in the regulatory documents	Not applicable	Limited IT usage	No forms, documents, archival records or other similar documents that are prepared, updated, deleted or searched
9, 11	Adequate	Complete	No IT usage	No forms, documents, archival records or other similar documents that are prepared, updated, deleted or searched
12	Adequate	Complete	IT used	Documents are stored both in backed-up network and word document form

Table A-2-9 SDD Review Process Metrics *(from 10 to 13)*

Activity Number	Computational Accuracy (10)	Data Exchangeability (11)	Access Auditability (12)	Functional Understandability (13)
1, 2, 3, 4, 5, 8, 9, 10, 11, 14, 16, 18	No specific accuracy requirement	No interaction	There is no access to data	No difficulties or misunderstandings
6, 12, 15, 17, 19	Design documents should be correct and meet the system requirements	No interaction	There is no access to data	No difficulties or misunderstandings
15, 16	No specific accuracy requirement	No interaction	Only staff with appropriate rights can open project document	No difficulties or misunderstandings
7, 13	No specific accuracy requirement	No interaction	Only staff with appropriate rights can open SAD document stored in the network	No difficulties or misunderstandings

Table A-2-10 SDD Review Process Metrics *(from 14 to 17)*

Activity Number	Completeness of Documentation (14)	Input Validity Checking (15)	Undoability (16)	Attractive Interaction (17)
1, 8	Not described	Not applicable	Recorded, document can be withdrawn by the project manager	Attractive interaction in preparing SAD document
2, 3, 5, 9, 11, 14, 16, 18	Not described	Not applicable	Not recorded formally	No attractive interaction
4, 10	Not described	Not applicable	Meeting can be re-scheduled or cancelled completely	No attractive interaction
6, 12, 15, 17, 19	Not described	Not applicable	SAD is recorded in the backed-up network	No attractive interaction
7, 13	Not described	Not applicable	Updated documents are stored in the backed-up network	No attractive interaction

APPENDIX B

SAMPLE RTM

In this section, a sample RTM format is presented that is being used at SED. SRNo denotes the sytem requirement number. SwRNo denotes the software requirement number. If a software requirement is dependent to another software and/or system requirement, the depended requirement(s) is given in the “Dep. To. Req” column.

Table B-1 Sample RTM

SR No	System Requirement	SwR No	Software Requirement	Dep. To Req.	Met at Block	Met at Unit
1.		**1.1**				
		1.2				
		1.3				
2.		**2.1**				
		2.2				
3.		**3.1**		SwRNo: **2.1** - **2.2**		
4.		**4.1**		-		
5.		**5.1**		SRNo: **2**		
6.		**6.1**		SwRNo: **2.1** - **2.2**		

APPENDIX C

SAMPLE EVALUATION QUESTIONNAIRE

The author has prepared a questionnaire to be sure that all the improvement suggestions are evaluated by the staff.

The original questionnaire distributed to the staff for evaluation consists of four sections. There was one section for each phase of the design process: SAD, SADR, SDD, and SDDR. But here, since the questionnaire is very straightforward, only the SAD section of the questionnaire is presented partially.

The staff was asked to select one or more out of the three qualifications, “Necessary”, “Applicable”, “Don’t Agree” by just indicating a check. They are also encouraged to add other comments if they wish. Explanations of the fields and the sample questionnaire are given in Table C-1 and Table C-2 respectively:

Table C-1 Questionnaire Field Descriptions

Field	Description
Necessary	Check this field if you think the suggested item is necessary.
Applicable	Check this field if you think the suggested item is applicable in terms of availability of necessary information, manpower, time or other resources.
Don’t Agree	Check this field if you don’t agree with the suggested item.
Other	Especially if the expected cost or overhead to be brought about by this item is considered significant, it should be indicated. Also, any expected quantitative advantages, such as work hours saved, error percentages reduced, etc. may be noted here.

Table C-2 SAD Improved Process Evaluation Questionnaire

Step	Original	Suggested	Comments by the Staff			
			Necessary	Applicable	Don't Agree	Other
Inputs	1. SRS 2. Hardware Architecture Plan.	No change	N/A	N/A	N/A	N/A
	Does not exist	*3. If exists, Design Template*				
		4. SDN				
		5. Measurement Guideline				
		6. Metrics Document				
To Do	1. Requirements are analyzed. 2. The software blocks required for the equipment/system, their functions and place on the hardware are determined.	No change	N/A	N/A	N/A	N/A
	Does not exist	*3. State the design decisions explicitly and document them.*				
		4. Form/Update Requirements Traceability Matrix (RTM) to represent which requirement is met at which block.				