DESIGN AND IMPLEMENTATION OF A SOFTWARE DEVELOPMENT PROCESS MEASUREMENT SYSTEM

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

BY

ÖZGÜR ERALP

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

JANUARY 2004

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. Mübeccel Demirekler
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 Bilgen
Supervisor

Examining Committee Members

Prof. Dr. Uğur Halıcı _____

Prof. Dr. Semih Bilgen _____

Assoc. Prof. Dr. Onur Demirörs _____

Asst. Prof. Dr. Cüneyt Bazlamaçcı _____

Levent Alkışlar (Ms.) _____

ABSTRACT

DESIGN AND IMPLEMENTATION OF A SOFTWARE DEVELOPMENT PROCESS MEASUREMENT SYSTEM

ERALP, Özgür

MSc. , Department of Electrical and Electronic Engineering

Supervisor: Prof. Dr. Semih BİLGEN

January 2004, 142 pages

This thesis study presents a software measurement program. The literature on software measurement is reviewed. Conditions for an effective implementation are investigated. A specific measurement system is designed and implemented in ASELSAN, Inc. This has involved organizational as well as technical work. A software tool has been developed to assist in aggregating measurements obtained from various CASE tools in use. Results of the implementation have started to be achieved. Lots of useful feedbacks have been returned to the organization as a result of analyzing of the measurement data.

Keywords: Software Measurement, Software Metric, PSM, GQM

ÖZ

YAZILIM GELİŞTİRME SÜRECİ İÇİN BİR ÖLÇÜM SİSTEMİ TASARIMI VE GERÇEKLEŞTİRİLMESİ

ERALP, Özgür

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Semih BİLGEN

Ocak 2004, 142 sayfa

Bu tez çalışması, bir yazılım ölçüm programını sunmaktadır. Yazılım ölçümü ile ilgili literatür incelenmiş, ve etkili bir uygulama için şartlar araştırılmıştır. ASELSAN AŞ özelinde bir ölçüm sistemi tasarlanmış ve organizasyonda uygulanmıştır. Bu, hem organizasyonel hem de teknik çalışmayı içermektedir. Kullanımdaki çeşitli CASE araçlarından elde edilen ölçüm verilerinin analizini kolaylaştırmak amacı ile bir yazılım aracı geliştirilmiştir. Uygulanan ölçüm programının sonuçlarına erişilmeye başlanmıştır. Verilerin analiz edilmesiyle, organizasyona birçok yararlı bilgi geri dönüşü gerçekleşmektedir.

Anahtar Kelimeler: Yazılım Ölçüm, Metrik, PSM, GQM

ACKNOWLEDGEMENTS

I would like to thank the following people:

- Prof. Dr. Semih Bilgen, for his help, professional advice and valuable supervision during the development and the improvement stages of this thesis. This thesis would not be completed without his guidance and support.
- The members of PAT-G team in MST Division of ASELSAN Inc. that are Levent Alkışlar, Ayşın Zaim, Özgü Özköse Erdoğan, Güliz Aykut, Aydan Doğru, Zühre Yılmazer, for their contributions on this study.
- My parents, Avni and Semahat Eralp; my brother, Arda Eralp, for their great encouragement and continuous morale support.

TABLE OF CONTENT

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS AND ACRONYMS

ASELSAN	Military Electronics Industry in Turkey
CASE	Computer Aided Software Engineering
CMM	Capability Maturity Model
COTS	Commercial Off-The-Shelf
DSP	Digital Signal Processing
GQM	Goal Question Metric
GUI	Graphical User Interface
I-C-M	Issue - Category - Measure
IEC	International Electrotechnical Commission
ISO	International Organizational for Standardization
KALDER	Turkish Quality Association
LOC	Line of Code
MST	Microwave and System Technologies Division in ASELSAN Inc.
NASA	National Aeronautics and Space Administration
PAT-G	One of Process Action Teams in YİE, called G
PSM	Practical Software Measurement
QA	Quality Assurance

QIP	Quality Improvement Paradigm
SCU	Software Configuration Unit
SDD	Software Design Document
SDP	Software Development Plan
SEL	Software Engineering Laboratory
SIDD	Software Interface Design Document
SLOC	Source Lines of Code
SRS	Software Requirement Specification
WBS	Work Breakdown Structure
YİE	Software Process Group in ASELSAN Inc.
YMM	Software Engineering Department in ASELSAN Inc.
YT	Software Test Department in ASELSAN Inc.

CHAPTER 1

INTRODUCTION

1.1. Measurement Process

Software measurement plays an important role in whole software development activities. Paul Goodman, writer of Practical Implementation of Software Metrics, claims that the role of software metrics is to enable engineers and managers to survive in today's business environment [9]. Measures that are obtained as a result of measurement are the numbers used to create the metrics, and the metrics are the numbers turned into information. Managers need a basis for evaluating product quality and analyzing issues or problems, and a foundation for quantitative control of the project management and engineering processes. In addition, measurement provides the insight a manager needs to make decisions critical to project success [5]. Effective measurement programs help and succeed them by enabling to develop achievable plans.

In 1987 Gabriel Pall defined a process as the logical organization of people, materials, energy, equipment, and procedures into work activities designed to produce specified end results [6].

The Figure 1 indicates measurement process life-cycle ([3],[5],[9]). The stages very resemble well-known software development life-cycle steps. This fact should not be surprising, because before starting implementation, analysis and design are fundamental stages in software engineering.

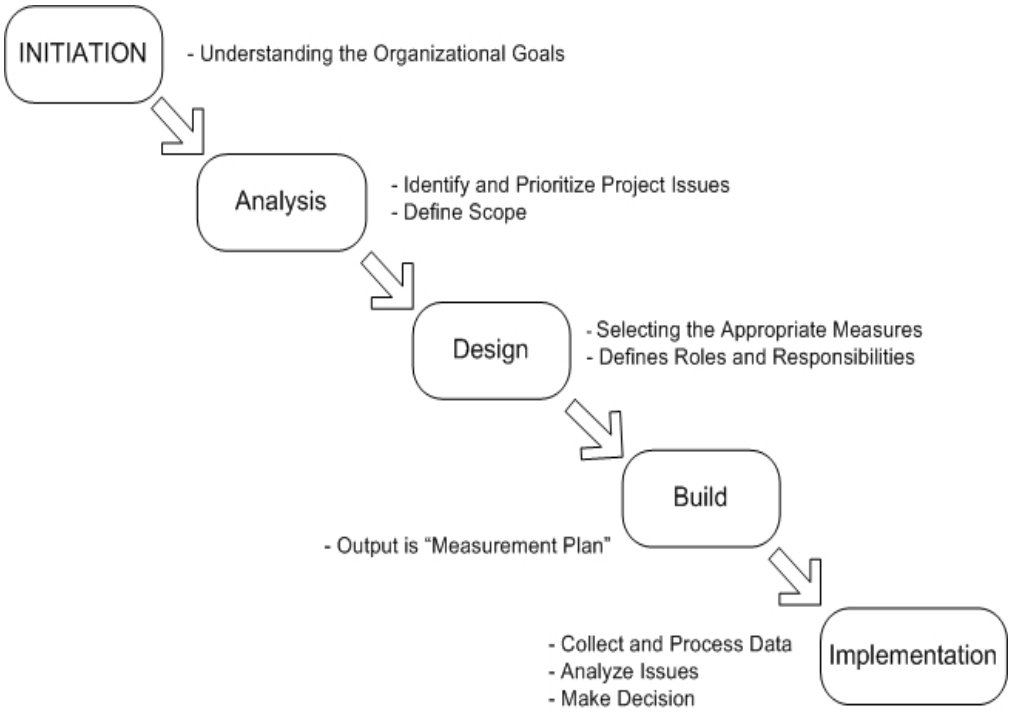


Figure 1 - Measurement Process Life-Cycle

The first stage of measurement process, called "Initiation", is described in Chapter 2 in detail. In this stage, the analyst, who can be a person or group, should understand all organizational goals clearly, since the analyst tailors measurement process in direction of these goals. If the costume is not

suitable, the organization can not put it on. Besides understanding, the analyst should obtain organizational support and required equipments for measurement program must be provided.

According to the KALDER survey in Turkey about difficulties in software measurement, the most encountered difficulties are in data collection process, organizational participation and support. The other difficulties are in analysis of the gathered data and the measurement plan [7].

In the "Analysis" stage, which is described in Chapter 3 in detail, project issues must be identified and prioritized. Also, the measurement scope should be well-defined according to project(s), and phase(s) of software life-cycle. Not excess, only adequate measures should be implemented to address those issues.

In the "Design" stage, which is described in Chapter 4 in detail, selecting appropriate measures is very critical for the measurement process. Only required and applicable measures should be implemented based on the issues and objectives of the organization. The roles and responsibilities are also identified in this stage.

The measurement roles and responsibilities bring additional costs to budget. The source of data brings 2 %, analysis and packaging brings 7 %, and technical support brings 4 % additional costs over budget approximately [4].

The output of the "Build" stage is a Measurement Plan. This plan is actively used at rest of the measurement process. This stage is described at Chapter 5 in details.

The last stage of software measurement process, called "Implementation", is described at Chapter 6 in details. There are 3 sub-tasks in this stage.

- Collect Data: At first, the data or information should be taken from source (Access Data), then ensure that the accessed data is relevant to requirements (Verify Data), finally normalize them to use in analysis (Normalize Data) [5].
- Analyze Issues: This sub-task has some direct relations with technical adequacy, development performance, growth and stability, resources and cost, schedule and progress and product quality. Estimation produces projections of software size, effort, schedule and quality. Feasibility Analysis deals with the technical accuracy and realism of plans, estimates or assumptions. Performance Analysis determines if the project is meeting targets and goals.
- Make Decision: It includes reporting, alternative selection and action. Not every analysis result requires action [5]. The one important point is that people are the most significant factor in software measurement success [11]. While making a decision, this point should not be disregarded.

1.2. The Purpose and Scope of the Study

In this thesis study, a software measurement program has been designed, and then implemented in order to provide a software development process measurement system at YMM Departments of MST Division in ASELSAN Inc. The objective of the study is to demonstrate the viability of software measurement process life-cycle in an existing organization.

There are three key reasons for implementing a software measurement program [4].

- Understanding: The fundamental requirement is to gather information about what organization does and how it operates. Better understanding leads to better management of software projects and improvements in process. It supports the managers make correct decisions.
- Management: Measurement is intended to help the project manager, to make a reasonable decision, not to make an automatic decision. Measurement also assists management processes such as planning, estimating, tracking and validating.
- Guiding Improvement: The primary objective of any software engineering organization is to produce a high-quality product within schedule and budget. This goal can be achieved by improving the software development process. Process improvement can be accomplished by modifying managerial or technical

processes. By measurement program, the organization can find weak points in its processes.

Different levels within the same organization have different information needs. Executive managers usually make investment decisions with respect to software process technology and tools. Project managers make decisions about specific technologies and resources to best satisfy project objectives. As a result, the reason for applying software measurement usually depends on information needs.

1.3. Basic Measures

In Figure 2, the core measures and their application phases are shown clearly. The important attributes in each type of measure are addressing the three key reasons (Understanding, Managing, and Guiding Improvement) and being easy to collect and achieve.

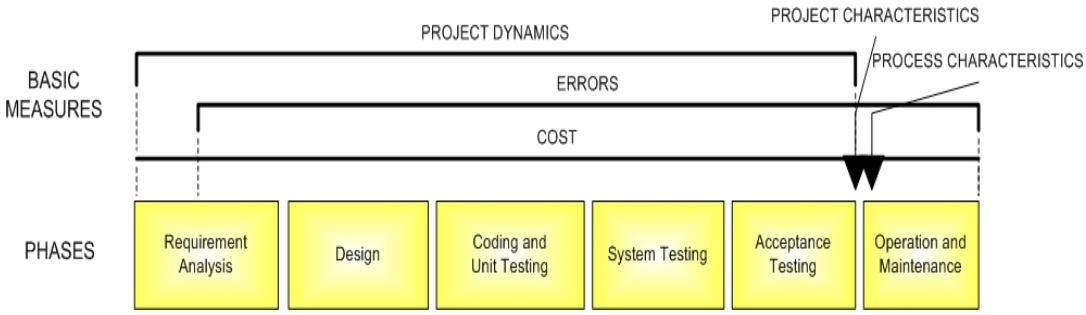


Figure 2 - Basic Measures [4]

The basic measures are cost, errors, process characteristics, project dynamics and project characteristics [4].

- Cost: It can be used for understanding and managing software processes and products. Its scope depends on the organization's goal. Measurement frequency is at least monthly or more frequently if needed.
- Errors: A better understanding of characteristics of software defects is necessary to reach higher quality and greater reliability. This measurement should be applied whenever the controlled unit is modified.
- Process Characteristics: It is applied at the end of acceptance testing phase. It is used for investigation of the effectiveness of various software engineering methods and techniques.
- Project Dynamics: It can be used for controlling the project dynamics that are changes in product requirements and source code. Measurement frequency can be weekly, biweekly or monthly.
- Project Characteristics: It is applied at the end of acceptance testing phase. It can be broken down into 5 categories. The first is "Development Dates" which includes beginning and ending of each life-cycle phase and final project completion date. The second is "Total Effort" which includes hours used by programmers, managers and support services. The third is "Project Size" which includes total size of software product and the total number of components within the product. The fourth is "Component

Information" which includes collecting size and origin information for software components and defines components as separately compliable units. And the last is "Software Classification" as Business/Administrive, Scientific/Engineering, Embedded/Non-Embedded, Real Time/Non-Real Time, and Secure/Nonsecure.

1.4. Types of Metrics

Based on their intended use, software metrics can be classified as [3]

- Process Metrics for improving the software development and maintenance process,
- Product Metrics for improving software product,
- Project Metrics for tracking and improving project.

In Figure 3, the types of quality metrics and their relationships are shown based on the ISO/IEC 9126.

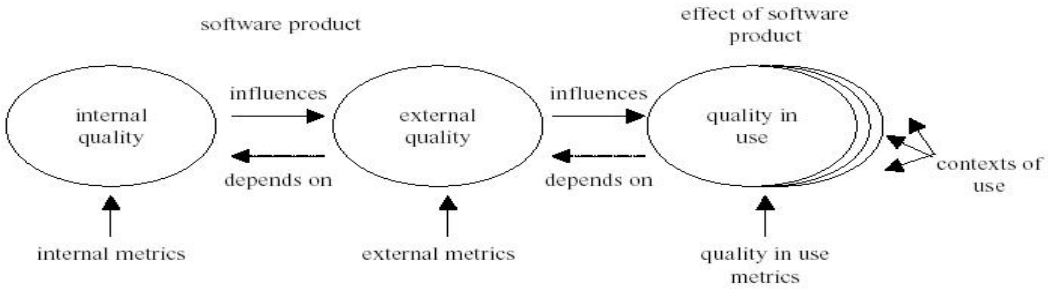


Figure 3 - Types of Metrics [2]

The internal metrics can be applied to a software product during its development stages and they provide features to measure the intermediate

deliverables, and predict final product. The external metrics can only be used during the testing stage of the life cycle process and during any operational stages. The quality in use metrics measure the level of product meets the specified needs and the specified goals. This type of metrics can be used in a realistic system environment [2].

1.5. Outline

Rather than the traditional approach of separating the literature study and the description of specific application, presentation of the approach taken in a specific project realized at ASELSAN Inc. right after a review of the literature on each stage of the measurement process has been preferred in this report.

Hence, this thesis is organized as follows; each chapter is presenting a brief review of related literature, followed by a description of how each stage has been realized in ASELSAN Inc.:

- In Chapter 2, the INITIATION stage of measurement process is described in detail.
- In Chapter 3, the ANALYSIS stage of measurement process is described in detail.
- In Chapter 4, the DESIGN stage of measurement process is described in detail.
- In Chapter 5, the BUILD stage of measurement process is described in detail.

- In Chapter 6, the IMPLEMENTATION stage of measurement process is described in detail.
- Finally, some discussion and concluding remarks are given in Chapter 7.

CHAPTER 2

INITIATION STAGE

2.1. The Organizational Goals

The first stage of measurement process life-cycle is INITIATION. At this stage, the key activities are understanding organizational goals and obtaining organizational support. At the end of this stage, everyone in the organization should understand what measurement process is, and why a measurement program is required. In order to do that, a briefing was given in MST division of ASELSAN Inc.

The prerequisites for applying a software measurement program can be enumerated. These are [13]

- A cost accounting system,
- A software configuration management system, and
- A problem reporting/corrective action system.

In ASELSAN Inc., the Rational's ClearCase tool is actively used for software configuration management, and the Rational's ClearDDTS tool is actively used for problem reporting/corrective action.

What about the reflections of these prerequisites in software industry in Turkey?

The KALDER survey has impressive results and one of them is about the software configuration management system [7]. From this survey, in approximately 35% of the firms, written procedures and standards are deployed but they are partly applied into the process. In addition, approximately 30% of the firms apply some rules but they are not written anywhere. The responsibility of software configuration management is given to project managers in approximately 70% of the firms.

The typical organizational goals are:

- Increasing functionality,
- Reducing cost,
- Reducing time to market (improve timing in schedule), and
- Improving product quality [6].

Apart from that, the organization specific goals can be declared according to its specific structure.

Understanding the organizational goals consists of goals, objectives, and expectations.

2.2. How?

There are two studies that have terrifying results about software projects and measurement process. First was carried out in 1995 by Standish Group and included software projects status. The Standish Survey was applied over 800 software projects and the results were:

- 52.7 % were completed but incurred cost and schedule overruns,
- Average cost overrun was 189%,
- Average schedule overrun was 222%, and
- 31.1% of all projects were cancelled [3].

These results indicate that the software development process must be controlled anyway, and one method is measuring.

The second survey was done by Howard Rubin. The result is

- Within the 610 measurement programs in 1998, only 140 survived after two years [11].

In other words, only one of the five started measurement programs had been survived within two years.

At the first stage of measurement process life-cycle, the organizational goals and objectives are defined. At the second stage, project issues that depend on these defined goals are identified and prioritized. After doing that the appropriate measures are selected. As a result, the measures are selected by goals, objectives, and issues. To make a correct decision about measures, the organizational goals and objectives should be defined correctly at the fist

stage. Table 1 shows a vision of these stages. It can be very useful while defining goals.

Table 1 - The Goals and Issues Relations [6]

Business Goals	Project Issues	Process Issues	Measurable Product and Process Attributes
increase functionality	product growth product stability	product conformance	number of requirements product size product complexity rates of change % nonconforming
reduce cost	budgets expenditure rates	efficiency productivity rework	product size product complexity effort number of changes requirements stability
reduce the time to market	schedule progress	production rate responsiveness	elapsed time, normalized for product characteristics
improve product quality	product performance product correctness product reliability	predictability problem recognition root cause analysis	number of defects introduced effectiveness of defect detection activities

One of the most important points is that each one of the selected measures should be matched with at least one or more of the organizational goals, objectives, and issues. Conversely, each organizational goal, objective, or issue should be matched with corresponding measure(s).

GQM is one of the popular methods for selecting appropriate metrics. This method starts with defining organizational goals and objectives. The goals constitute questions. Finally, the answers of questions form the metrics. In this method, the goals must have some information about object, purpose, quality focus, viewpoint, and environment [12]. In short view,

A GOAL $\rightarrow$ [object] [purpose] [quality focus] [viewpoint] [environment]

2.3. Applications in the Industry

First application example is from Motorola [13]. They use metrics for both process improvement and in-process project control. For Motorola, measurement is not a goal; the goal is improvement with measurement, analysis, and feedback [13]. They adopt GQM model to select appropriate metrics for their measurement program.

An example of use of GQM model for defining metrics is from [13]:

Goal: Decrease Software Defect Density

Question1: What is the currently known effectiveness of the defect detection process prior to release?

Metric 1: Total Defect Containment Effectiveness.

Question 2: What is the currently known containment effectiveness of faults introduced during each constructive phase of software development for a particular software product?

Metric 2: Phase Containment Effectiveness for phase i.

Second application example is from Nokia [14]. They have derived "Nokiaway" metrics program from GQM method. There are some differences between GQM and Nokiaway. GQM identification goals include characterizing projects and organizations, and identifying improvement of both measurement and GQM goals. Nokiaway uses a quality metrics library instead of defining a new set of metrics for each project. In GQM method, person who takes part in the operative tasks in measurement program is a

full-time employee, on the other hand, he is a part-time employee in Nokiaway method [14].

Each organization has special objectives and goals. The author's opinion is that it is too hard to implement the popular methods, which are GQM and PSM, without modification. The mixture of methods can be used in the measurement program. The goal identification stage of GQM is very powerful, and the GQM goal definition, which is mentioned at the previous part, should be used. However, the selection of appropriate measures becomes very easy by using the PSM guide.

In KALDER Survey 2001 in software industry, names of the applied software measures are asked to the firms [7]. Approximately 50% of the answers contain the number of requirements, approximately 45% contain realized effort (person-hours), and approximately 40% contain software errors.

Another important question in the survey is about the obstacles in adopting the measurement process. Approximately 65% of the answers indicate big work-load and approximately 40% refer to the reluctance of staff [7].

2.4. At the Organization

ASELSAN Inc. is the biggest military electronics industry firm in Turkey; in addition the MST Division has mostly system projects. These projects include huge software components. The process and standards that contain managing and engineering activities are well defined in the organization. As

a result of the investigation in 2002 done by Undersecretariat for Defense Industries (Savunma Sanayi Müsteşarlığı), ASELSAN Inc. has obtained "Class-A" software organization qualification.

The error tracking and configuration management systems are actively used at the organization. However, the author thinks that the collected data should be analyzed more effectively. One reason of the scheduling problems in organization is lacking of detailed both software estimation and risk management. The absence of sufficient productivity analysis may be another reason. The effort is measured only as person-month, but systematic calculation of lines of code or module number hasn't been done yet except some projects. The product performance and reliability are important points, so organizational managers indicate that they want to work on these issues and improve them.

In order to realize deployment of software measurement process in MST Division, the process action team PAT-G has been formed in organization. The members are

- L. A., Headmaster of Software Engineering Department,
- A. Z., Headmaster of Software Engineering Department,
- Ö. Ö. E., Technical Leader in Software Engineering,
- G. A., Technical Leader in Software Testing Department,
- A. D., Principal Engineer in Product Quality Department,
- Z. Y., Senior Engineer in Product Quality Department,

- Ö. E., Engineer in Software Engineering Department.

The organizational goals have been listed and prioritized respectively by members of PAT-G,

1. Track and analyze the schedule to improve and minimize it from the viewpoint of software development team leader,
2. Analyze the software product and its functionality to improve the performance,
3. Analyze the development cost in order to minimize it,
4. Evaluate and analyze the productivity to improve it from the viewpoint of department manager, and
5. Collect and analyze required data to make software estimation.

Measurement is a tool to illuminate the project situation to managers [15]. This tool can be more useful and effective when one first understand exactly what one want to accomplish. The defined organizational goals contain this information.

CHAPTER 3

ANALYSIS STAGE

3.1. Introduction to Analysis Stage

Quality Improvement Paradigm (QIP) is defined by V.R. Basili, famous software engineering expert, based on the reusing experience [18]. Experience can be more useful when it is recorded and suitably packaged. The Quality Improvement Paradigm is identified by steps as

1. Characterize the project and its environment,
2. Set quantifiable goals for successful project performance and improvement,
3. Choose the appropriate process model and supporting models,
4. Execute the process, construct the products, collect and validate the prescribe data, and analyze it to provide real-time feedback for corrective action,
5. Analyze the data to evaluate the current practices, determine problems, record findings, and make recommendations for future projects,

6. Package the experience in the form of updated and refined models and save it in an experience base to be reused on future projects [18].

Software measurement is a necessary component for developing experience and retrieving knowledge on software development in the Quality Improvement Paradigm.

In ASELSAN Inc., the measurement program will be firstly deployed in detail, so one of the outcomes of this thesis will be the starting point of metric-experience database.

At analysis stage, issues must be identified and prioritized. Also, the scope of measurement program should be defined.

The organizational goals and issues are related to each other tightly. In order to define project issues, it is necessary to understand what can be an issue. A problem, a risk, or a lack of information can be an issue [5].

- A problem: As an example, development team has some newly graduated members who lack sufficient skills about the project area.
- A risk can be noticed but it is not certain. A risk is a potential problem. As an example, as a result of the slower than estimated speed of project development, slipping in the schedule can occur.
- A lack of information means that the available information is inadequate; e.g., the lack of information about project size to be developed.

It is obvious that not all defined issues are problems. In addition, issue identification and tracking operation may protect the project from probable problems. These issues can vary from project to project, and also change over time within a given project. As a result, besides the issues defined at the beginning of the project, new issues may appear while the project carries on. Furthermore, the list of issues should be revisited periodically during the project life cycle.

In ISO 15939, accepting of requirements for measurement activity includes a clear statement that the scope of measurement shall be identified [1]. At this point, the effective questions which should be answered are:

- Which projects should be included in the organization's measurement program?
- Which phases of the software life cycle should be included?

The answers to these two questions may be a single project, two projects, and one phase of software life cycle.

- Which elements of the project staff should be included?

For example, the effort of one or more level managerial support (i.e., department manager, software development team leader) can be considered as an answer.

3.2. Identify Project Issues

In Figure 4, the model which is derived from Practical Software and Systems Measurement (version 4.0b) is shown. The difference between PSM model

and the derived model is the direct usage of defined organizational goals in the derived model. It has filtering properties in order to prevent unnecessary measures and provide measures within appropriate scope. As emphasized in the previous parts, measurement programs should be driven by organizational goals.

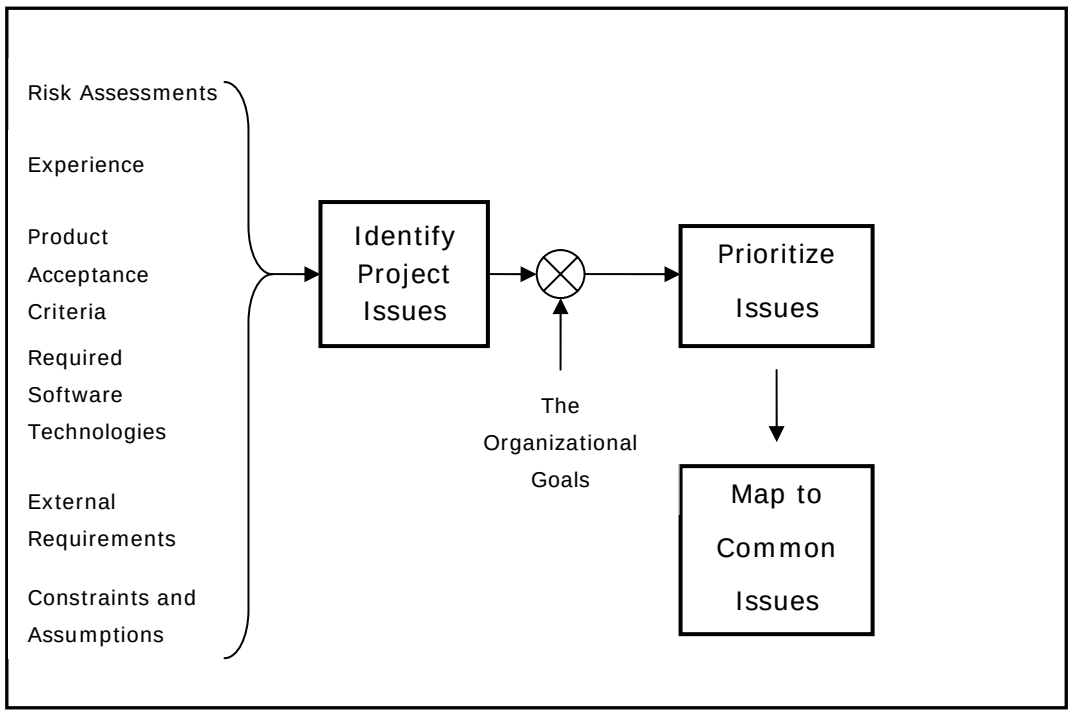


Figure 4 - The Issue Identification Model [5]

When performing the issue identification process, there are useful sources that can help to reveal the correct ones. The sources are [5]

- Risk Assessments: Risk assessment may point to potential requirements, technology, process, cost, and schedule issues. Risk may be identified informally in the absence of structured risk management process.

- The developer's and manager's experience with similar past projects.
- Product Acceptance Criteria: If there is a doubt about the systems capability to meet defined acceptance criteria, then satisfaction of these criteria should be marked as an issue.
- Required Software Technologies: Entire risk assessment can find out this type of issue.
- External Requirements
- Constraints and Assumptions: For example, the lack of information about effort and schedule estimates should be treated as issues. Schedules and budgets are usually inflexible constraints so if some deviations threaten the project success then they are also issues.

The source of identified issues is mostly the lack of information to determine the state. In the relation of the identified goals at previous stage, the issues can be listed as follows.

The risks are

1. The intensive project schedule,
2. Unstable requirements,
3. Constant budget, and
4. Staff experience.

The lack of information about

5. Whether project going on schedule or not,
6. Whether scheduled milestones meeting or not,
7. Whether software product ready to delivery or not,
8. Whether all identified problems resolved or not,
9. Whether staff effort is adequate or not,
10. Whether the number of staff is adequate or not,
11. How much the requirements are changing, and
12. How much the product size is changing,
13. How much difficult the software is to maintain.

In ISO 15939, the requirement of prioritization of the identified information needs is stated clearly. The identified information needs are based on goals, constraints, risks, and problems of the organizational unit. Another important statement is that the selected measures should reflect the priority of the information needs [1].

3.3. Prioritize Issues

In Table 2, the relations between the identified organizational goals listed in Section 2.4 and issues are shown.

Table 2 – The Organizational Goals and Issue(s)

Goal #	Related Issue(s)
1	The risk of the intensive project schedule. The lack of information about whether project going on schedule or not. The lack of information about whether scheduled milestones meeting or not.
2	The lack of information about whether software product ready to delivery or not. The lack of information about whether all identified problems resolved or not. The lack of information about how much difficult the software is to maintain.
3	The risk of constant budget.
4	The risk of staff experience. The lack of information about whether staff effort is adequate or not. The lack of information about whether the number of staff is adequate or not.
5	The risk of unstable requirements. The lack of information about how many the requirements are changing. The lack of information about how much the product's size is changing.

The issues have two important properties, namely their probability and impact according to [5]. Probability contains information about how likely it will result in a problem. The probability of occurrence can be expressed on a scale of 0 to 1. In addition, the impact contains information about what impact it will have on project success if occurs. A scale of 1 to 10 can be used for the impact of an issue [5].

The prioritization formula is from PSM methodology [5] as

Priority = [Probability x Impact].

Table 3 is formed with average of given values from members of the PAT-G.

Table 3 - Issue Prioritization

ISSUE	PROBABILITY		IMPACT		TOTAL
The intensive project schedule	0,9		8		7,2
Unstable requirements	0,8		7		5,6
Constant budget	0,5		4		2,0
Staff experience	0,6		6		3,6
Whether project going on schedule or not	0,7		7		4,9
Whether scheduled milestones meeting or not	0,6	X	7	=	4,2
Whether software product ready to delivery or not	0,6		7		4,2
Whether all identified problems resolved or not	0,6		7		4,2
How much difficult the software is to maintain	0,7		0,6		4,2
Whether staff effort is adequate or not	0,9		8		7,2
Whether the number of staff is adequate or not	0,6		6		3,6
How much the requirements are changing	0,7		7		4,9
How much the product's size is changing	0,4		3		1,2

3.4. Mapping to Common Issues

The defined organizational goal and common issue relation is exhibited by using [5] in Table 4.

Table 4 - Goal and Common Issue Relation [5]

Goal #	Related Common Issue	Questions Addressed
1	Schedule and Progress	-Is the project meeting scheduled milestones? -Are critical tasks or delivery dates slipping? -Is capability being delivered as scheduled in incremental builds and releases?
2	Product Quality	-Is the product good enough for delivery? -Are identified problems being resolved? -How difficult is it to maintain? -Does the target system make efficient use of system resources? -To what extent can the functionality be re-hosted on different platforms? -Are operator errors within acceptable bounds? -Are failure rates within acceptable bounds?
3	Resources and Cost	-Is project spending meeting budget and schedule objectives?
4	Resources and Cost	-Is effort being expended according to plan? -Is there enough staff with the required skills?
5	Product Size and Stability	-How much is the product's size, content, physical characteristics, or interfaces changing? -How much are the requirements and associated functionality changing?

In Practical Software and Systems Measurement Guide, there are seven "common issue areas" which contains the most project-specific software issues based on experiences [5]. The seven common software issues are as follows:

1. Schedule and Progress issue relates to the achievement of major milestones and individual work units.

2. Resources and Cost issue relates to the balance between the work to be performed and personnel resources assigned to the project.

3. Growth and Stability issue relates to the stability of the functionality or capability required of the software.

4. Product Quality issue relates to the ability of the delivered software product to support the user's needs without failure.

5. Process or Development Performance issue relates to the capability of the developer and the life cycle processes relative to project needs.

6. Technology Effectiveness or Technical Adequacy issue relates to the viability of the proposed technical approach.

7. Customer Satisfaction issue relates to the customer's perception of product value.

After combining prioritized issues with common issues, the Table 5 is constructed. It shows the particular relations between organizational and common issues. Table 6 relates organizational goals to common issues.

Table 5 – Common and Related Issues

#	COMMON ISSUE	RELATED ISSUES
1	Schedule and Progress	- The risk of the intensive project schedule. - The lack of information about whether project going on schedule or not. - The lack of information about whether scheduled milestones meeting or not.
2	Product Quality	- The lack of information about whether software product ready to delivery or not. - The lack of information about whether all identified problems resolved or not. - The lack of information about how much difficult the software is to maintain.
3	Resources and Cost	- The risk of staff experience. - The lack of information about whether staff effort is adequate or not. - The lack of information about whether the number of staff is adequate or not. - The risk of constant budget.
4	Product Size and Stability	- The risk of unstable requirements. - The lack of information about how many the requirements are changing. - The lack of information about how much the product's size is changing.

Table 6 – Prioritized Goals

priority	GOAL	Priority	Common Issue
1	Track and analyze the schedule to improve and minimize it from the viewpoint of development team leader.	5,4	Schedule and Progress
2	Analyze the product and its functionality to improve software performance.	4,2	Product Quality
3	Analyze the development cost in order to minimize it.	4,1	Resources and Cost
4	Evaluate and analyze the productivity to improve it from the viewpoint of department headmaster.	4,1	Resources and Cost
5	Collect and analyze required data to make software estimation.	3,2	Product Size and Stability

3.5. Measurement Scope

The three questions that can be informative about which projects should be included at which phases of the software life cycle with which elements of the project staff, have been listed above in section 3.1. With the scope of this activity, all stakeholders, individuals or organizations who sponsor measurement, provide data, and use results, should be identified [5].

Two rules from [6] which can help in defining scope are "focus locally" and "start small". It means that the answers to the questions should be as short as possible.

In ASELSAN Inc., starting one project from analysis stage of software development life cycle is considered on account of the two important rules mentioned above. In addition, the stakeholders are identified as PAT-G team and the software development team leader of the selected project.

CHAPTER 4

DESIGN STAGE

4.1. Introduction to the Design Stage

The important step in establishing a measurement program is selecting the measures to be used. [4]

One of the common measurement problems is "No Measurement Plan/Design". [19] Furthermore, measurement success critical factors are listed in [20] as following:

- Collect meaningful, valid, reliable measures,
- Use consistent measures,
- Management must require and use the derived measurement information,
- Management must be willing to change the process.

The measures, which have these properties, should be clearly defined according to the related to goals. In addition, the required source data should be available [20].

When selecting measures, the next important rule is "make sure the measures apply to the goals". They should directly relate to the defined goals of the organization. For example, if there is no goal to relate with a selected measure, it is a waste of time and effort to collect data about this measure.

Another rule is "keep the number of measures to a minimum" [4].

Steps of selecting and specifying project measures are shown in Figure 5.

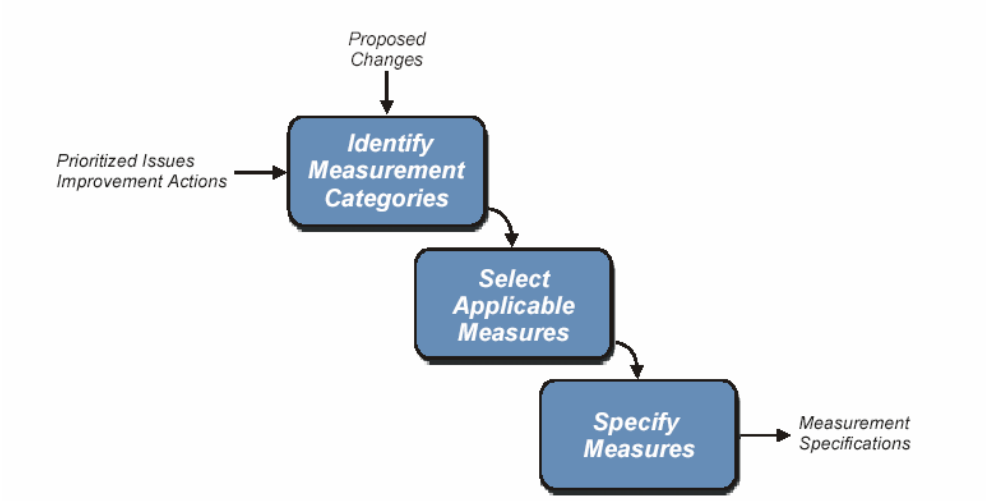


Figure 5 – Selecting Measures [5]

After the analysis stage of measurement life-cycle, the issues and the common issue areas are identified and prioritized. The first step of design stage, which is identifying measurement categories, should be realized by using outputs of the previous stage. While implementing all three steps shown in Figure 5, various types of tables may be used. The tables and included information are critical elements of success of the design stage.

4.2. Issue Measure Mapping

Figure 6 shows the relationship among project issues, common issue areas, measurement categories, and measures. Selecting a common issue area narrows the range of categories; also selecting a category narrows the range of measures that should be considered.

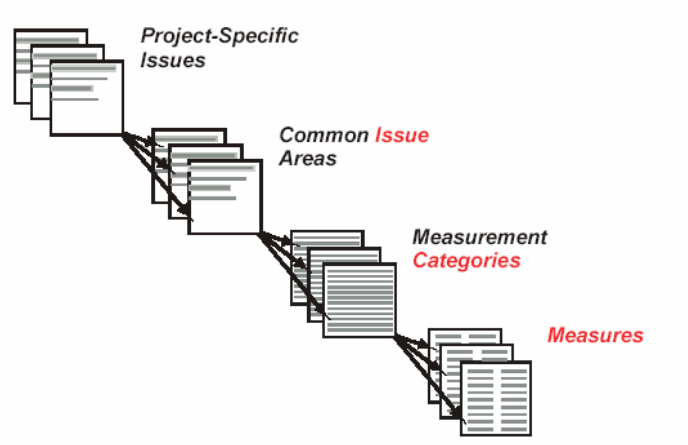


Figure 6 – Measurement Selection Mechanism [5]

One way to determine a category, which addresses an issue, is to consider the table of measurement categories and related questions as shown in Figure 7. For critical or high-priority issues, selecting more than one measurement category should be considered. This will lead to different types of measures, allowing for more effective analysis.

Using Table 5 from the previous stage and Table 7 below, common issues are mapped to measurement categories as shown in Table 8. These tables provide a link between the goals, issues or information needs and the candidate measures.

Table 7 - Measurement Categories and Related Questions [5]

Common Issue Area	*Measurement Category*	*Questions Addressed*
Schedule and Progress	Milestone Performance	Is the project meeting scheduled milestones? Are critical tasks or delivery dates slipping?
	Work Unit Progress	How are specific activities and products progressing?
	Incremental Capability	Is capability being delivered as scheduled in incremental builds and releases?
Resources and Cost	Personnel	Is effort being expended according to plan? Is there enough staff with the required skills?
	Financial Performance	Is project spending meeting budget and schedule objectives?
	Environment and Support Resources	Are needed facilities, equipment, and materials available?
Product Size and Stability	Physical Size and Stability	How much are the product's size, content, physical characteristics, or interfaces changing?
	Functional Size and Stability	How much are the requirements and associated functionality changing?
Product Quality	Functional Correctness	Is the product good enough for delivery to the user? Are identified problems being resolved?
	Supportability - Maintainability	How much maintenance does the system require? How difficult is it to maintain?
	Efficiency	Does the target system make efficient use of system resources?
	Portability	To what extent can the functionality be re-hosted on different platforms?
	Usability	Is the user interface adequate and appropriate for operations? Are operator errors within acceptable bounds?
	Dependability - Reliability	How often is service to users interrupted? Are failure rates within acceptable bounds?
Process Performance	Process Compliance	How consistently does the project implement the defined processes?
	Process Efficiency	Are the processes efficient enough to meet current commitments and planned objectives?
	Process Effectiveness	How much additional effort is being expended due to rework?
Technology Effectiveness	Technology Suitability	Can technology meet all allocated requirements, or will additional technology be needed?
	Impact	Is the expected impact of the leveraged technology being realized?
	Technology Volatility	Does new technology pose a risk due to too Many changes?
Customer Satisfaction	Customer Feedback	How do our customers perceive the performance on this project? Is the project meeting user expectations?
	Customer Support	How quickly are customer support requests being addressed?

Table 8 - Common Issue Mapping to Categories

#	COMMON ISSUE	MEASUREMENT CATEGORY
1	Schedule and Progress	Milestone Performance Work Unit Progress
2	Product Quality	Functional Correctness Supportability and Maintainability
3	Resources and Cost	Personnel Financial Performance
4	Product Size and Stability	Physical Size and Stability Functional Size and Stability

In Table 9, the relationship between the project issues and measurement category is shown and it is formed by using Table 8 and Table 5.

Table 9 - Related Issues and Measurement Categories

ISSUE	MEASUREMENT CATEGORY
- The risk of the intensive project schedule. - The lack of information about whether project going on schedule. - The lack of information about whether scheduled milestones meeting or not.	Milestone Performance Work Unit Progress
- The lack of information about whether software product ready to delivery or not. - The lack of information about whether all identified problems resolved or not. - The lack of information about how much difficult the software is to maintain.	Functional Correctness Supportability and Maintainability
- The risk of staff experience. - The lack of information about whether staff effort is adequate. - The lack of information about whether the number of staff is adequate or not. - The risk of constant budget.	Personnel Financial Performance
- The risk of unstable requirements. - The lack of information about how many the requirements are changing. - The lack of information about how much the product's size is changing.	Physical Size and Stability Functional Size and Stability

In Table 10, the whole relationship among common issues, measurement categories, and measures is shown clearly. List of the measures corresponding to selected categories is also given [5]. Description tables, where the properties of a measure are listed in detail, are very useful and suitable for a measurement program.

Table 10 – I-C-M Mapping [5]

Issue - Category - Measure Mapping		
Common Issue Area	**Measurement Category**	**Measures**
Schedule and Progress	Milestone Performance	Milestone Dates
	Work Unit Progress	Critical Path Performance Requirements Status Problem Report Status Review Status Change Request Status Component Status Test Status Action Item Status
	Incremental Capability	Increment Content - Components Increment Content - Functions
Resources and Cost	Personnel	Effort Staff Experience Staff Turnover
	Financial Performance	Earned Value Cost
	Environment and Support Resources	Resource Availability Resource Utilization
Product Size and Stability	Physical Size and Stability	Database Size Components Interfaces Lines of Code Physical Dimensions
	Functional Size and Stability	Requirements Functional Change Workload Function Points
Product Quality	Functional Correctness	Defects Technical Performance
	Supportability - Maintainability	Time to Restore Cyclomatic Complexity Maintenance Actions
	Efficiency	Utilization Throughput Timing
	Portability Usability Dependability - Reliability	Standards Compliance Operator Errors Failures Fault Tolerance

4.3. Schedule Measures

In the previous section as shown in Table 8, the Milestone Performance and the Work Unit Progress measurement categories are selected for Schedule and Progress common issue.

The Milestone Performance measures provide basic schedule and progress information for key development activities and events. The measures also help to identify and assess dependencies among development activities. Monitoring schedule changes helps to assess the risk in achieving future milestones. This category is applicable to all types and sizes of projects, and all process models. The measures of this category do not address the amount of effort to complete a scheduled activity [5]. The measures of Milestone Performance category are shown in Table 10.

Work Unit Progress measures address progress, based on the completion of hardware and/or software work units. If objective completion criteria are defined, Work Unit Progress measures are very effective for assessing progress at any point in the project. This category is applicable to all types and sizes of projects, and all product-oriented process models [5]. The measures of Work Unit Progress category are shown in Table 10.

The list of candidate measures for Schedule and Progress common issues is presented in Table 11.

Table 11 – Schedule Measurement Candidates

	CATEGORY	MEASURES
Schedule And Progress	Milestone Performance	· Milestone Dates · Critical Path Performance
	Work Unit Progress	· Requirements Status · Problem Report Status · Review Status · Change Request Status · Component Status · Test Status · Action Item Status

The following decisions about candidate measures are made by the author:

- The Milestone Dates Measure is selected as one of measures in the software measurement program, since required data for this measurement can be obtained easily from MS Project tool which is actively used at the MST-YMM departments of ASELSAN Inc. On account of the structural properties of software development process in ASELSAN Inc., the data items for this measure will be collected for each SCU (Software Configuration Unit) of project.
- In the Critical Path Performance Measure, all schedule dependencies, and assumptions and causes of dependency between activities should be identified in order to determine and track dependent activities. Because of the decision to "focus locally and start small", the Critical Path Performance Measure is excluded from the measurement program.

- The Requirements Status Measure is selected on grounds of applied test activities both in YMM and software test departments of MST division.
- The Problem Report Status Measure is selected on grounds of the fact that in the MST Division the most projects use a problem reporting system, whose name is ClearDDTS.
- The Review Status Measure is selected since review activities in software development process are in use at entire organization. Due to properties of the process in organization, the description and data items in PSM table may need changes.
- The change request system is actively in use at most projects in the MST Division of ASELSAN. Therefore, the Change Request Status Measure is selected.
- The Component Status Measure is selected since the necessity of configuration management system is provided in YMM departments. The required data can also be obtained from documentation process in the development.
- The Test Status Measure is selected due to similar reasons in the selection of the Requirements Status Measure. Both YMM and YT departments are applied test activities with procedural manner.
- A process for identifying, handling, and tracking action items is partially available in organization, so the requirement of the

Action Item Status Measure is not achieved completely. As a result, this candidate measure is not selected.

Detailed information about the measures is given in description tables.

- Table 12 contains The Milestone Dates Measure,
- Table 13 contains The Requirements Status Measure,
- Table 14 contains The Problem Report Status Measure,
- Table 15 contains The Review Status Measure,
- Table 16 contains The Change Request Status Measure,
- Table 17 contains The Component Status Measure,
- Table 18 contains The Test Status Measure,

In tables:

- **Typical Data Items** identifies typical data that is collected in the measure,
- **Typical Attributes** are characteristics or properties used to categorize the data,
- **Typical Aggregation Structure** is the levels used to aggregate data to the system level including component, function, or activity,
- **Counts Actuals Based On** identifies typical exit criteria used to determine when a measure is counted as an actual.

Table 12 - Milestone Dates Measure [5]

Description: Milestone Dates measures the start and end dates for activities, events, and products. The measure provides a view of scheduled activities. Comparison of plan and actual milestone dates provides insight into significant and repetitive schedule changes at the activity level.

Selection Guidance

Project Application
Applicable to all sizes and types of projects. Included in most government and industry measurement practices.

Process Integration
Required data is generally obtained from project scheduling systems and/or documentation.
Detailed milestones provide a better indication of progress and allow earlier identification of problems.
If activities or events are re-planned to occur at a different time, the original dates should be retained to observe planned schedule changes.

Usually Applied During
Project Planning (Estimates)
Requirements Analysis (Estimates and Actuals)
Design (Estimates and Actuals)
Implementation (Estimates and Actuals)
Integration and Test (Estimates and Actuals)
Operations and Maintenance (Estimates and Actuals)

Specification Guidance

Typical Data Items
Start date of activity or event
End date of activity or event

Typical Attributes
Activity or event name
Version of the plan
Increment

Typical Aggregation Structure
Component
Activity

Count Actuals Based On
Documents base lined
Milestone review held
Successful completion of tasks

This measure answers questions such as:
Is the current schedule realistic?
How many activities are concurrently scheduled?
How often has the schedule changed?
What is the projected completion date for the project?
What activities, events, or products are on time, ahead of schedule, or behind schedule?

Table 13 - Requirements Status Measure [5]

Description: It counts the number of defined requirements that have been allocated to test cases, and the number that have been successfully tested. The measure is an indication of product design and test progress. When used to measure test status, the measure is used to evaluate whether required functionality has been successfully demonstrated against the specified requirements, and the amount of testing that has been performed. The measure provides excellent test coverage and is also known as "Breadth of Testing."

Selection Guidance

Project Application
Generally applicable to all sizes and types of projects with a requirements or design activity.

Process Integration
Requires disciplined requirements traceability and testing processes for successful implementation. Allocated requirements should be testable and mapped to test sequences.
Some requirements may not be testable until late in the testing process. Others are not directly testable. Some may be verified by inspection.

Usually Applied During
Requirements Analysis (Estimates)
Design (Estimates and Actuals)
Implementation (Estimates and Actuals)
Integration and Test (Estimates and Actuals)

Specification Guidance

Typical Data Items
Total number of requirements
Number of requirements traced to detailed specifications
Number of requirements traced to test specifications
Number of requirements tested successfully

Typical Attributes
Increment
Specification reference
Test sequence reference

Typical Aggregation Structure
Function

Count Actuals Based On
Completion of specification review
Baselining of specifications
Baselining of requirements traceability matrix
Successful completion of all tests in the appropriate test sequence

This measure answers questions such as:
Are the requirements being tested as scheduled?
Is implementation of the requirements behind or ahead of schedule?

Table 14 - Problem Report Status Measure [5]

Description: Problem Report Status counts the number of hardware or software problems reported and resolved. This measure provides an indication of product maturity and readiness for delivery. The rates at which problem reports are written and resolved can be used to estimate product completion. This measure can also indicate the quality of the problem resolution process, based on the average age of reported problems and the average time to resolve them.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all sizes and types of projects. **Process Integration** Data is generally available, since most projects have an established problem reporting system. On development projects, data is generally available during integration and test. Problem report data is more difficult to collect earlier (during requirements analysis, design, and implementation) because a formal problem reporting system is usually not in place and enforced. When this data is available, it provides good progress information. An inspection or peer review can provide this information. Projects may track the phase or source where the problem was injected and detected. **Usually Applied During** Requirements Analysis (Estimates) Design (Estimates and Actuals) Implementation (Estimates and Actuals) Integration and Test (Estimates and Actuals) Operations and Maintenance (Actuals)	**Typical Data Items** Number of problem reported Number of problem resolved Average age of problems Average time between assignment and resolving Average time between submission and opening **Typical Attributes** Increment **Typical Aggregation Structure** Component **Count Actuals Based On** Problem reported Problem resolved **This measure answers questions such as:** Are open problem being closed at a sufficient rate to meet the test completion date? Is the product maturing? When will testing be complete? What components have the most problem reports?

Table 15 - Review Status Measure [5]

Description: The measure provides an indication of progress in completing review activities. The Review Status measure also counts the number of types of review items determined during the review process. The relationship between total identified numbers in review and total page number of reviewed software product can be established by using the results of this measure.

Selection Guidance	*Specification Guidance*
Project Application Used on medium to large projects. **Process Integration** Easy to collect if formal reviews are a part of the development process. **Usually Applied During** Requirements Analysis (Estimates and Actuals) Design (Estimates and Actuals) Implementation (Estimates and Actuals) Integration and Test (Estimates and Actuals)	**Typical Data Items** Number of reviews Number of reviews completed successfully Number of “important” items Number of "minor" items Number of “incomprehensible” items Number of “total” items Number of items which are not agreed on at meeting. **Typical Attributes** Name of the component being reviewed Increment **Typical Aggregation Structure** Component **Alternatives to Reviews Include** Inspections Walkthroughs **Count Actuals Based On** Completion of review Resolution of all associated action items **This measure answers questions such as:** What types of review items are determined?

Table 16 - Change Request Status Measure [5]

Description: The Change Request Status measure counts the total number of change requests that affect a product. The measure provides an indication of the amount of rework that has been performed or will be required. This measure only identifies the number of changes; it does not report on the functional impact of changes or the amount of effort required to implement them.

Selection Guidance

Project Application
Applicable to all sizes and types of projects. Often used on operations and maintenance programs.

Process Integration
Data should be available from most projects that put Change Requests under configuration control.

Usually Applied During
Requirements Analysis (Actuals)
Design (Actuals)
Implementation (Actuals)
Integration and Test (Actuals)
Operations and Maintenance (Actuals)

Specification Guidance

Typical Data Items
Number of change requests generated
Number of change requests resolved

Typical Attributes
Increment
Priority
Change classification (defect correction, enhancement)
Valid/Invalid

Typical Aggregation Structure
Function
Component

Count Actuals Based On
Change Request Approval
Change Request Implemented
Change Request Integrated
Change Request Tested

Alternatives to Change Requests Include:
Enhancements
Corrective Action Reports
Engineering Change Proposals

This measure answers questions such as:
How many change requests have impacted the product?
Are change requests being implemented at a sufficient rate to meet the schedule?
Is the trend of new change requests decreasing as the project nears completion?

Table 17 - Component Status Measure [5]

Description: The Component Status measure counts the number of hardware or software components that complete a specific activity. A comparison of plans and actual helps assess the status of development progress. Early in the development activity, planning changes should be expected. Later in the process, an increase in the planned number of components that are scheduled for a specific activity may indicate unplanned or excessive growth.

Selection Guidance

Project Application
Usually used on medium to large projects.

Process Integration
Easier to collect if formal reviews, inspections, or walkthroughs are included in the development process.
Data is sometimes available from configuration management systems or development tools.
Data is generally available if there is a mature and disciplined development process.
Component status during system test activities is generally one of the more difficult Work Unit Progress measures to collect since most integration and test activities are based on requirements or functions instead of components.

Usually Applied During
Requirements Analysis (Estimates)
Design (Estimates and Actuals)
Implementation (Estimates and Actuals)
Integration and Test (Estimates and Actuals)
Operations and Maintenance (Estimates and Actuals)

Specification Guidance

Typical Data Items
Total number of components
Number of components completed successfully

Typical Attributes
Increment

Typical Aggregation Structure
Component

Additional Information
Progress can be measured for individual processes such as preliminary design, detailed design, implementation, component test.

Count Actuals Based On
Completion of component reviews, inspections, or walkthroughs
Successful completion of specified test
Release to configuration management

This answers questions such as:
Are components completing development activities as scheduled?
Is the planned rate of completion realistic?
What components are behind schedule?

Table 18 - Test Status Measure [5]

Description: The Test Status measure counts the number of test cases that have been attempted and the number that have been completed successfully. This measure can be used with the Requirement Status measure to evaluate test progress. This measure helps assess product quality based on the proportion of attempted test cases that have been successfully executed.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all sizes and types of projects. Especially important to those projects with high reliability requirements, security implications, or catastrophic failure potential. **Process Integration** Disciplined test planning and tracking processes are needed to implement this measure successfully. There should be a mapping between defined test cases and requirements to analyze which functions are passing test and which ones are not. Easy to collect if projects define and allocate a quantifiable number of test cases to each product test sequence. Can utilize design or architecture information, concentrating on interfaces among components or configuration items. **Usually Applied During** Integration and Test (Estimates and Actuals) Operations and Maintenance (Estimates and Actuals)	**Typical Data Items** Total number of test cases Number of test cases attempted Number of test cases passed **Typical Attributes** Increment Test environment Test configuration **Typical Aggregation Structure** Component **Count Actuals Based On** Successful completion of each test case in the appropriate test sequence **Alternatives to Test Cases Include:** Test procedures Test threads Logical paths **This measure answers questions such as:** Is test progress sufficient to meet the schedule? Is the planned rate of testing realistic? What functions have been tested or are behind schedule?

4.4. Product Quality Measures

The Functional Correctness is selected for Product Quality common issue.

The measures of Functional Correctness identify the accuracy that is achieved in product functions and the number of functional defects that are observed. These measures provide an indication of product quality. This category is applicable to all types and sizes of projects, and all process models. Measures in this category do not address the effort that is required to implement changes to correct the problems. A defect results from a product's non-conformance with its functional specification, or a deficiency in that specification [5]. The measures of Functional Correctness category were shown in Table 10.

The list of candidate measures for Product Quality common issues is shown in Table 19.

Table 19 – Product Quality Measurement Candidates

	CATEGORY	MEASURES
Product Quality	Functional Correctness	· Defects · Technical Performance
	Supportability and Maintainability	· Cyclomatic Complexity Measure

The following decisions about candidate measures are made by the author:

- The ClearDDTS tool, which is used actively within organization, provides information about defects. As a result, The Defects

Measure is selected as one of measures in the software measurement program.

- The Technical Performance Measure is selected because of importance of performance information about system component functions, response time, data handling capability, and signal processing in real time embedded softwares. In addition, the required data can also be obtained from functional test records. The data items in measurement table are modified according to properties of the projects in organization.
- The software maintenance has an important role in software projects within organization. So the Cyclomatic Complexity Measure is selected.

Detailed information about the measures is given in description tables.

- Table 20 contains The Defects Measure,
- Table 21 contains The Technical Performance Measure,
- Table 22 contains The Cyclomatic Complexity Measure.

Table 20 - Defects Measure [5]

Description: The Defects measure provides useful information on the ability of a supplier to find and fix defects in hardware, software or documentation. The number of defects indicates the amount of rework, and has a direct impact on quality. Arrival rates can indicate product maturity. Closure rates can be used to predict test completion. Tracking the length of time that defects have remained open can be used to determine whether progress is being made in fixing defects, or whether rework is being deferred. A Defect Density measure, which is an expression of the number of defects in a quantity of product, can be derived from this measure.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all sizes and types of projects. **Process Integration** Requires a well-defined testing and inspection process and a disciplined defect tracking process. Easy to collect actual when an automated defect tracking system is used. The number of discovered defects is relative to the amount of discovery activity, such as number of inspections and amount of testing. Defect density requires the collection of both defect and size data for each component. **Usually Applied During** Requirements Analysis (Estimates and Actuals) Design (Estimates and Actuals) Implementation (Estimates and Actuals) Integration and Test (Estimates and Actuals) Operations and Maintenance (Actuals)	**Typical Data Items** Defect Statistics **Typical Attributes** Increment Defect Status Defect Severity Defect Category When Found How Found When Fixed How Resolved **Typical Aggregation Structure** Component **Count Actuals Based On** Defects accepted Defects validated Defect correction successfully tested/inspected Defect assessment of readiness for delivery **This measure answers questions such as:** How many critical defects have been reported for each component? Do defect reporting and closure rates support the scheduled completion date of integration and test? What components have a disproportionate amount of defects, and therefore require additional testing, review, or are candidates for rework?

Table 21 - Technical Performance Measure [5]

Description: The Technical Performance measure is a combination of other measures that are defined by the system's functional and technical requirements. These measures address any functional characteristics that can be quantitatively defined and demonstrated. Various types of functional requirements may be measured including user and mission functions, interoperability of components, security features, accuracy of the system component functions, response time, data handling capability, or signal processing. These measures provide an indication of the overall ability of a system to meet the users' functional requirements.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all sizes and types of projects. **Process Integration** It is often difficult to generate accurate estimates early in the project, especially for new technologies and new projects. Data may not be available until late in a project, when system functional testing is performed. Resource and technology limitations may prohibit demonstration and measurement of all technical performance parameters. Data may be available from functional test records. Modeling and simulation results may be used to estimate functional performance levels. Specific measures are defined by the technical requirements of the system, software and components. **Usually Applied During** Requirements Analysis (Estimates) Design (Estimates) Implementation (Estimates and Actuals) Integration and Test (Actuals) Operations and Maintenance (Actuals)	**Typical Data Items** Datum Interface Speed Block Processing Speed **Typical Attributes** Increment **Typical Aggregation Structure** Component Function **Count Actuals Based On** Passing functional test **This measure answers questions such as:** How accurate was the signal processing function in this release? Is the system able to read all the required data files in the required time? Was the system able to perform all required functions within the specified system response time?

Table 22 – Cyclomatic Complexity Measure [5]

<table>
<tr><td colspan="2">Description: The Cyclomatic Complexity measure is usually applied to count the number of unique logical paths in a software component. However, the concept of Cyclomatic Complexity also can be used to evaluate the complexity of control or information flow in a system. This measure provides an indication of both design quality and the amount of testing required. A high complexity rating is often a leading indicator of a high defect rate. Components with high complexity usually require additional reviews, increased, testing, or rewriting.</td></tr>
<tr><td>Selection Guidance

Project Application
Applicable to projects with testability, reliability, or maintainability concerns.

Process Integration
Operational requirements may require efficient, highly complex code.
The interpretation of complexity is different for each high-order language.
Estimates are generally not produced, but a desired threshold or expected distribution may be specified, based on experience.

Usually Applied During
Design (Estimates)
Implementation (Estimates and Actuals)
Integration and Test (Actuals)
Operations and Maintenance (Actuals)</td><td>Specification Guidance

Typical Data Items
Complexity Value

Typical Attributes
Increment
Source (new, reused, or COTS)
Language
Delivery status (deliverable, non-deliverable)

Typical Aggregation Structure
Component

Count Actuals Based On
Passing inspection
Passing component test
Release to configuration management

This measure answers questions such as:
How many complex components exist in this project?
What components are the most complex?
What components should be subject to additional testing or reviews?
What is the minimum number of test cases required to test the logical paths through the component?</td></tr>
</table>

4.5. Resource and Cost Measures

The Personnel and Financial Performance measurement categories are selected for Resource and Cost common issue.

The Personnel characterize the amount of effort that is planned and expended by defined activities or products. These measures may also describe the number and experience of personnel assigned to a project and may evaluate the rate at which people are added to and removed from a project. Personnel measures can be used to assess the adequacy of planned effort and to analyze the actual allocation of labor. They are essential to evaluating development productivity. Personnel measures are especially critical for a software project, since it is a labor-intensive process. Measures are not always available at lower levels of product and activity detail. Measures may not capture the total effort applied to a project if they do not distinguish between full and part-time personnel. This category is applicable to all types and sizes of projects, and all process models [5]. The measures of Personnel category were shown in Table 10.

The Financial Performance measures report the difference between budgeted and actual cost for a specific product or activity. These measures are used to assess whether the project can be completed within cost and schedule constraints, and to identify potential cost overruns. This category is applicable to all types and sizes of projects, and all process models [5]. The measures of Financial Performance category were shown in Table 10.

The list of candidate measures for Resource and Cost common issues is shown in Table 23.

Table 23 – Resource and Cost Measurement Candidates

	CATEGORY	MEASURES
Resource and Cost	Personnel	· Effort · Staff Experience · Staff Turnover
	Financial Performance	· Cost

The following decisions about candidate measures are made by the author:

- The Effort Measures is selected since the managers' request to measure the performance absolutely. In addition, the required data can be obtained from "İşçilik Bildirim Sistemi" which is actively used in the MST division of ASELSAN Inc.
- The Staff Experience Measure is selected since required data is available in related organization. Also, the managers records personnel information about their staff.
- The Staff Turnover Measure is selected due to similar reasons in the selection of the Staff Experience Measure. These two measures are highly related with each other.
- Demonstration of variation in cost against progression in schedule is useful in order to track the financial performance within organization. However, accessing required data may become

unavailable because of the organizational rules. As a result, the Cost Measure is not selected.

Detailed information about the measures is given in description tables.

- Table 24 contains The Effort Measure,
- Table 25 contains The Staff Experience Measure,
- Table 26 contains The Staff Turnover Measure.

In tables:

- **Typical Data Items** identifies typical data that is collected for the measure,
- **Typical Attributes** are characteristics or properties used to categorize the data,
- **Typical Aggregation Structure** is the levels used to aggregate data to the system level including component, function, or activity,
- **Counts Actuals Based On** identifies typical exit criteria used to determine when a measure is counted as an actual.

Table 24 - Effort Measure [5]

Description: The Effort measure counts the number of labor hours or number of personnel applied to all tasks. This measure can address cost, Schedule and Progress, and Process Performance.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all sizes and types of projects. **Process Integration** Data is usually derived from a financial accounting and reporting system. This measure is most effective when financial accounting and reporting systems are directly tied to individual products and activities at a WBS component level of detail. Counting personnel may be difficult if they are not allocated to the project on a full-time basis or if they are assigned to more than one WBS component. Planning data is usually based on estimation models, historical data, or engineering judgment. **Usually Applied During** Project Planning (Estimates) Requirements Analysis (Estimates and Actuals) Design (Estimates and Actuals) Implementation (Estimates and Actuals) Integration and Test (Estimates and Actuals) Operations and Maintenance (Estimates and Actuals)	**Typical Data Items** Number of labor hours (days, months, etc.) Number of personnel **Typical Attributes** Labor category Increment **Typical Aggregation Structure** Activity/Component **Count Actuals Based On** Financial reporting criteria **This measure answers questions such as:** Are development resources being applied according to plan? Are certain tasks or activities taking more or less effort than expected?

Table 25 – Staff Experience Measures [5]

Description: The Staff Experience measure counts the total number of experienced personnel in defined areas. The measure determines whether sufficient experienced personnel are available.

Selection Guidance	*Specification Guidance*
Project Application Applicable to projects that require particular expertise and level of experience to complete. **Process Integration** Requires a personnel database that includes experience data. Difficult to collect and keep up-to-date as people are added to or removed from a project. Generally has to be maintained manually. Experience factor may be defined for software language, system engineering discipline, domain, hardware, application, platform, and length of time together as a team. **Usually Applied During** Project Planning (Estimates) Requirements Analysis (Actuals) Design (Actuals) Implementation (Actuals) Integration and Test (Actuals) Operations and Maintenance (Actuals)	**Typical Data Items** Number of personnel Number of years of experience **Typical Attributes** Branch (GUI, Control, DSP) **Typical Aggregation Structure** Activity **Typically Collected for Each** Project **Count Actuals Based On** Staff changes **This measure answers questions such as:** Are sufficient experienced personnel available? Will additional training be required?

Table 26 - Staff Turnover Measures [5]

Description: The Staff Turnover measure counts staff losses and gains. Excessive turnover impacts learning curves, productivity, and the ability of the supplier to implement the system with the resources provided within cost and schedule.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all sizes and types of projects. **Process Integration** It is useful to categorize the number of personnel lost into planned and unplanned losses, since most projects plan to add and remove personnel at various stages of the project. Experience factor may be defined for software language, system engineering discipline, domain, hardware, application, platform, and length of time together as a team. **Usually Applied During** Requirements Analysis (Actuals) Design (Actuals) Implementation (Actuals) Integration and Test (Actuals) Operations and Maintenance (Actuals)	**Typical Data Items** Number of personnel Number of personnel gained (per period) Number of personnel lost (per period) **Typical Attributes** Experience factor Branch (GUI, Control, DSP) Sex (Male/Female) Degree (MSc., PHD, ..) School (METU, BU, HU) **Typical Aggregation Structure** Activity **Typically Collected for Each** Project **Count Actuals Based On** Financial reporting criteria Organization restructuring or new organizational charts **This measure answers questions such as:** How many people have been added or have left the project? How are the experience levels being affected by the turnover rates? What areas are most affected by turnover?

4.6. Size and Stability Measures

The Physical Size and Stability, and Functional Size and Stability are selected for Product Size and Stability common issue.

Physical Size and Stability measures quantify the physical size of a system or product. Size is a critical factor for estimating development schedules and costs. These measures also provide information on the amount and frequency of change to products. This category is applicable to all types and sizes of projects, and all process models. Physical size measures do not always map directly to the amount of functionality in the system. Measures in this category do not generally address product quality, complexity, or difficulty. Accurate estimates are dependent on the availability of good historical data or engineering experience [5]. The measures of Physical Size and Stability category were shown in Table 10.

Functional Size and Stability measures quantify the functionality of a system or product. Functional size may be used to estimate development schedule and cost. These measures also provide information about the amount and frequency of change to the system's functionality. Functional changes generally correlate to effort, cost, schedule, and product size changes. This category is applicable to all types and sizes of projects, and all process models. Functional size does not generally address the quality of the product or system measured [5]. The measures of Functional Size and Stability category were shown in Table 10.

The list of candidate measures for Product Size and Stability common issues is shown in Table 27.

Table 27 – Product Size and Stability Measurement Candidates

	CATEGORY	MEASURES
Product Size and Stability	Physical Size and Stability	· Database Size · Components · Interfaces · Source File
	Functional Size and Stability	· Requirements · Functional Change Workload · Function Points

The following decisions about candidate measures are made by the author:

- The candidates in Physical Size and Stability category are basic, easy, and fundamental measures and an organization should measure these metrics. Therefore, Database Size, Components, Interfaces, and Source File measures are selected. The Source File measure is more enhanced according to available data from this measurement.
- Tracking changes in user requirements is required for ASELSAN Inc., where customer requirements are mostly change during development process. The Requirements Measure provides the data in order to evaluate the variation in requirements.
- The Functional Change Workload Measure is very convenient to determine (and estimate) the amount of person-hour required for implementing functional change.

- The Function Points Measure can be used to estimate weighted factor in function point's evaluation, and normalize productivity measures. In order to construct the base for Function Point Analysis in the organization, the measurement table will be formed after a study in the direction of collected data from related measures.

Detailed information about the measures is given in description tables.

- Table 28 contains The Database Size Measure,
- Table 29 contains The Components Measure,
- Table 30 contains The Interfaces Measure,
- Table 31 contains The Source File Measure,
- Table 32 contains The Requirements Measure.

In tables:

- **Typical Data Items** identifies typical data that is collected for the measure,
- **Typical Attributes** are characteristics or properties used to categorize the data,
- **Typical Aggregation Structure** is the levels used to aggregate data to the system level including component, function, or activity,
- **Counts Actuals Based On** identifies typical exit criteria used to determine when a measure is counted as an actual.

Table 28 - Database Size Measure [5]

Description: The Database Size measure counts the number of words, records, or tables in each database. The measure indicates how much data must be handled by the system.

Selection Guidance

Project Application
Applicable to all domains. Often used on information system software projects.
Used for any project with significant database processing. Especially important for those with performance constraints.

Process Integration
In order to estimate the size of a database, a data model and an operational profile must be developed. This is generally a manual process that can be difficult.
Actuals are relatively easy to collect.

Usually Applied During
Requirements Analysis (Estimates)
Design (Estimates)
Implementation (Estimates and Actuals)
Integration and Test (Actuals)
Operations and Maintenance (Actuals)

Specification Guidance

Typical Data Items
Number of tables
Number of records or entries
Number of words or bytes

Typical Attributes
Increment
Database identifier

Typical Aggregation Structure
Component

Count Actuals Based On
Schema design released to configuration management
Schema implementation released to configuration management

This measure answers questions such as:
How much data has to be handled by the system?
How many different data types have to be addressed?

Table 29 - Components Measure [5]

Description: The Components measure counts the number of elementary components in a system or product, and the number that are added, modified, or deleted. The total number of components defines the size of the system. Changes in the number of estimated and actual components indicate risk due to product size volatility and additional work that may be required.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all sizes and types of projects. **Process Integration** Requires a well-defined and consistent component allocation structure. Required data is generally easy to obtain from design tools, configuration management tools, or documentation. Counts of deleted and added components are relatively easy to collect. Modified components are sometimes not tracked. Volatility in the planned number of components may represent instability in the requirements or in the design of the system or product. **Usually Applied During** Requirements Analysis (Estimates) Design (Estimates and Actuals) Implementation (Estimates and Actuals) Integration and Test (Actuals) Operations and Maintenance (Actuals)	**Typical Data Items** Number of units Number of units added Number of units deleted Number of units modified **Typical Attributes** Increment Source (new, reused, or COTS) Language (if software) Delivery status (deliverable, non-deliverable) End-use environment (operational, support) **Typical Aggregation Structure** Component **Count Actuals Based On** Release to configuration management Passing unit test Passing inspection **This measure answers questions such as:** How many components need to be implemented and tested? How much has the approved system baseline changed? Have the components allocated to each increment changed?

Table 30 - Interfaces Measure [5]

Description: This measure is particularly useful when allocating functions during architecture development, to quantify the number of pair-wise relationships between components. This measure also counts the number of interfaces that are added, modified, or deleted. Changes in the number of estimated and actual interfaces indicate risk due to requirements, architectural, or design volatility and may result in additional work.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all application domains. Applicable to all sizes and types of projects, generally with different interface definitions. **Process Integration** Requires a definition of the component level where interfaces must be counted. Requires a well-defined and consistently detailed architecture or design. Required data is generally easy to obtain from design tools, configuration management tools, or documentation. Counts of deleted and added interfaces are relatively easy to collect; counts of modified interfaces are more difficult to obtain. **Usually Applied During** Requirements Analysis (Estimates) Design (Estimates and Actuals) Implementation (Actuals) Integration and Test (Actuals)	**Typical Data Items** Number of interfaces Number of interfaces added Number of interfaces deleted Number of interfaces modified **Typical Attributes** Increment Nature of interface (e.g., data, control signals, mechanical action) **Typical Aggregation Structure** Component **Count Actuals Based On** Release to configuration management Passing an integration test **This measure answers questions such as:** How many interfaces need to be implemented and tested? How much has the approved system or software baseline changed? Have the interfaces allocated to each increment changed?

Table 31 – Source File Measure [5]

Description: Lines of code are a well-understood software measure that helps in estimating project cost, required effort, schedule, and productivity. Changes in the number of lines of code indicate development risk due to product size volatility, and possible additional work.

Selection Guidance	*Specification Guidance*
Project Application Used for projects of all sizes. Not usually tracked for COTS software unless changes are made to the source code. **Process Integration** Define lines of code for each language. Lines of code from different languages are not equivalent. It may be necessary to calculate an effective or equivalent SLOC count based on source. New and modified lines would count at 100% while reused code would count at a lower percentage (to address the effort required to integrate and test the reused code). It is sometimes difficult to generate accurate estimates early in the project, especially for new types of projects. Estimates should be updated on a regular basis. Actuals can easily be counted using automated tools. **Usually Applied During** Project Planning (Estimates) Requirements Analysis (Estimates) Design (Estimates) Implementation (Estimates and Actuals) Integration and Test (Actuals) Operations and Maintenance (Actuals)	**Typical Data Items** Count Line Code Count Line Comment Count Line Inactive Count Source File Number of Functions in Source File **Typical Attributes** Increment Source (new, reused, or COTS) Language Delivery status (deliverable, non-deliverable) **Typical Aggregation Structure** Software Configuration Unit / Component **Count Actuals Based On** Release to configuration management Passing unit test Passing inspection **This measure answers questions such as:** How accurate was the project size estimate on which the schedule and effort plans were based? How much has the project size changed? In what components have changes occurred? Has the size allocated to each increment changed?

Table 32 - Requirements Measure [5]

Description: The Requirements measure counts the number of requirements in the system or product specifications. It also counts the number of requirements that are added, modified, or deleted. The measure provides information about the total number of requirements and the development risk due to growth and/or volatility in requirements.

Selection Guidance	*Specification Guidance*
Project Application Applicable to all domains. Useful for any size and type of project that tracks requirements. Effective for both non-developed (COTS/Reuse) and newly developed components. **Process Integration** It is sometimes difficult to specifically define discrete requirements. A consistently applied definition makes this measure more effective. Requires a good requirements traceability process. If an automated design tool is used, the data is more readily available. Count changes against a baseline that is under formal configuration control. Both stated and derived requirements may be included. To evaluate stability, a good definition of the impacts of each change is required. Organize requirements hierarchically (e.g. user requirements lead to system requirements which are decomposed into software, hardware, operations, and maintenance requirements. **Usually Applied During** Project Planning (Estimates) Requirements Analysis (Estimates and Actuals) Design (Actuals) Implementation (Actuals) Integration and Test (Actuals) Operations and Maintenance (Actuals)	**Typical Data Items** Number of requirements (user, system, component, etc.) Number of requirements added Number of requirements deleted Number of requirements modified **Typical Attributes** Increment Change source (supplier, acquirer, user) System component Priority (high, medium, low) Level of requirement (user, system, software) **Typical Aggregation Structure** Function **Typically Collected for Each** Requirement specification **Count Actuals Based On** Passing requirements inspection Release to configuration management **This measure answers questions such as:** Have the requirements allocated to each incremental delivery or increment changed? Are requirements being deferred to later increments? How much has functionality changed? What components have been affected the most? Is the number of requirements growing? If so, at what rate?

4.7. Roles and Responsibilities

There are three tasks that support measurement implementation in an organization. They are obtaining organizational support, defining responsibilities and providing resources.

The organizational support is obtained at Initiation stage of the measurement process life-cycle. At this stage, the roles and responsibilities are defined.

Three important components of a measurement program can be listed as [4]:

1. The source of data: The responsibility of the development and maintenance component is to provide project data. Providing data is the only responsibility imposed on the development and maintenance personnel; they are not responsible for analyzing the data.

2. Analysis and packaging: Analysis and packaging personnel must design and develop the data forms and receive the raw data from the repository. They are responsible for examining project data; producing tailored development and maintenance processes for the specific project. The analysis and packaging personnel are necessarily separate from the development and maintenance personnel because their objectives are significantly different.

3. Technical support: This component provides essential support services including implementing the database as specified by the analysis and packaging component.

Each component must perform its distinct role and responsibilities. In Figure 7, the three important components of a measurement program are shown in detail.

Typical roles and responsibilities in measurement process are usually assigned as [5]:

- Executive manager: The executive manager is generally an organizational or enterprise manager responsible for multiple projects. He/She uses measurement results to make organizational and enterprise level decisions.
- Project or technical manager: He/She is responsible for identifying issues, reviewing analysis results, and acting on measurement information.
- Measurement analyst: This role can be assigned to either an individual or a team. Developing the project measurement plan, collecting and analyzing measurement data, and reporting results are responsibilities of analyst.
- Project team: This is the team of project personnel responsible for development and maintenance of software and system projects.

The team is source of measurement data and uses the measurement results to guide engineering activities.

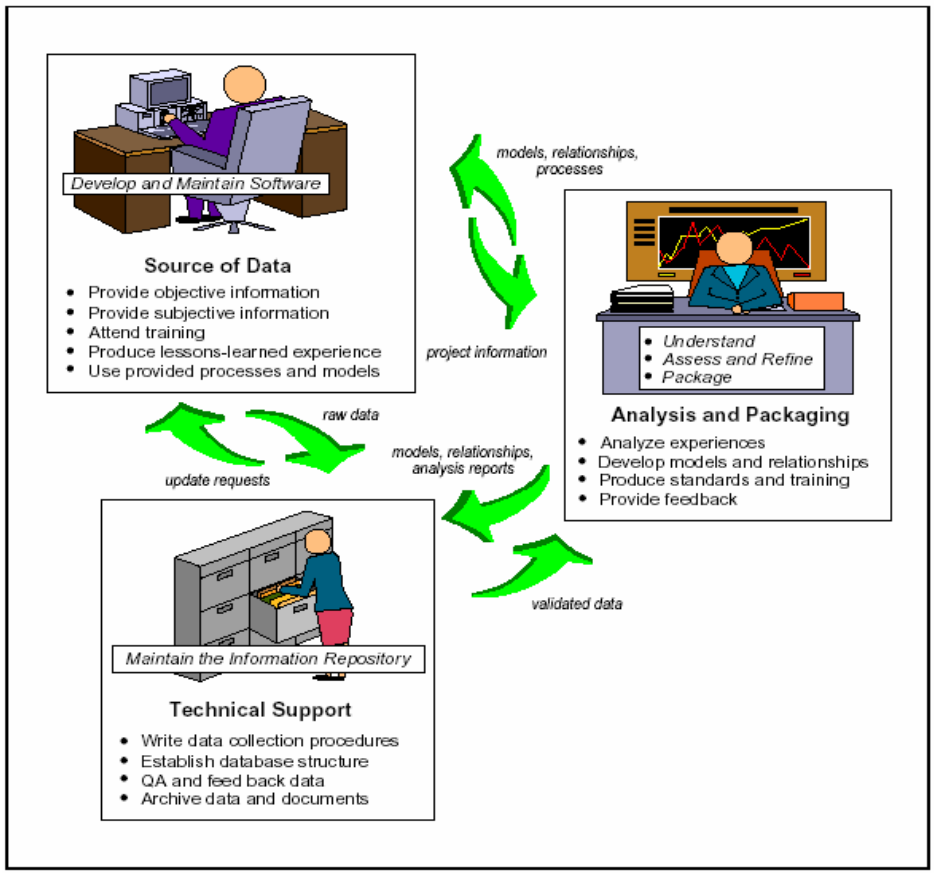


Figure 7 – The Three Components of a Measurement Program [4]

4.8. Tailoring

New issue areas, categories, and measures may be defined during the tailoring activity. As an organization gains experience in implementing measurement, it may update the I-C-M table [5].

The need for a new common issue area typically becomes apparent during the tailoring phase when a project specific issue cannot be mapped to a PSM common issue area. Also, the need for a new measure, or an entirely new

category of measures, might arise when none of the candidate measures are appropriate for target development environment or existing measures do not provide the insight needed to address an issue [5].

As new elements are proposed, it is recommended that complete issue descriptions and full category and measurement tables be constructed. This level of definition clarifies why, what, and how data is being measured and provides the information needed to effectively implement measurement collection and reporting.

In section 4.7, the measurement Table 14, 15, 20, 21, 25, 26 and 31 were formed after tailoring activities. The others were reviewed carefully, and small modifications in tables such as adding some new items, deleting unreachable and non-existent attributes, changing typical level to SCU (software configuration unit) were made when necessary.

CHAPTER 5

BUILD STAGE

5.1. Introduction to Build Stage

The answer of how the measurement process can be effectively applied appears in the "Measurement Plan" which is the output of the Build Stage. The measurement process can be integrated with the technical and managerial process according to measurement plans.

Figure 8 shows the evolution of an information need (project issues) with a measurement plan. [8]

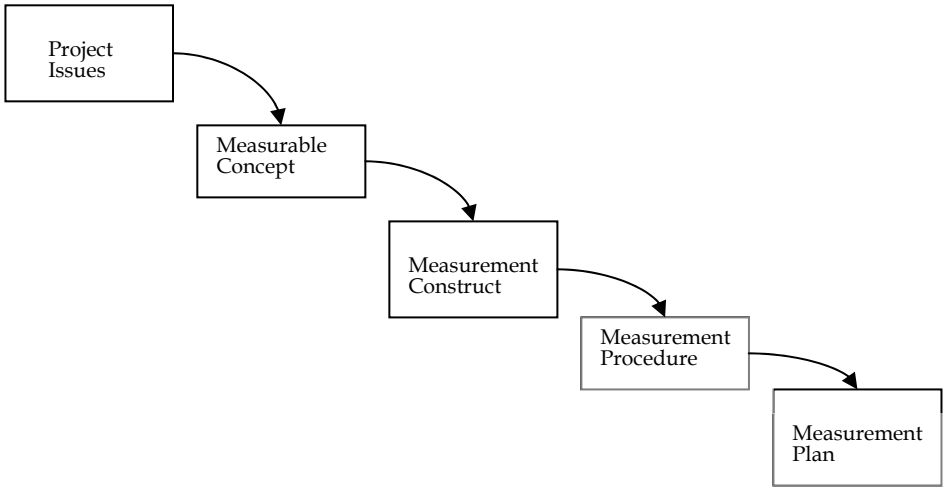


Figure 8 - Evolution of Project Issues

Defining of the information needs is starting point of measurement planning. The measurable concept is an idea about entities that should be measured in order to satisfy an information need. The measurable concept can be formalized as a measurement construct that specifies exactly what will be measured and how data will be combined to produce results. A measurement procedure defines the mechanics of collecting and reporting. The sub-tasks of the Build stage are shown in Figure 9.

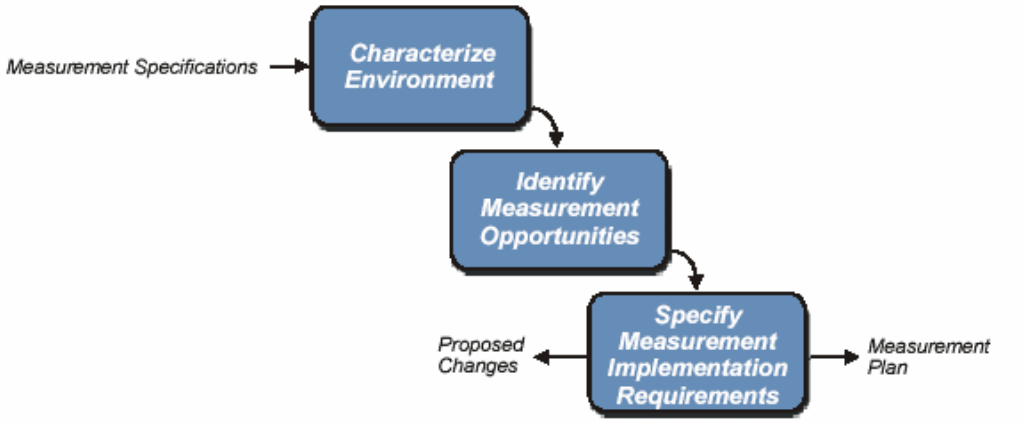


Figure 9 - Sub-Tasks of Build Stage [5]

The inputs of this stage are measurement specifications and the outputs are the Measurement Plan. At first, the measurement environment will be characterized. Then measurement opportunities will be identified. Lastly, the measurement implementation requirements will be specified.

5.1.1. Characterize Environment

In ISO 15939, one of the activities in measurement process is "Characterize Organizational Unit" [1]. Significant aspects that characterize an organizational unit are [5]:

- The life-cycle model used,
- Current measurement activities employed,
- System and software technology, including design techniques, software programming languages, and tools used,
- Planned sources of software components (i.e. COTS, reused) ,
- Management, review, test, and inspection practices,
- Engineering and management standards to be applied,
- Process maturity of the organizations, and
- Project organization and teaming structure.

In ASELSAN Inc., the MST division projects' typical properties, which are defined by members of the PAT-G team, are

- Contractual Projects,
- Project End Time and Price defined,
- Military and Professional System Softwares,
- Project includes new technology intensively,
- At least 1.5 years development times,
- Variety at application areas, and
- Integration software and hardware.

5.1.2. Identify Measurement Opportunities

Measurement data comes from many sources such as forms, databases, and tools. Extracting data from electronic sources is usually more cost effective than manual collection methods. Especially CASE tools used actively in organization are very suitable source of data items.

Three primary forms of data are [5]

- Historical Data – This form of data is collected from past projects in order to help in estimating and in determining the feasibility of plans.
- Planning Data – This form of data must be collected from all plans that include incremental changes to plans.
- Actual Performance Data – While a project evolves, actual data will become available. Many sources of data exist within the life-cycle process.

The configuration management tool ClearCASE, the defect and problem-tracking tool ClearDDTS, the project management tool MS Project, effort record tool Iscilik Bilgi Sistemi, and Change Request Tracking System are some of the tools that are used actively in MST Division of ASELSAN Inc. Other sources are usually at document form. In order to obtain data from documents systematically, some tools are intended to be developed in the scope of the thesis. These data sources are combined within a table shown in Table 33.

Table 33 – Data Sources in MST Division

Measurement Category	Electronic Source	Document Source
Milestone Performance	MS Project	SDP
Work Unit Progress	MS Project / Configuration Management System (ClearCASE)	Status Report
Functional Correctness	Defect/Problem Tracking System (ClearDDTS)/ Configuration Management System (ClearCASE) / Case Tools / Test Automation Tools	Review/Inspection Reports / Design Review Notes and Actions / Test Reports
Supportability and Maintainability	Analysis Tool	
Personnel	Information System (İscilik Bilgi Sistemi)	
Physical Size and Stability	Analysis Tool / Configuration Management System (ClearCASE)	
Functional Size and Stability	Change Request Tracking System / Configuration Management System / CASE Tools	Requirements and Design Specifications / Change Request

5.1.3. Specify Measurement Implementation Requirements

This step involves developing a combination of operational definitions and procedures that guide the application of measurement activity. At this stage, the issues are [5]

- Measurement Definitions: The definition of selected measure is given in specification table at the previous design stage.

- Measurement Scope: For each selected measure, the life cycle phases or activities should be described.
- Data Collection: This includes defining the measurement source, responsibility for conducting the measurement, and periodicity of data collection, as well as the tools, forms, and databases used to collect and store the data.
- Data Analysis: The basic indicators to be generated from measures should be defined and the process for generating and analyzing each indicator should be described. This includes defining the periodicity and responsibility for conducting the analysis. However, serious experience in measurement programs is prerequisite to determine indicators in details, so new indicators can be added at following phases of the measurement program.
- Result Reporting: The process for reporting analysis results should be described. This includes selecting the analyses to be reported, responsibility for preparing the reports, format, and periodicity of reporting.
- Measurement Evaluation: The measurement process and measures need to be evaluated periodically.

5.2. Measurement Plan for ASELSAN's System41 Project

5.2.1. Introduction

The software measurement program is intended to monitor and control software development process in this project that has been recently started in ASELSAN-MST called System41.

5.2.2. Project Description

Confidential.

5.2.3. Measurement Roles and Responsibilities

- Executive manager: L. A.
- Software Development Team Leader: H. K.
- Measurement analyst: Ö. E.
- Project team: ASELSAN - MST

5.2.4. Description of Project Issues

The prioritized lists of goals and related issues are defined in analysis stage of software measurement program. They are valid for the System41 project and shown in Table 34 and Table 35 below.

Table 34 – Common and Related Issues

#	COMMON ISSUE	RELATED ISSUES
1	Schedule and Progress	- The risk of the intensive project schedule. - The lack of information about whether project going on schedule or not. - The lack of information about whether scheduled milestones meeting or not.
2	Product Quality	- The lack of information about whether software product ready to delivery or not. - The lack of information about whether all identified problems resolved or not. - The lack of information about how much difficult the software is to maintain.
3	Resources and Cost	- The risk of staff experience. - The lack of information about whether staff effort is adequate or not. - The lack of information about whether the number of staff is adequate or not. - The risk of constant budget.
4	Product Size and Stability	- The risk of unstable requirements. - The lack of information about how many the requirements are changing. - The lack of information about how much the product's size is changing.

Table 35 – Prioritized Goals

Priority	GOAL	Priority	Common Issue
1	Track and analyze the schedule to improve and minimize it from the viewpoint of development team leader.	5,4	Schedule and Progress
2	Analyze the product and its functionality to improve software performance.	4,2	Product Quality
3	Analyze the development cost in order to minimize it.	4,1	Resources and Cost
4	Evaluate and analyze the productivity to improve it from the viewpoint of department headmaster.	4,1	Resources and Cost
5	Collect and analyze required data to make software estimation.	3,2	Product Size and Stability

In the following Figure 10, an overview of the measurement process is shown in detail.

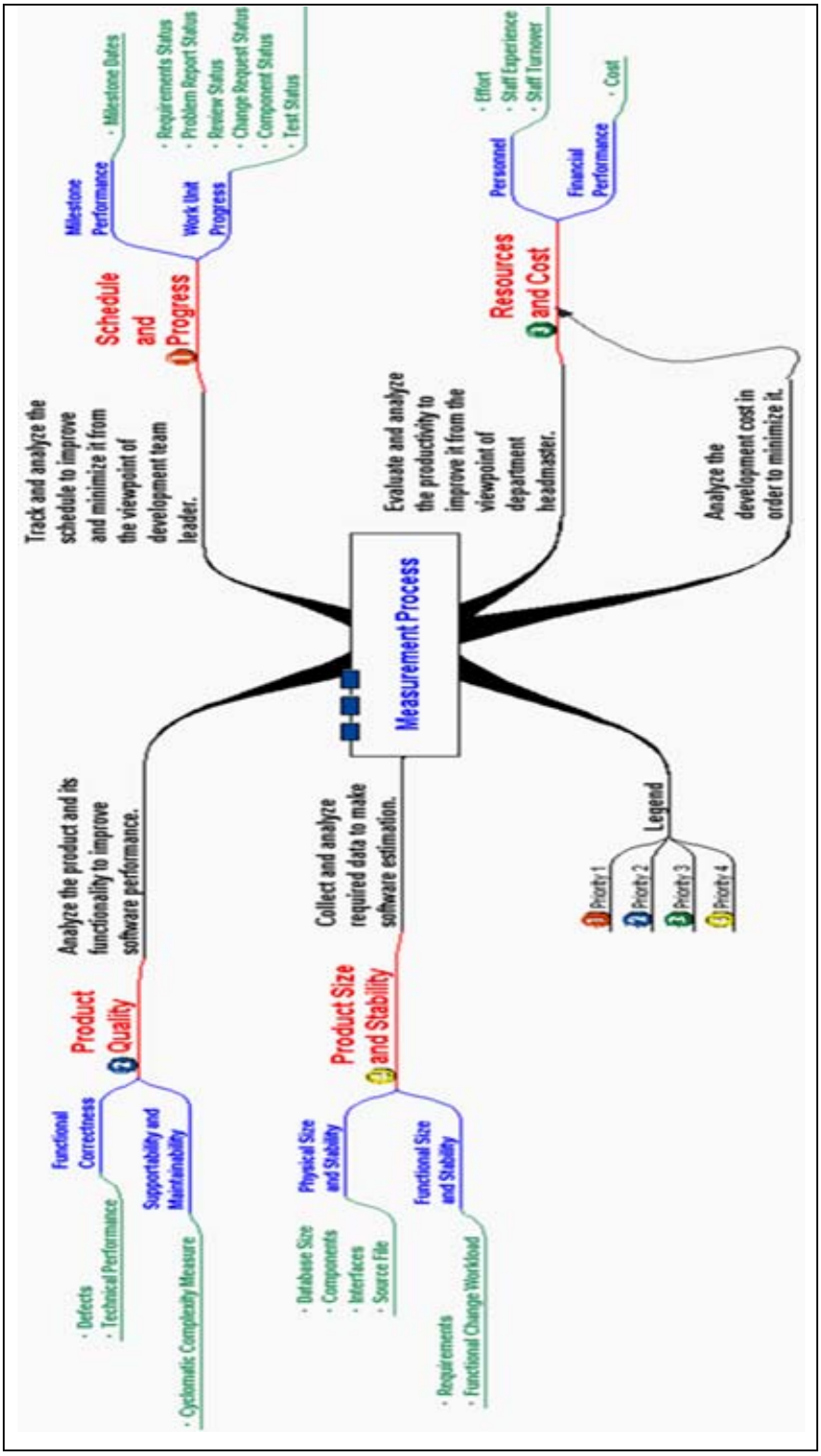


Figure 10 – An Overview of the Measurement Process

5.2.5. Measurement Specifications

Specifications of the measurements to be applied in ASELSAN are given in tables, 36 through 53.

Table 36 – Milestone Dates Specification

MEASURE	**Milestone Dates** Category: Milestone Performance Issue: Schedule and Progress
Data Items	Start date of activity or event End date of activity or event
Attributes	Activity or event name Version of the plan Increment Organization
Aggregation Structure	Component Activity
Definition	Milestone Dates measures the start and end dates for activities, events, and products. The measure provides an easy-to-understand view of scheduled activities and events. Comparison of plan and actual milestone dates provides insight into significant schedule changes.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Successful completion of tasks Documents base lined Milestone review held
Applied During	Project Planning, Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from Project Management tool (MSProject) and can be collected and reported manually.
Periodicity	Monthly

Table 37 – Requirements Status Specification

MEASURE	**Requirements Status** Category: Work Unit Progress Issue: Schedule and Progress
Data Items	Total number of requirements Number of requirements traced to detailed specifications Number of requirements traced to test specifications Number of requirements tested successfully
Attributes	Increment Specification reference Test sequence reference
Aggregation Structure	Function
Definition	The measure is an indication of product design and test progress. When used to measure test status, the measure is used to evaluate whether required functionality has been successfully demonstrated in the specified requirements, and the amount of testing that has been performed.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Completion of specification review Baselining of specifications Baselining of requirements traceability matrix Successful completion of all tests in the appropriate test sequence
Applied During	Requirement Analysis, Design, Implementation, Integration and Test.
Data Reporting Process	Data is available from SRS, SDD, and Test Reports. A tool is intended to develop in order to make data collection and reporting systematically.
Periodicity	Monthly

Table 38 – Problem Report Status Specification

MEASURE	**Problem Report Status** Category: Work Unit Progress Issue: Schedule and Progress
Data Items	Number of problem reported Number of problem resolved Average age of problems Average time between assignment and resolving Average time between submission and opening
Attributes	Increment
Aggregation Structure	Component
Definition	Problem Report Status measure provides an indication of product maturity and readiness for delivery. The rates at which problem reports are written and resolved can be used to estimate product completion. This measure can also indicate the quality of the problem resolution process, based on the average age of reported problems and the average time to resolve them.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Problem report reported Problem report implemented Problem report integrated Problem report tested
Applied During	Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from ClearDDTS and its reports. A tool is intended to develop in order to make data collection systematically. (YazOlc-YARDIM tool)
Periodicity	Monthly (Requirement Analysis, Design, Implementation Stages) Two weekly (Integration and Test, Operations and Maintenance)

Table 39 – Review Status Specification

MEASURE	**Review Status** Category: Work Unit Progress Issue: Schedule and Progress
Data Items	Number of reviews Number of reviews completed successfully Number of “important” items Number of "minor" items Number of “incomprehensible” items Number of “total” items Number of items which are not agreed on at meeting.
Attributes	Name of the component being reviewed Increment
Aggregation Structure	Component
Definition	The measure provides an indication of progress in completing review activities. The Review Status measure also counts the number of types of review items determined during the review process. The relationship between total identified numbers in review and total page number of reviewed software product can be established by using the results of this measure.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Completion of review
Applied During	Requirement Analysis, Design, Implementation, Integration and Test.
Data Reporting Process	Data is available from the review reports. A tool is intended to develop in order to make data collection systematically. (YazOlc-YARDIM tool)
Periodicity	Monthly

Table 40 – Change Request Status Specification

MEASURE	**Change Request Status** Category: Work Unit Progress Issue: Schedule and Progress
Data Items	Number of change requests generated Number of change requests resolved
Attributes	Increment Priority Change classification (defect correction, enhancement) Valid/Invalid
Aggregation Structure	Function Component
Definition	The Change Request Status measure counts the total number of change requests that affect a product. The measure provides an indication of the amount of rework that has been performed or will be required. This measure only identifies the number of changes; it does not report on the functional impact of changes or the amount of effort required to implement them.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Change Request Approval Change Request Implemented Change Request Integrated Change Request Tested
Applied During	Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from the Change Request Tracking system and ClearDDTS.
Periodicity	Monthly

Table 41 – Component Status Specification

MEASURE	**Component Status** Category: Work Unit Progress Issue: Schedule and Progress
Data Items	Total number of components Number of components completed successfully
Attributes	Increment
Aggregation Structure	Component
Definition	A comparison of plans and actual helps assess the status of development progress. Early in the development activity, planning changes should be expected. Later in the process, an increase in the planned number of components that are scheduled for a specific activity may indicate unplanned or excessive growth.
Collection Level	Project
Count Actual Based On	Completion of component reviews, inspections, or walkthroughs Successful completion of specified test Release to configuration management
Applied During	Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from Configuration Management System, and can be collected manually.
Periodicity	Monthly (Requirement Analysis, Design, Operations and Maintenance Stages) Two weekly (Implementation, Integration and Test)

Table 42 – Test Status Specification

MEASURE	**Test Status** Category: Work Unit Progress Issue: Schedule and Progress
Data Items	Total number of test cases Number of test cases attempted Number of test cases passed
Attributes	Increment Test environment Test configuration
Aggregation Structure	Component
Definition	The Test Status measure counts the number of test cases that have been attempted and the number that have been completed successfully. This measure can be used with the Requirement Status measure to evaluate test progress. This measure helps assess product quality based on the proportion of attempted test cases that have been successfully executed.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Successful completion of each test case
Applied During	Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from Test Reports prepared by V&V department, and will be collected manually.
Periodicity	Monthly

Table 43 – Defects Specification

MEASURE	**Defects** Category: Functional Correctness Issue: Product Quality
Data Items	Defect Statistics
Attributes	Increment Defect Status Defect Severity Defect Category When Found How Found When Fixed How Resolved
Aggregation Structure	Component
Definition	The Defects measure provides useful information on the ability of a supplier to find and fix defects in hardware, software or documentation. The number of defects indicates the amount of rework, and has a direct impact on quality. Arrival rates can indicate product maturity. Closure rates can be used to predict test completion. A Defect Density measure, which is an expression of the number of defects in a quantity of product, can be derived from this measure. Defect Density can identify components with the highest concentration of defects.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Defects accepted by configuration control Defects validated Defect correction successfully tested/inspected Defect assessment of readiness for delivery to a field
Applied During	Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from ClearDDTS and its reports. (YazOlc-YARDIM tool)
Periodicity	Monthly (Requirement Analysis, Design, Implementation Stages) Two weekly (Integration and Test, Operations and Maintenance)

Table 44 – Technical Performance Specification

MEASURE	**Technical Performance** Category: Functional Correctness Issue: Product Quality
Data Items	Datum Interface Speed Block Processing Speed
Attributes	Increment
Aggregation Structure	Component
Definition	The Technical Performance measures address any functional characteristics that can be quantitatively defined and demonstrated. Various types of functional requirements may be measured including user and mission functions, security features, accuracy of the system component functions, response time, data handling capability, or signal processing. These measures provide an indication of the overall ability of a system to meet the users' functional requirements.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Passing functional test
Applied During	Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from test reports, and will be collected manually.
Periodicity	Monthly

Table 45 – Cyclomatic Complexity Specification

MEASURE	**Cyclomatic Complexity** Category: Supportability and Maintainability Issue: Product Quality
Data Items	Complexity Value
Attributes	Increment
Aggregation Structure	Component Source (new, reused, or COTS) Language Delivery status (deliverable, non-deliverable)
Definition	The concept of Cyclomatic Complexity also can be used to evaluate the complexity of control or information flow in a system. This measure provides an indication of both design quality and the amount of testing required. A high complexity rating is often a leading indicator of a high defect rate. Components with high complexity usually require additional reviews, increased, testing, or rewriting.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Passing inspection Passing component test Release to configuration management
Applied During	Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available by using Understand For C++ tool and its report. A tool is intended to develop in order to report result systematically. (YazOlc-YARDIM tool)
Periodicity	Two weekly (Implementation Stage) Monthly (Integration and Test, Operations and Maintenance)

Table 46 – Effort Specification

MEASURE	**Effort** Category: Personnel Issue: Resource and Cost
Data Items	Number of labor hours (days, months, etc.) Number of personnel
Attributes	Labor category Increment
Aggregation Structure	Component/Activity
Definition	The Effort measure counts the number of labor hours or number of personnel applied to all tasks. This is a straightforward, easily understood measure. This measure usually correlates directly with cost, but can also address other common issue areas including Schedule and Progress, and Process Performance.
Collection Level	Project
Count Actual Based On	Financial reporting criteria
Applied During	Project Planning, Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from İscilik Bildirim Sistemi and its report.
Periodicity	Monthly

Table 47 – Staff Experience Specification

MEASURE	**Staff Experience** Category: Personnel Issue: Resource and Cost
Data Items	Number of personnel Number of years of experience
Attributes	Branch (GUI, Control, DSP)
Aggregation Structure	Activity
Definition	The Staff Experience measure counts the total number of experienced personnel in defined areas. The measure determines whether sufficient experienced personnel are available. The experience factors are based on the requirements of each individual project, such as environment or application. Experience is usually measured in years.
Collection Level	Project
Count Actual Based On	Staff changes
Applied During	Project Planning, Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from managerial reports.
Periodicity	Monthly

Table 48 – Staff Turnover Specification

MEASURE	**Staff Turnover** Category: Personnel Issue: Resource and Cost
Data Items	Number of personnel Number of personnel gained Number of personnel lost
Attributes	Branch (GUI, Control, DSP) Sex (Male/Female) Degree (MSc., PHD, …) School (METU, BU, HU)
Aggregation Structure	Activity
Definition	The Staff Turnover measure counts staff losses and gains. This measure is most effective when used in conjunction with the Staff Experience measure. Loss of key and experienced personnel is critical.
Collection Level	Project
Count Actual Based On	Financial reporting criteria Organization restructuring or new organizational charts
Applied During	Project Planning, Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from managerial reports.
Periodicity	Monthly

Table 49 – Database Size Specification

MEASURE	**Database Size** Category: Physical Size and Stability Issue: Product Size and Stability
Data Items	Number of tables Number of records or entries Number of words or bytes
Attributes	Increment Database identifier
Aggregation Structure	Component
Definition	The Database Size measure counts the number of words, records, or tables in each database. The measure indicates how much data must be handled by the system.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Schema design released to configuration management Schema implementation released to configuration management
Applied During	Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from SRS, SDD and source code and will be collected manually or by using development tool.
Periodicity	Two weekly (Implementation Stage) Monthly (Requirement Analysis, Design, Integration and Test, Operations and Maintenance)

Table 50 – Components Specification

MEASURE	**Components** Category: Physical Size and Stability Issue: Product Size and Stability
Data Items	Number of units Number of units added Number of units deleted Number of units modified
Attributes	Increment Source (new, reused, or COTS) Language Delivery status (deliverable, non-deliverable) End-use environment (operational, support)
Aggregation Structure	Component
Definition	The Components measure counts the number of elementary components in a system or product, and the number that are added, modified, or deleted. The total number of components defines the size of the system. Changes in the number of estimated and actual components indicate risk due to product size volatility and additional work that may be required.
Collection Level	Project
Count Actual Based On	Release to configuration management Passing unit test Passing inspection
Applied During	Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from Configuration Management Reports.
Periodicity	Monthly

Table 51 – Interfaces Specification

MEASURE	**Interfaces** Category: Physical Size and Stability Issue: Product Size and Stability
Data Items	Number of interfaces Number of interfaces added Number of interfaces deleted Number of interfaces modified
Attributes	Increment Component boundary Nature of interface (e.g. data, control signals, mechanical action)
Aggregation Structure	Component
Definition	The Interfaces measure is particularly useful when allocating functions during architecture development, to quantify the number of pair-wise relationships between components. This measure also counts the number of interfaces that are added, modified, or deleted. Changes in the number of estimated and actual interfaces indicate risk due to requirements, architectural, or design volatility.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Release to configuration management Passing an integration test
Applied During	Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from SIDD and can be collected manually or by using software development tool.
Periodicity	Monthly

Table 52 – Source File Specification

MEASURE	**Source File** Category: Physical Size and Stability Issue: Product Size and Stability
Data Items	Count Line Code Count Line Comment Count Line Inactive Count Source File Number of Functions in Source File
Attributes	Increment Source (new, reused, or COTS) Language Delivery status (deliverable, non-deliverable)
Aggregation Structure	Software Configuration Unit / Component
Definition	The Source File helps in estimating project cost, required effort, schedule, and productivity. Changes in the number of data indicate development risk due to product size volatility, and possible additional work.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Release to configuration management Passing unit test Passing inspection
Applied During	Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from source code and can be collected by using software development tool or Understand For C++ tool. A tool is intended to develop in order to report result systematically. (YazOlc-YARDIM tool)
Periodicity	Two weekly (Implementation Stage) Monthly (Requirement Analysis, Design, Integration and Test, Operations and Maintenance)

Table 53 – Requirements Specification

MEASURE	**Requirements** Category: Functional Size and Stability Issue: Product Size and Stability
Data Items	Number of requirements (user, system, component, etc.) Number of requirements added Number of requirements deleted Number of requirements modified
Attributes	Increment Change source (supplier, acquirer, user) System component Priority (high, medium, low) Level of requirement (user, system, software)
Aggregation Structure	Function
Definition	The Requirements measure counts the number of requirements in the system or product specifications. It also counts the number of requirements that are added, modified, or deleted. The measure provides information about the total number of requirements and the development risk due to growth and/or volatility in requirements.
Collection Level	SCU (Software Configuration Unit)
Count Actual Based On	Passing requirements inspection Release to configuration management
Applied During	Requirement Analysis, Design, Implementation, Integration and Test, Operations and Maintenance.
Data Reporting Process	Data is available from SRS and will be collected manually or by using development tool.
Periodicity	Monthly

5.2.6. Reporting Mechanisms and Periodicity

The report that includes results of applied measurements in ASELSAN will be prepared monthly. The period can be shortened if needed.

CHAPTER 6

IMPLEMENTATION STAGE

6.1. User's Guide for the YazOlc-Yardim Tool

Since there is no way to collect data automatically for some selected measurements, the need for an auxiliary tool appears in the applied measurement program. The YazOlc-Yardim is designed and developed with in the scope of this study in order to collect data and report analysis measures related to Defects, Problem Report Status, Review Status, Source File, and Complexity Measurements.

The YazOlc-Yardim has to be inter-operable with other tools, namely, Rational ClearDDTS (version 4.5.1) and Understand for C++ (version 1.4). In other words, their outputs constitute the inputs of YazOlc-Yardim tool.

The tool has been put to use in MST Division of ASELSAN Inc. The user interfaces are designed in Turkish. The users' Guidelines document of "YazOlc-Yardim" tool is written in Turkish.

For more details, please see APPENDIX A.

6.2. Historical Data Collection

Defects, Problem Report Status, Source File, Complexity, and Review Status measurements are applied at the organization in order to collect historical data about these measures. The measurement specification tables defined at Design stage of the program have been implemented. These measures were applied at 8 projects in MST Division of ASELSAN Inc. The historical data and measurement results are reported by using YazOlc-Yardim tool. The some of these reports and overview of measurements are reviewed below.

6.2.1. Reports of Problem Report Status Measurement

These reports include:

- The life time of all problems (from submit to resolve),
- The open time of all problems (from submit to open), and
- The resolve time of all problems (from assign to resolve).

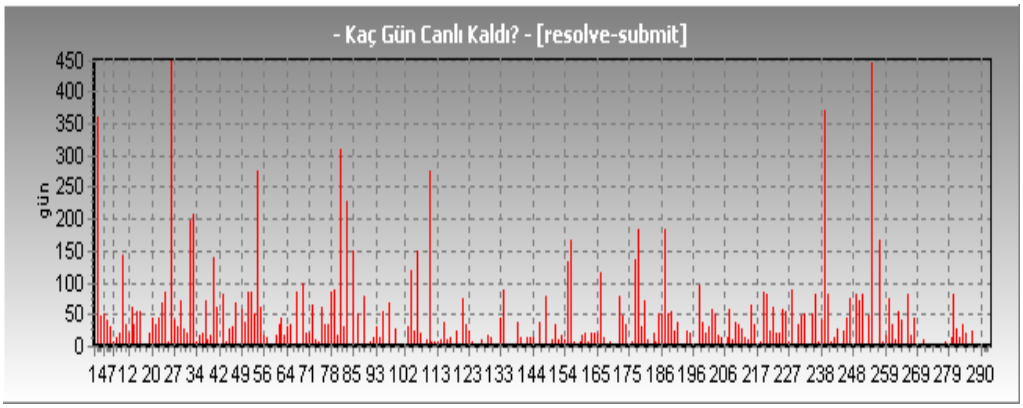
The average values are also calculated and presented within these reports.

The report produced after the measurement was applied in System11 project is shown in Figure 11.

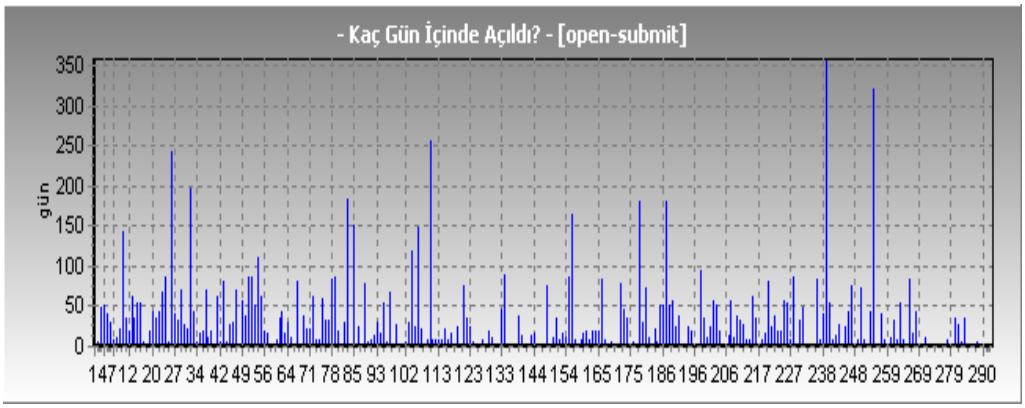
The System11 project is still in progress, so the measurement results show snapshot data taken within software life-cycle.

Project No - System11

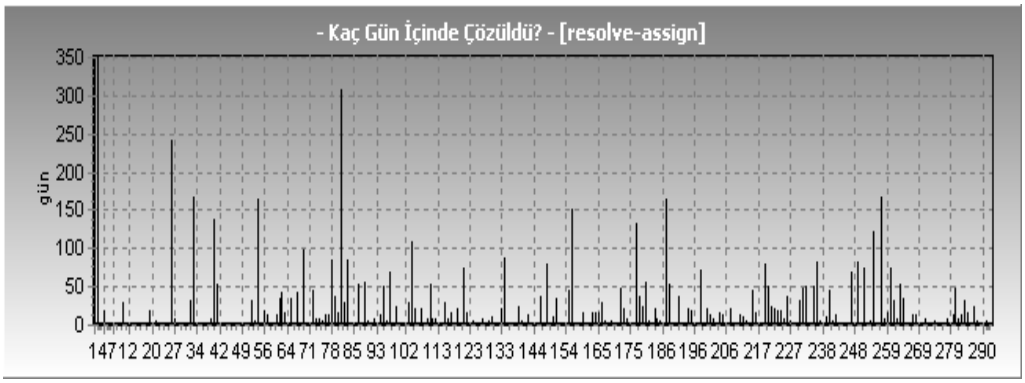
Measurement Date: November, 2003



Average Life Time of Problem [resolve - submit] = 41 days



Average Opening Time [open - submit] = 32 days



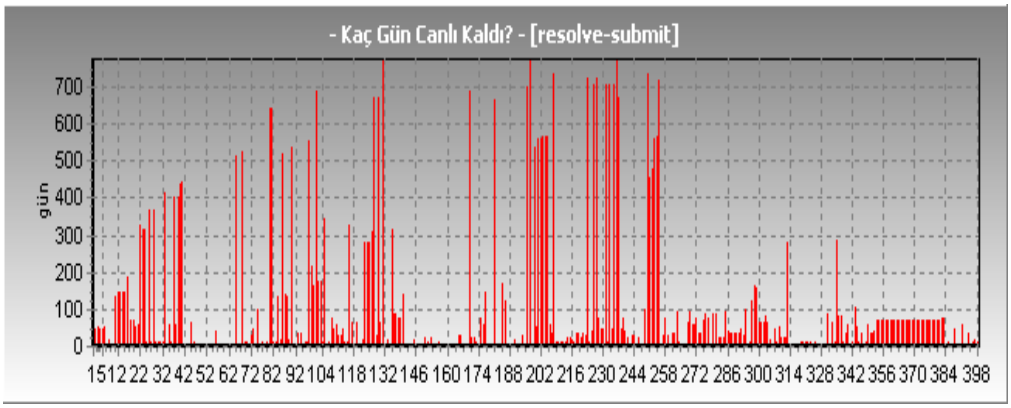
Average Resolve Time [resolve - assign] = 20 days

Figure 11 - System11 Measurement Result

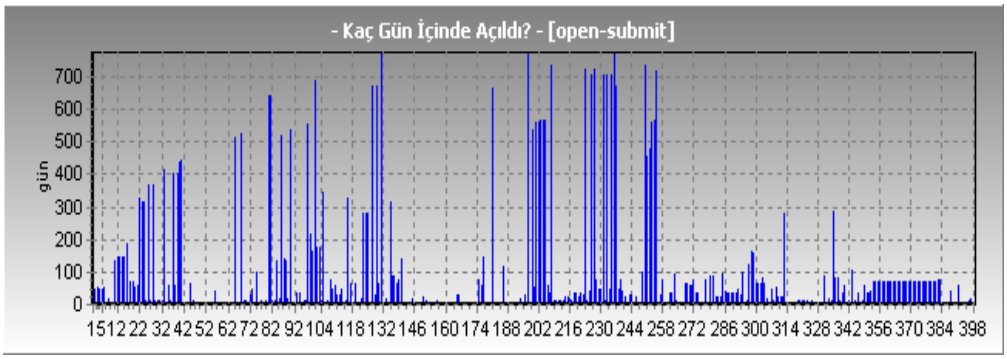
The next report produced after measurement was applied in System60 project is shown in Figure 12. The System60 project is still in progress, so the measurement results show snapshot data taken within software life-cycle.

Project No - System60

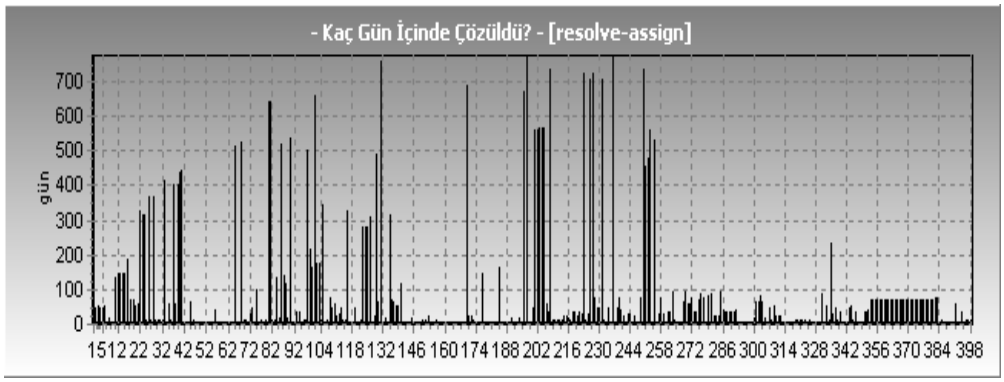
Measurement Date: December, 2003



Average Life Time of Problem [resolve - submit] = 105 days



Average Opening Time [open - submit] = 99 days



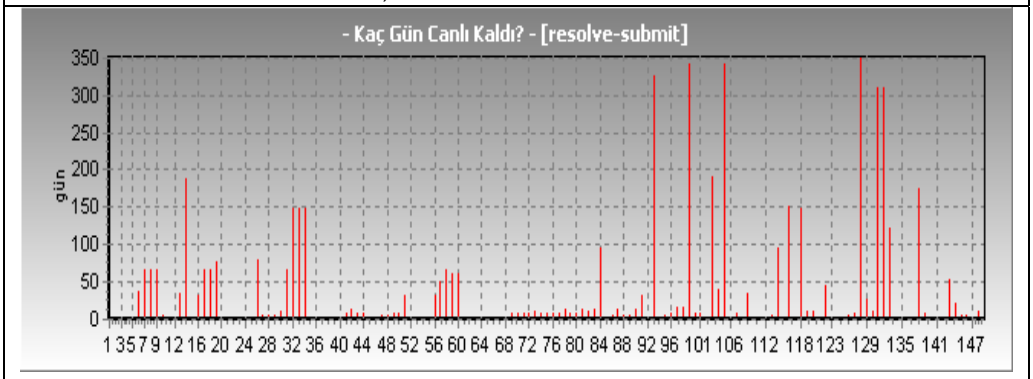
Average Resolve Time [resolve - assign] = 87 days

Figure 12 – System60 Measurement Result

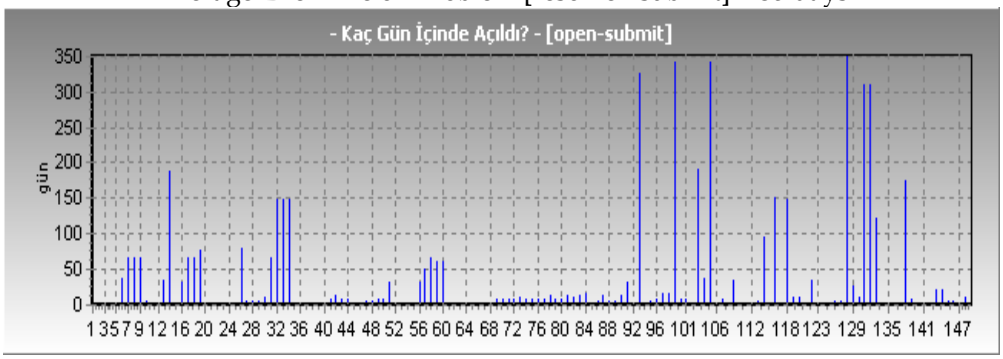
The report produced after measurement was applied in System20 project is shown in Figure 13. This project is still in progress, and the measurement results show snapshot data taken within software life-cycle.

Project No - System20

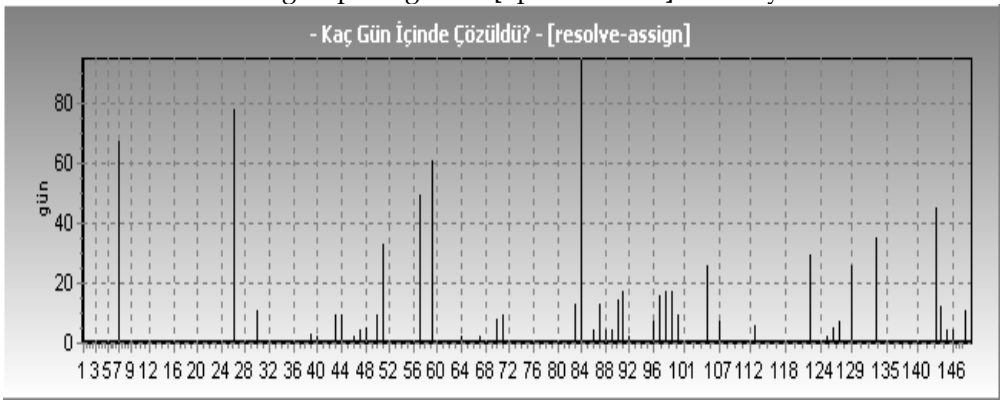
Measurement Date: December, 2003



Average Life Time of Problem [resolve - submit] = 35 days



Average Opening Time [open - submit] = 34 days

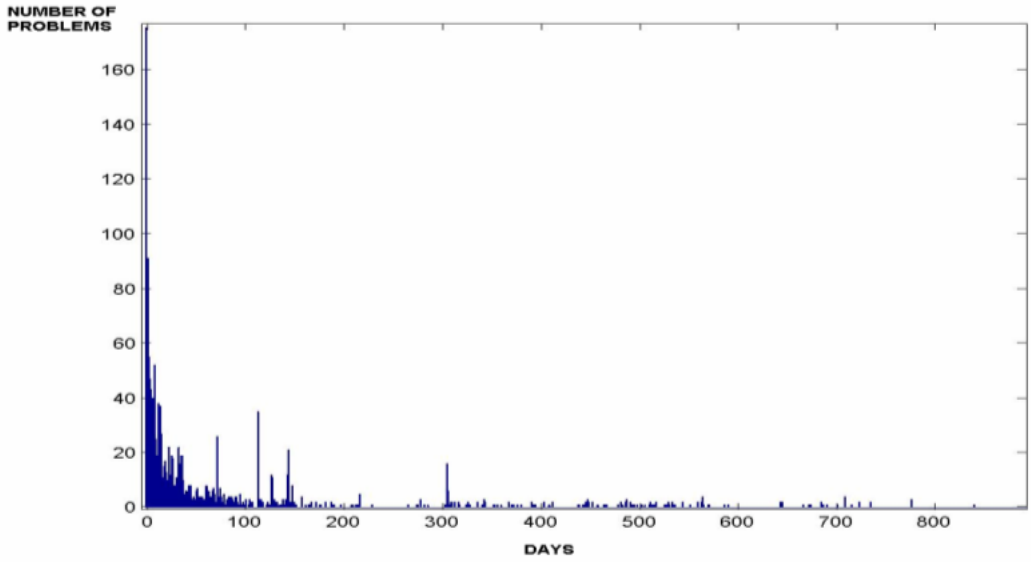


Average Resolve Time [resolve - assign] = 5 days

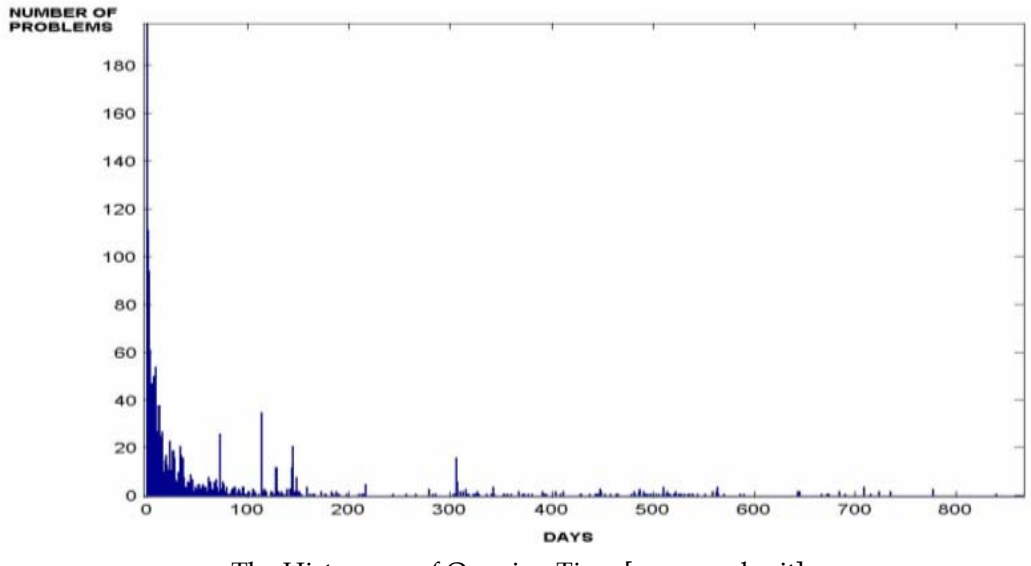
Figure 13 – System20 Measurement Result

6.2.2. Overview of the Problem Report Status Measurement

The projects for which problem report status measurement has been applied are System20, System60, System37, System38, System12, System90, System96, and System11. Figure 14 shows the histograms of the problem life time, opening time and resolve time.

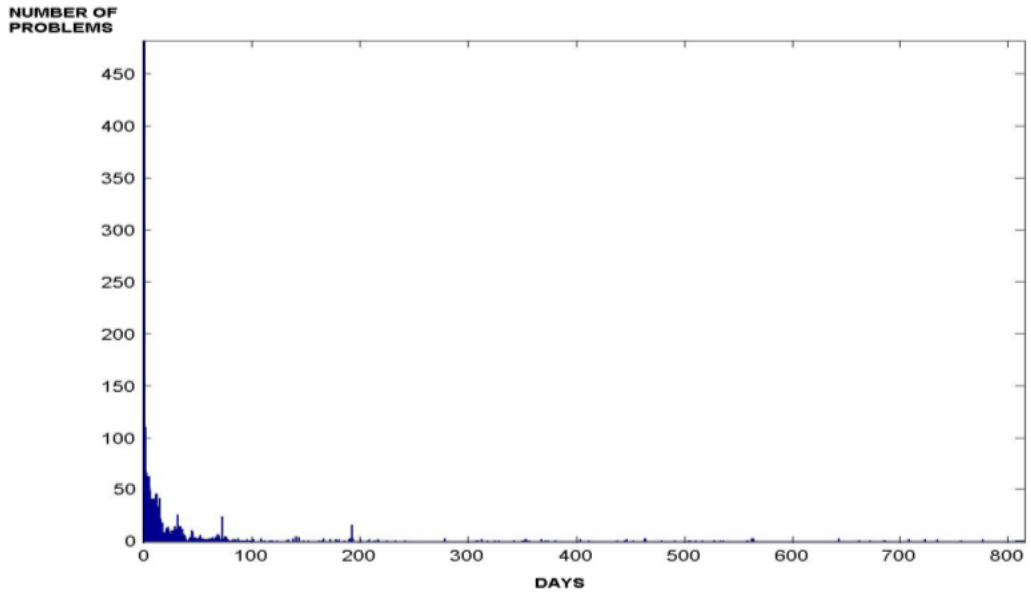


The Histogram of Problem Life Time [resolve - submit]



The Histogram of Opening Time [open - submit]

Figure 14 continued.



The Histogram of Resolve Time [resolve - assign]

Figure 14 – Histograms of All Measurement Results

The following mean values are obtained over these eight projects:

- Life Time of Problem [resolve - submit] is 63.2 days.
- Opening Time [open - submit] is 59.6 days.
- Resolve Time [resolve - assign] is 35.1 days.

These measurement results are discussed in a PAT-G meeting in ASELSAN Inc. As a result, improvements in the structure of evaluation meeting where the submitted items are discussed by development team members and usage of ClearDDTS tool within organization are decided. The decisions are summarized below.

- The evaluation meeting should be done more frequently. It should not have a period longer than two weeks, especially in later phases of implementation phase of software life cycles.

- When an item which has high severity is submitted, the evaluation meeting should be done immediately, possibly using electronic mail facilities.
- The exact harmony between ClearDDTS tool and software development process in organization should be constructed in order to provide the most effective usage of ClearDDTS tool.
- The collective usage of configuration management tool ClearCASE and ClearDDTS should be provided.
- A state between "submitted" and "assigned" called "not agreed in meeting" should be added in order to track the item closely.
- A directive about the more effective usage of ClearDDTS tool should be prepared and published within the organization.

6.2.3. Reports of Defects Measurement

These reports include:

- The actual states of defects,
- The actual severity levels of defects,
- The information about how defects are found,
- The information about how defects are resolved,
- The information about when defects are found, and
- The information about when defects are resolved.

In order to achieve the measurement data, the ClearDDTS records and YazOlc-YARDIM tool were used.

The report produced after measurement was applied in System20 project is shown in Figure 15. The System20 project is still in progress, and the measurement results show snapshot data taken within software life-cycle.

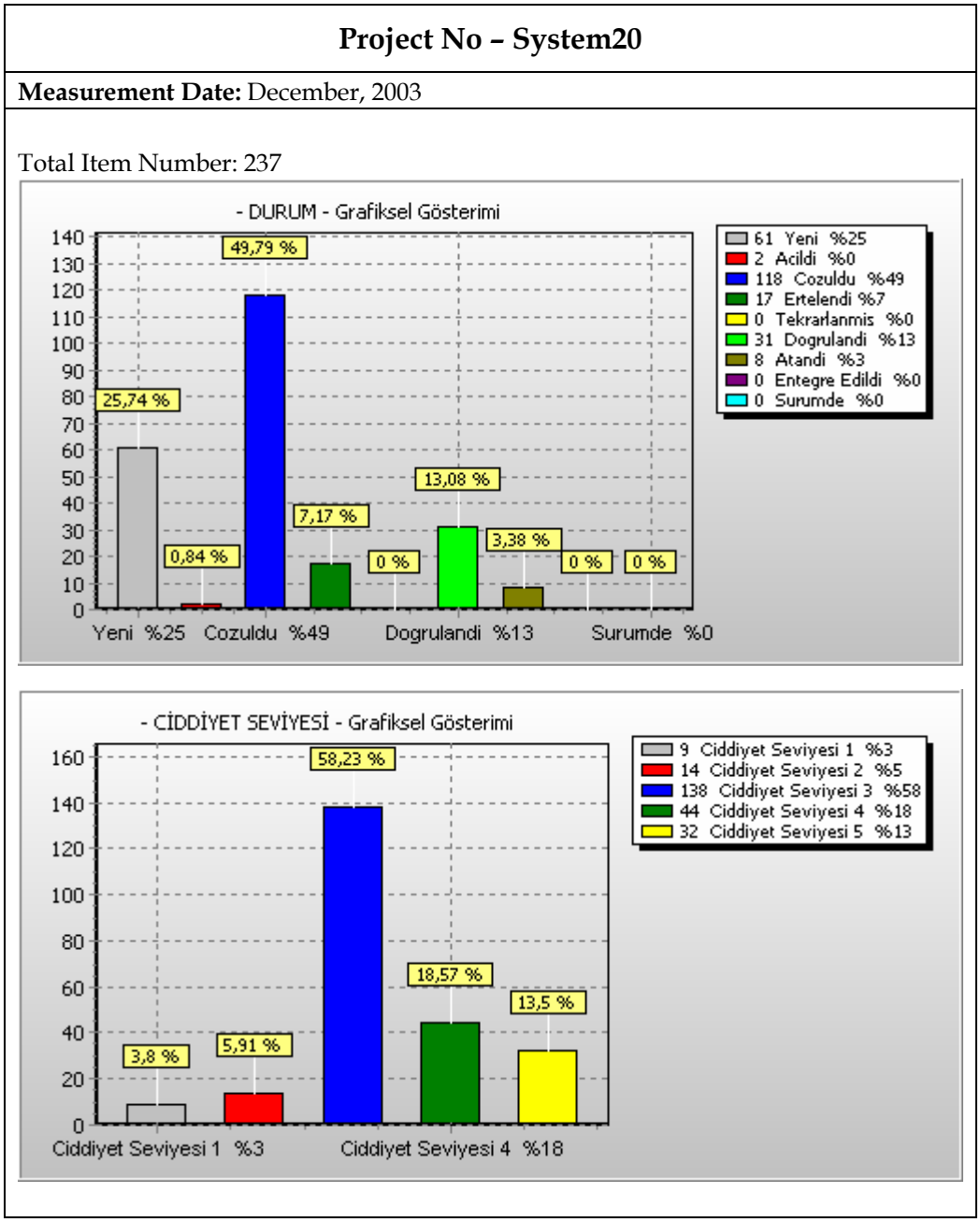


Figure 15 – System20 Defects Measurement Result

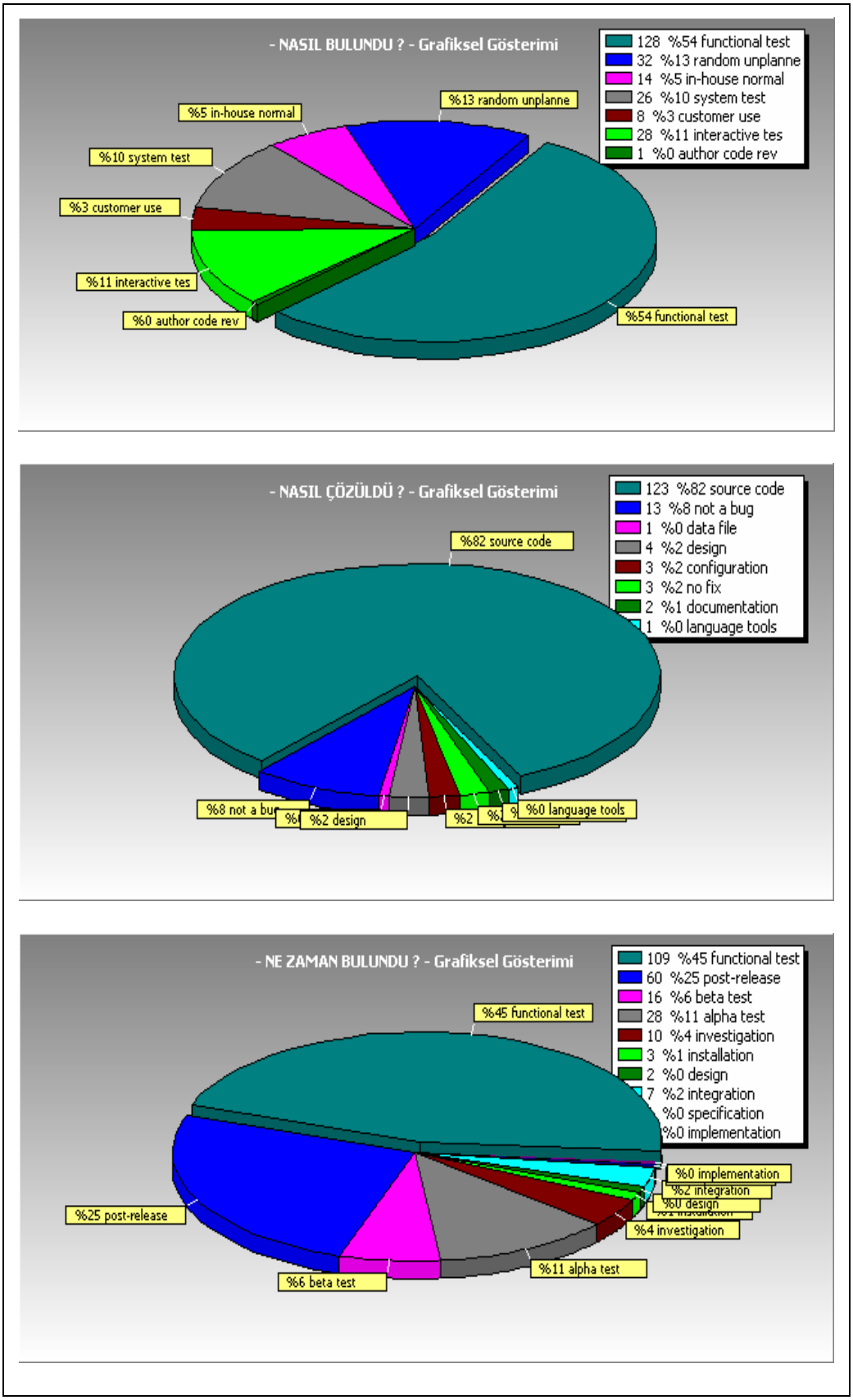


Figure 15 continued.

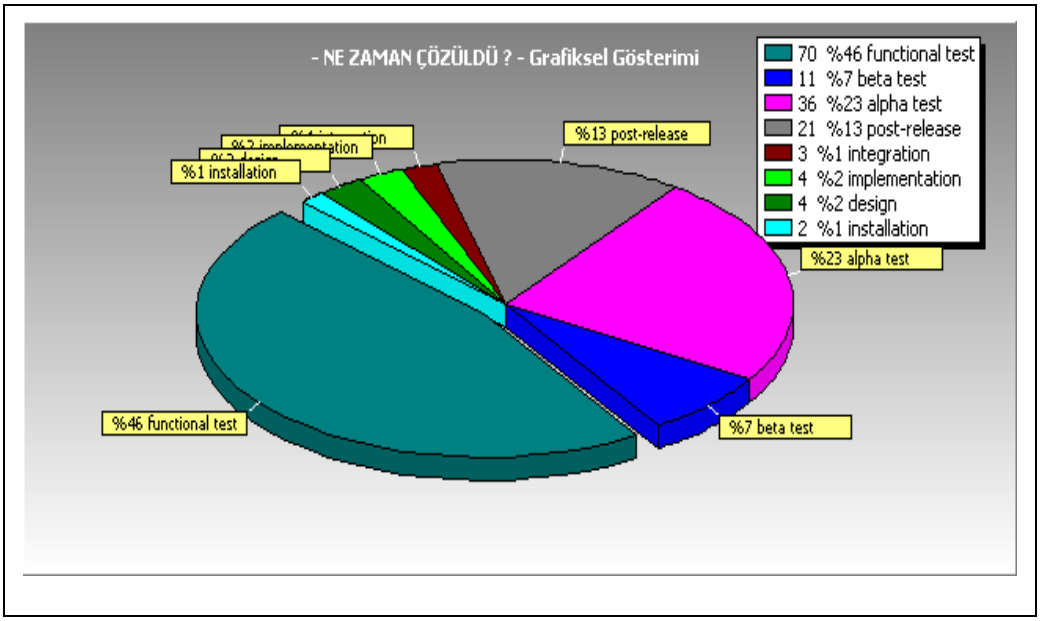


Figure 15 – System20 Defects Measurement Result

The defect measurement report produced for the System37 project is placed in Figure 16.

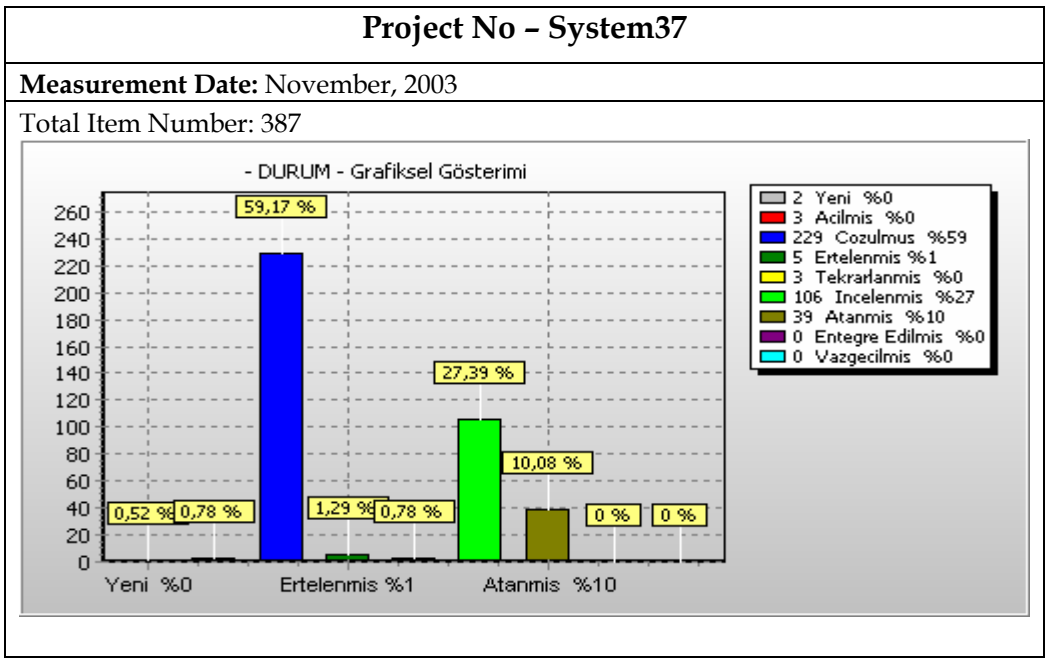


Figure 16 – System37 Defects Measurement Results

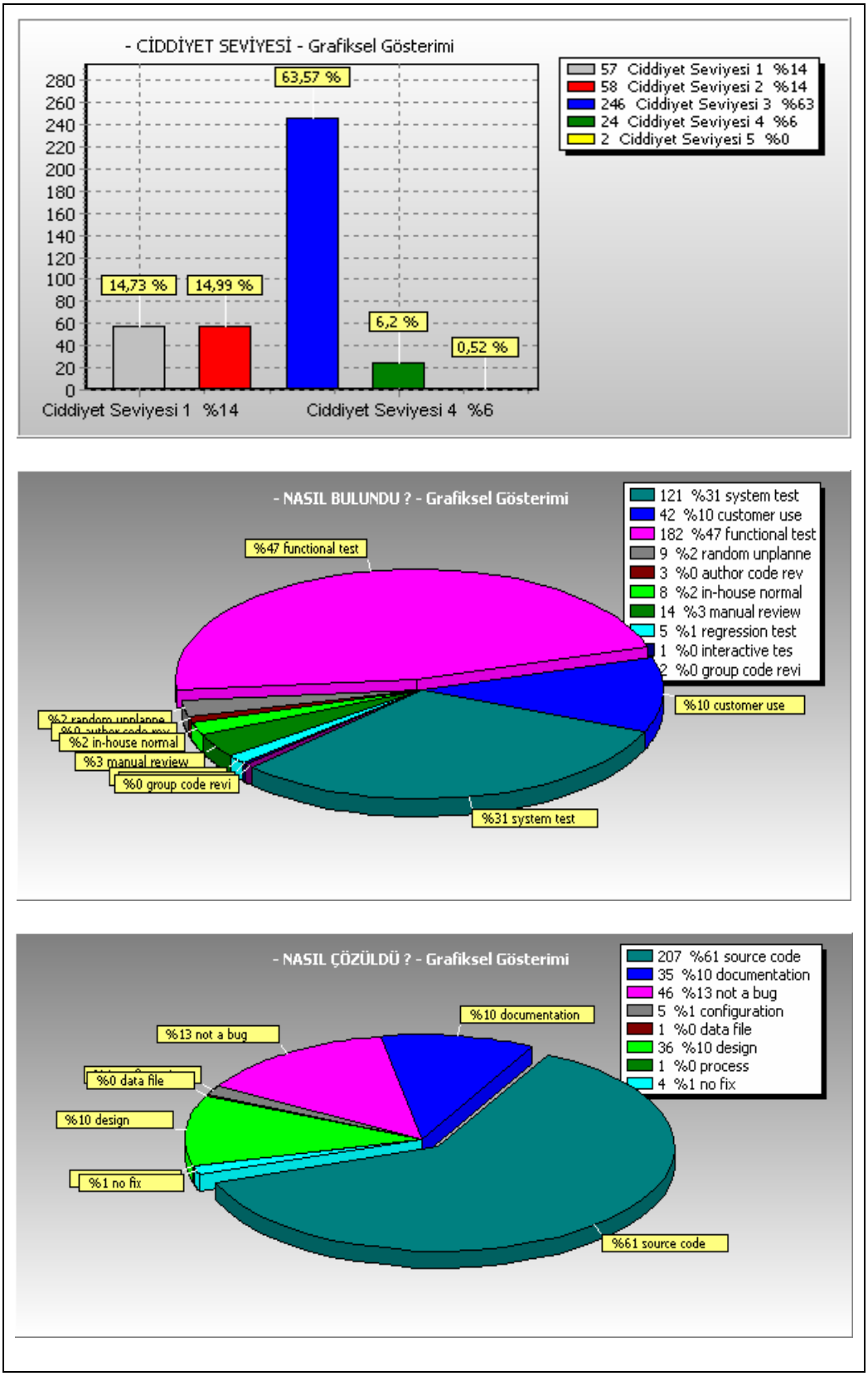


Figure 16 continued.

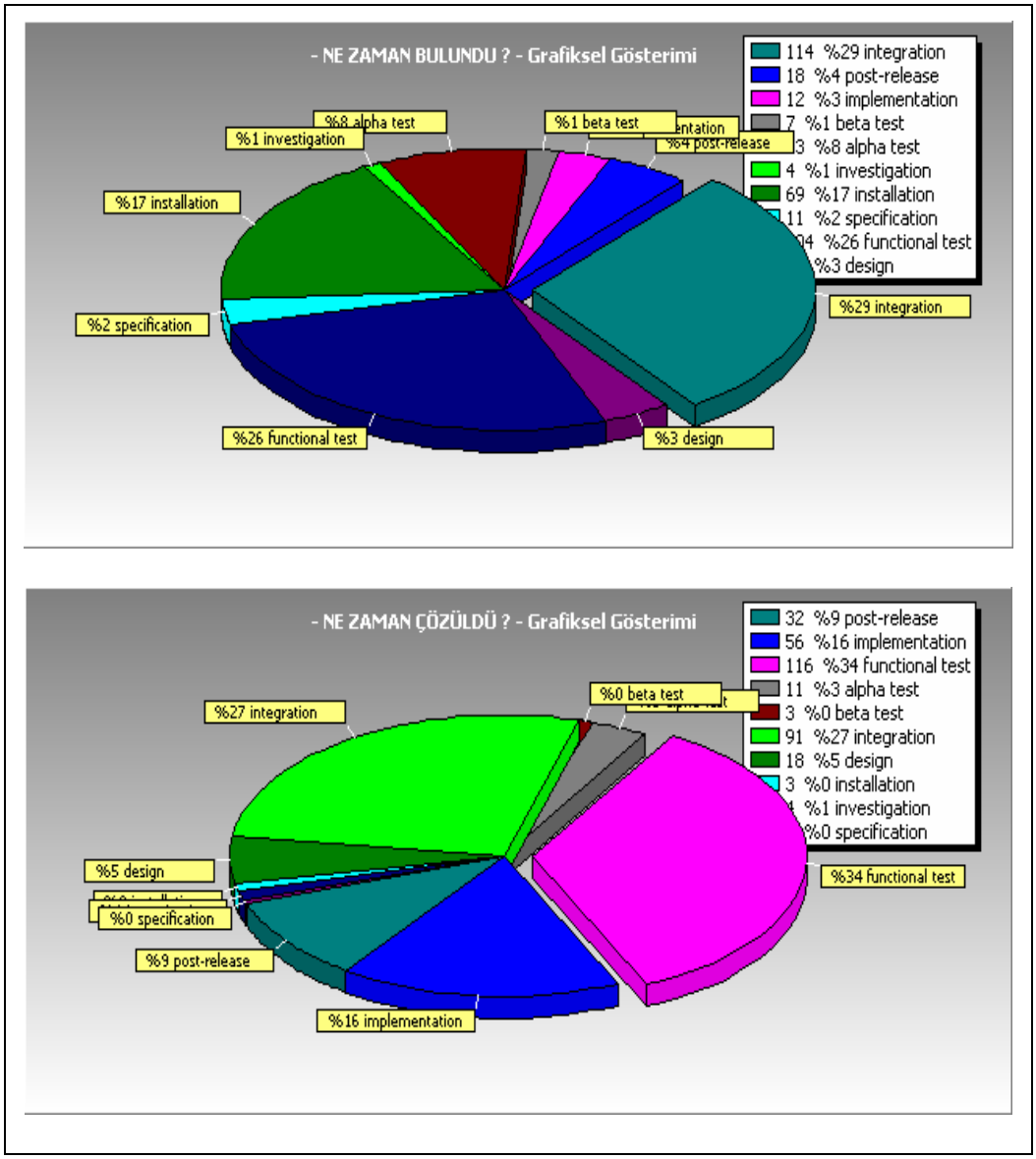


Figure 16 – System37 Defects Measurement Results

6.2.4. Overview of the Defects Measurement

The projects in which defects measurement has been applied are System20, System60, System37, System38, System12, System90, System96, and System11. The distribution of the techniques by which the problems are noticed is as follows: (How Found)

- Functional test (%37),
- System test (%30),
- Customer in-use (%16),
- Random unplanned test (%3).

The problems are resolved usually via modifications to: (How Resolved)

- Source code (%64),
- Not a bug (%15),
- Design (%14),
- Documentation (%3).

The problems are found usually during: (When Found)

- Integration (%27),
- Functional test (%24),
- Post-release (%16),
- Implementation (%8),
- Installation (%3).

The problems are resolved usually during: (When Resolved)

- Post-release (%25),
- Functional test (%21),
- Implementation (%20),
- Integration (%16),
- Design (%5),
- Alpha-test (%4).

These measurement results are discussed in a PAT-G meeting in ASELSAN Inc. As a result, the following interpretations are agreed on.

- The problems are found in usually later stages of implementation in software development process, they should be found in early stages.
- The ratio of the problems that are found by customer use is not very low, so the delivered product (given to customer) has few flaws.
- The test procedures should be more effective within the development process. So, the ratio of problems found in post-release should be less.
- The problems are usually resolved by modification in source code. This seems as a problem in implementation phase of development process. The code review activity should become more considerable in implementation phase.
- Fifteenth out of hundred submitted defects is not a bug, so more effective usage of the ClearDDTS tool should be provided.

6.2.5. Review Status Measurement

These reports include:

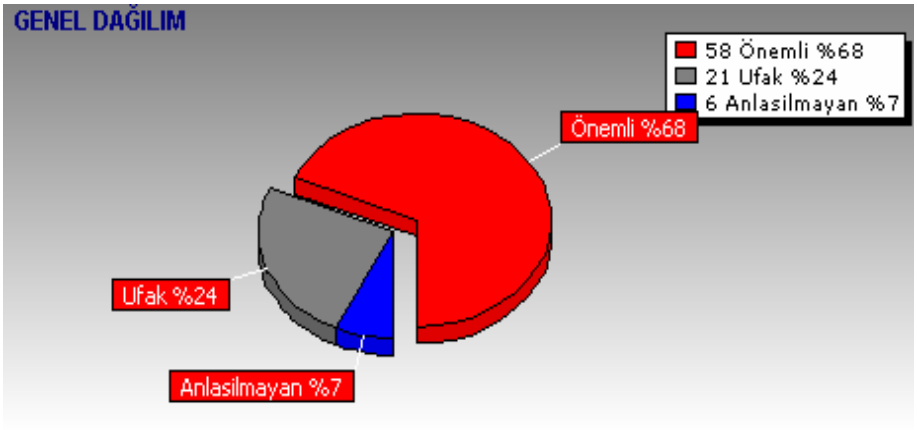
- The information about ratios of "important", "minor", and "incomprehensible" items,

- The information about number of items which is not agreed on at review meeting.

The following report produced after measurement was applied to software specification document of DSP software configuration unit in System37 project and it is shown in Figure 17.

Software Configuration Unit: DSP SRS

Measurement Date: 29/12/2002
Ver. : 01.01



Toplantıda Karar Alınamayan Madde Sayısı = 1

Figure 17 – Kontrol SCU Measurement Result

6.2.6. Overview of the Review Status Measurement

The Review Status Measurement is applied in five software configuration units of System37 project. The following average values are obtained.

- Average ratio of "important" items is % 56.
- Average ratio of "minor" items is % 30.
- Average ratio of "incomprehensible" items is % 14.
- Average ratio of "un-agreed" items is % 3.

These values express that the reviews are quite necessary and useful in the organization since lots of important items are noted during review process. In addition, more than half of them are "important" items.

After review, the distribution of items should be examined. The "un-agreed" items should be tracked during and after review process, and they must be reached a decision anyway.

The ratio of "incomprehensible" items can give some idea about product understandability. Threshold value for the ratio of incomprehensible items can be defined for software product. Then it can be used to make a decision about readability and understandability. But lots of historical data may be needed in order to decide this threshold value.

6.2.7. Source File and Complexity Measurements

These reports include:

- Count Line Code
- Count Line Inactive
- Count Source File
- Number of Functions in Source File
- Complexity Value

This measure can show a snapshot of situation of the existent project. For example, one can easily see the progress in the software configuration unit within one year.

An example of the report in text form is enclosed at Appendix B.

The following Figure 18 was produced for Control software configuration unit from the System37 project. This configuration unit has features of control processor and it has been generated automatically by Rhapsody development tool in C++ programming language. The time span between version 01.01 and version 01.05 is approximately one year and indicates that project is in the last part of implementation phase. The version 01.01 indicates the time of first delivery of product, and the version 01.05 indicates the second delivery of product.

The time spans between versions are:

- Ver 01_01 - Ver.01_02 : 90 work days
- Ver 01_02 - Ver.1_021 : 40 work days
- Ver 1_021 - Ver.1_022 : 20 work days
- Ver 1_022 - Ver.01_03 : 20 work days
- Ver 01_03 - Ver.1_031 : 12 work days
- Ver 1_031 - Ver.01_04 : 16 work days
- Ver 01_04 - Ver.01_05 : 15 work days

Software Configuration Unit: Kontrol

Measurement Date: November, 2003

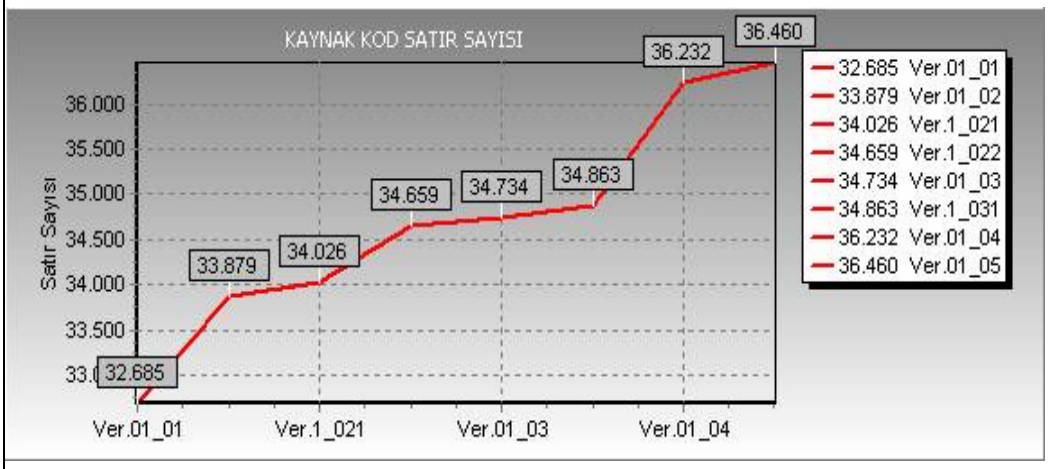


Figure 1 – LOC

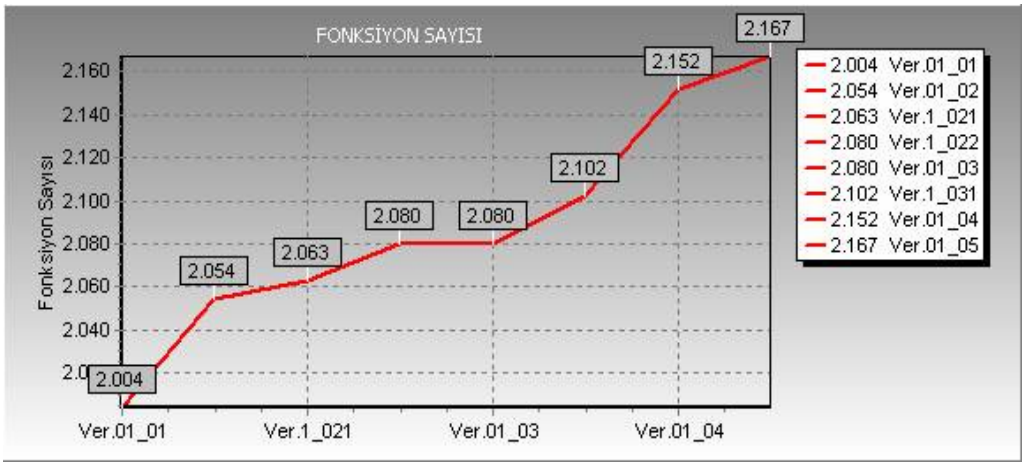


Figure 2 – Function Number

Figure 18 – Control SCU Measurement Result

A report included details of progress was prepared by using both configuration management system reports and measurement results, then it was given to the manager. The main objective of this report was determining the cost of newly added customer requirement after the first delivery of product.

This new functional requirement was implemented between versions 1_031 and 01_04 in the graphs. It took sixteen days with including integration to the system.

Within this period, the lines of code increased 3.9 percent, from 34.863 lines to 36.232 lines. This functionality includes 3.7 percent of all lines of code.

In the same period of time, the number of functions increased 2.3 percent. The number of total functions added in the last year is 163, and the number of functions implemented for adding this functionality is 50. As a result, approximately one of the three added functions was implemented in this period.

Another example is shown in Figure 19, and it was produced by YazOlc-Yardim tool after measurement was applied on the DSP software configuration unit of the System37 project. This configuration unit has signal processing features and it is coded by using C programming language.

From version 00_01 to version 00_02, the number of functions decreased however the lines of code increased. By using the records of configuration management system for this unit, two or more functions were combined in one function and new functionality was also added to the system in this period of time. Another point is that the complexity value was increased due to this modification.

Threshold values about complexity can be defined for softwares. Then it can be used to make a decision about design quality and amount of testing

required. But lots of historical data for various software units are needed in order to decide this threshold value.

Software Configuration Unit: DSP

Measurement Date: November, 2003

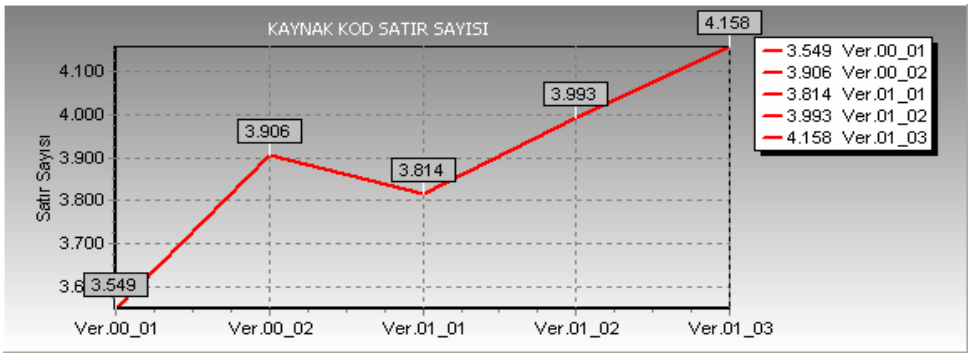


Figure 1 – LOC

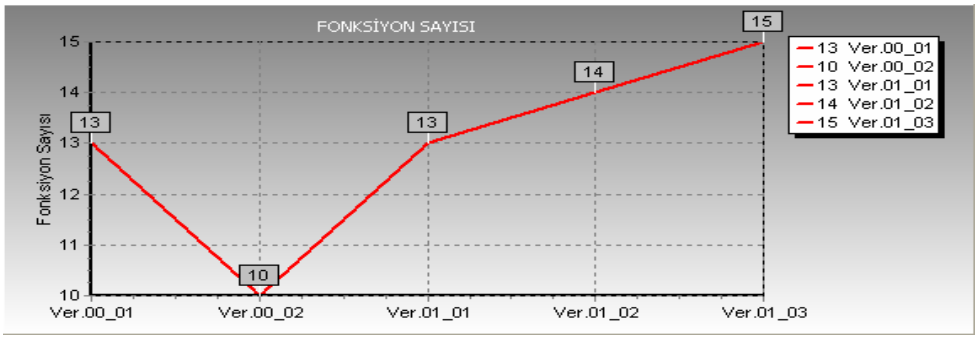


Figure 2 – Function Number

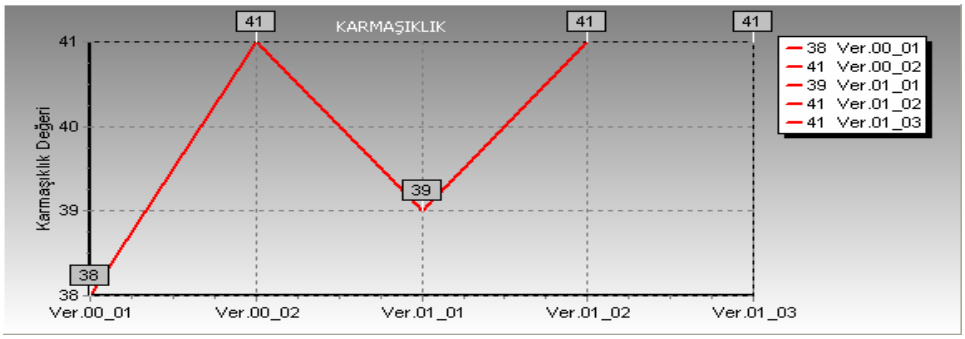


Figure 3 – Complexity

Figure 19 – DSP SCU Measurement Result

CHAPTER 7

DISCUSSION AND CONCLUSIONS

In this thesis study, a software measurement program has been designed and then implemented in order to provide a software development process measurement system at YMM Departments of MST Division in ASELSAN Inc. Software measurement by itself cannot solve problems, but it can clarify and focus one's understanding of them. It is a supporting discipline. Also, managers require methods to plan, track and control the complex software and system processes and products [5]. Measurement can provide the information required to make key project decisions and to take appropriate action.

The stages of software measurement life-cycle are Initiation, Analysis, Design, Build and Implementation; they resemble well-known software development life-cycle steps.

First, the organizational support for measurement was obtained. A briefing was given to the members of YİE team, which is software process engineering group in MST Division. After this briefing, YİE formed a

subgroup to establish software measurement process within organization. This process action team is called PAT-G, and it consists of managers, technical leaders and experts. The member details are given in Chapter 2. The thesis study was sometimes in parallel progress with this group's activities. The PAT-G still works on this objective.

The measurement program started with the initiation stage where the organizational goals were defined.

Then the issues related with these goals were identified in the analysis stage. They were also prioritized and fixed because of the idea that "focus locally and start small". The scope of the measurement program was also defined at this stage. The organization undertakes system projects, but only the software components of projects were considered in the scope of this measurement program. The projects that were in the development phase have been used for obtaining historical data about some measures.

While defining organizational goals and prioritizing the issues, only the managers in the software engineering department and the members of PAT-G participated in assessment. Completion of the first two stages was painless. However, in PAT-G study whose scope was very wide, there were more than two stakeholders, namely, software engineering department, test engineering department, and product quality department. It was too hard to prioritize various issues originating from the various departments since each department had individual priorities which could be very different from

others. So, these stages would become more difficult unless the team focuses locally at first application of a measurement program in an organization.

The measures were selected and also roles were identified at the design stage. In the thesis study, the schedule, product quality, resource and cost, and size and stability measurement categories were selected according to the organizational goals and issues.

It should be emphasized that selecting appropriate measures is very critical for a measurement program. The measurement program should include only the required and realistic measures based on the issues and objectives. This is very important because each measure has a cost which is mentioned in Chapter 1. So, one should avoid selecting unnecessary measures.

In measurement table, all required information about a measure and its application are given in detail. The PSM measurement specifications are useful and applicable to the MST Division of ASELSAN Inc. However, some modifications in tables were done with respect to the organizational structure and data availability. Especially the data items and attributes should be reviewed. In Chapter 4, the measurement Table 14, 15, 20, 21, 25, 26 and 31 were formed after tailoring activities. The others were reviewed carefully, and small modifications in tables were made where necessary. These modifications were such as adding some new items, deleting unreachable and non-existent attributes, changing typical level to SCU (software configuration unit) in order to adapt the tables to the organization.

All attained information up to the build stage in measurement process had to be reflected in a measurement plan. In addition, the measurement plan contains more details about each selected measure such as data reporting process, and measurement application periodicity. This information was defined according to measure data sources in the organization such as tools, forms, and databases.

The output of build stage is the measurement plan. Within the scope of thesis, the measurement plan for System41 project was prepared and then presented to project team leader who is determined and willing to use a measurement program in the project.

In order to constitute a basis and collect historical data for recent projects, some measures have been applied over existent projects at the implementation stage of the measurement program. An important point at this stage is that the team should verify and normalize the collected data. They should be sure about the collected data are the required ones and they are ready for analysis. The verification and normalization activities were realized in the applied measures at MST Division. Some project measures were discarded as a result of these activities. These abnormal data were not used in analysis activity. For instance, the obtained data about severity level of defect in the defects measurement were not analyzed. Default value for severity in the defect tracking tool was three, and some users were not be careful about severity level when they were entering a defect to the system.

So, the severity values usually indicated the third level, which reflected the default value rather than the considered opinion of the users.

The results of these applied measures were beneficial and lots of feedbacks were returned to the organization from these measures. Some of them showed the organizational average values that were discussed in PAT-G meetings. Some decisions about some processes were taken, and they are given in details in Chapter 6. A directive about the more effective usage of ClearDDTS tool is being prepared at the time of writing. After being prepared, it will be published within the MST division. These measures were very useful especially for managers of software engineering departments. The main objectives of these measures are understanding, management, and guiding improvement. The managers reached some important indicators such as average of problem resolution time and when defects were mostly found (in which phase of software development process). In the author's opinion, these measures, problem report status and defects measures should be applied periodically in order to continuously observe and track organizational average values. Other details were presented in Chapter 6.

The applied measures were also snapshots that show the status of an existent project. So, these measures were very useful for technical managers of the projects. The main objectives of these measures are understanding and management. They can easily see progress in the software configuration unit

within one year. They can also obtain a graphical representation of current status of defects in their project. In the lights of these values, they can make a decision with quantitative manner. As a result of the source file measure, a report included details of each version was prepared also by using records of configuration management tool, then it was presented to the manager.

The author's thought is that the source file and complexity measures should be applied periodically especially at implementation stage in order to observe and track progress in software product. A point that the author wishes to emphasize at this point is that, when sample points in the graphics are increased (i.e., versions are given more frequently), one will obtain more sensitive and detailed representations. Other comments about these measures have been presented in Chapter 6.

Although the measures have lots of important information themselves, the combined analysis of two or more measures can give more meaningful information. When the problem status measure result in System37 project was analyzed with respect to activity dates, the long problem solution times can be connected to acceptance phase of the project when lots of new customer requirements added so the project team becomes overloaded.

Also, a measure can be derived from one or more measure. The defect density measure, which is an expression of the number of defects in a quantity of product, can be an example in this case.

To provide data collection and reporting automatically, YazOlc-YARDIM tool has been developed. It can be used for in defects, problem status, source file, complexity, and review status measures. It has feature of report generating. The users' guide of this tool was prepared and published within the organization. An important point is that the measurement reports about problem status measure, which were generated by this tool, were presented at AQAP-150 audit of the MST Division in December 2003.

The most important conclusion of this thesis study is that the measurement activity can be undertaken in a satisfactory way within the MST Division of ASELSAN Inc. The organization has implemented the infrastructure in order to apply a measurement program.

From now on, the prepared measurement plan will be applied in the System41 project within 2004. This plan has the property of being a baseline for other projects that have been started recently in the author's opinion. It includes fundamentals, general, and easy-applicable measures. So, with small modifications it becomes convenient for any project. All the measures in plan should be applied as widely as possible.

This thesis has focused on preparing the organization for measurement. At this point, the organization was analyzed, all needed and applicable measures were identified, the measurement tables, which include all information about when and how measures applied, were defined, and the

measurement plan was prepared. Five important measures were implemented within the organization.

The organization has all materials and documents to apply the basic measures in measurement plan. In the author's opinion, the measures in the prepared measurement plan should be applied at least once in the organization, since the practice of measurement has been shown to appropriate for analysis and applicable in the organization.

One should evaluate the measures and the measurement activities, and store lessons learned [1]. The feedback from previous application and analysis will be very useful for enhancement in measure and increasing effectiveness of measurement program. It will be helpful to identify improvements.

Especially when making an on/off decision, threshold value is usually needed. In order to obtain the average value correctly, one should have a strong database of past measures where results are stored. The threshold value may depend on kind of the software unit. For instance, the complexity threshold value can be different for graphical user interface softwares than embedded softwares. To define the threshold for embedded software, the average values in database should be examined. In short, to have a strong database, the results of applied measures should be reported and stored systematically.

REFERENCES

[1] ISO/IEC, *ISO/IEC CD 15939 – Information Technology – Software Measurement Process*, 2000.

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

[3] Pete Christensen, John Kennedy, Tom Ullrich, "Software Metrics", MITRE, 2001, http://www.mitre.org.

[4] SEL / NASA, *Software Engineering Program – Software Measurement Guidebook*, 1995.

[5] PSM Group, *Practical Software and Systems Measurement*, Version 4.0b, October 2000.

[6] William A. Florac ,Robert E. Park, Anita D. Carleton, "Practical Software Measurement: Measuring for Process Management and Improving", April 1997.

[7] KALDER, *Software Industry Survey in 2001*, 2001.

[8] John McGarry, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, Fred Hall, *PSM Objective Information for Decision Makers*, Addison Wesley, 2002.

[9] Paul Goodman, *Practical Implementation of Software Metrics*, 1993.

[10] Bertrand Meyer, "The Role Of Object-Oriented Metrics", 1998, IEEE Computer.

[11] Carol A. Dekkers, Patricia A. McQuaid, "The Dangers Of Using Software Metrics To (Mis)Manage", 2002, IEEE IT Pro.

[12] Frank van Latum, Rini van Solingen, Markku Oivo, Barbara Hoisl, Dieter Rombach, Günther Ruhe, "Adopting GQM-Based Measurement in an Industrial Environment", 1998, IEEE Software January/February.

[13] Micheal K. Daskalantonakis, "A Practical View of Software Measurement and Implementation Experiences Within Motorola", 1992, IEEE Transactions on Software Engineering.

[14] Tapani Kilpi, "Implementing a Software Metrics Program at Nokia", 2001, IEEE Software November/December.

[15] Betsy Clark, "Eight Secrets Of Software Measurement", IEEE, IT Pro 2000.

[16] Peer Kulik, "A Practical Approach to Software Metrics", IEEE, IT Pro 2000.

[17] Rini van Solingen, Frank van Latum, Markku Oivo, Egon Berghout, "Application of Software Measurement at Schlumberger RPS", The sixth European Software Cost Modeling conference (ESCOM), http://www.escom.co.uk.

[18] V.R. Basili, "The Experience Factory and its Relationship to Other Improvement Paradigms", Springer Verlag, 1993.

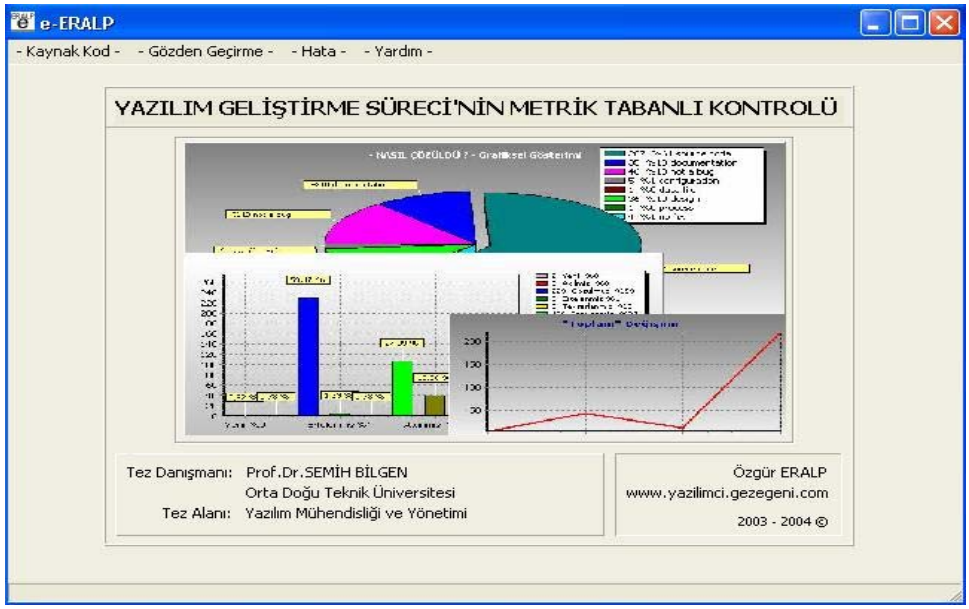
[19] Beth Layman, "Measurement and the CMMs", 6th PSM Conference, 2002, www.teraquest.com.

[20] John VanOrden Carol Dekkers," PSM and Successful Software Measurement", Quality Plus Technologies, Inc., 2002, www.qualityplustech.com.

APPENDIX A

YAZOLC-YARDIM

Bu araç, ASELSAN şirketi MST kısmı YMM bölümünde uygulanan yazılım ölçüm programında kullanılmak üzere tasarlanmıştır. Ölçüm programı içerisinde yer alan 5 temel ölçüm için verilerin toplanmasına, analizine ve raporlanmasına yardımcı olmaktadır. Program içinde bu ölçümlerle ilişkili belirlenen ölçüm tabloları esas alınıp, bu tablolarda yer alan veriler toplanarak, kullanıcıya grafiksel gösterim sağlanmıştır. Ayrıca elde edilen veriler ile grafiklerin dokümantasyonuna da olanak sağlanmaktadır.



Şekil 1 – Ana Mönü

Bu aracın, ölçümünün gerçekleşmesine yardımcı olduğu 5 ölçüm:

- **Kaynak Kod Ölçümü:** Bu ölçüm; proje maliyetinin, gerekli işçiliğinin, zaman çizelgesinin ve üretkenliğin doğru tahminine ve izlenmesine olanak verir. Ayrıca, ürün boyutundaki değişime bakarak muhtemel ek çalışma ve risk tahmin edilebilir.
- **Karmaşıklık Ölçümü:** Bu ölçüm; tasarım kalitesinin ve gerekli test büyüklüğünün belirlenmesine olanak sağlar. Yüksek karmaşıklık düzeyi, yüksek hata/kusur oranının göstergesi olabilmektedir. Yüksek karmaşıklık oranına sahip yazılım bileşenleri ek gözden geçirme, test ve yeniden kodlama gerektirebilmektedir.
- **Gözden Geçirme Ölçümü:** Bu ölçüm; gözden geçirme sırasında elde edilen veriler ile ürünün boyutunun ilişkilendirilmesine olanak sağlamaktadır. Ayrıca, süreç içerinde karar alınamayan maddelerin sayının belirlenmesi ve takibi önemlidir.
- **Hata/Kusur Ölçümü:** Hata gözlenebilen işlevsel bir bozukluktur. Kusur ise, kaynak kodun içerisinde yer alan bir yanlışlıktır, görülebilir veya görülemez. Kusur bir hataya yol açabilir veya açmayabilir. Bu ölçüm; projedeki hataların ve kusurların takibine ve tahminine olanak sağlamaktadır. Tespit edilen hata/kusur sayısı ürünün kalitesi hakkında önemli bir göstergedir. Ayrıca, çeşitli grafiklerden (tespit safhası, çözüm

safhası, kaynağı, ciddiyeti vb.) çeşitli verilere ulaşmak mümkün olmaktadır. Örneğin, hata/kusur tespit yoğunluğuna bakılarak ürünün ulaştığı olgunluk ile ilgili tahmin yapılabilmektedir. Bu ölçüm ile birlikte Kaynak Kod ölçümü kullanılarak "Hata/Kusur Yoğunluğu" kolayca hesaplanabilmektedir.

- **Problem Durum Ölçümü**: Bu ölçüm; projede tespit edilen problemlerin çözüm oranının, buna bağlı olarak çözüm sürecinin kalitesinin belirlenebilmesine olanak sağlamaktadır. Ayrıca, problemin ortala ömrü ve ortalama çözüm süresi, proje değerlendirmesi için önemli göstergelerdir.

1. Kaynak Kod Kısmı

Bu kısım 2 farklı ölçümü (Kaynak Kod ve Karmaşıklık Ölçümleri) kapsamaktadır. Bu ölçümü etkili olarak gerçekleştirebilmek için proje içerisinde "Konfigürasyon Kontrolü" yapılmalıdır. Proje gelişimi esnasında erişilen versiyonlara ait Understand for C++ raporları elde edilebilmelidir. Girdi olarak Understand for C++ aracının "Dosya Metrikleri Ortalamaları (File Average Metrics) ve Proje Metrikleri (Project Metrics)" raporunu kullanmaktadır.

Üretilecek raporun bu bilgileri içermesi için Understand aracında yapılması gereken, üst menüden:

"Projects ► Reports Generate ► Choose Reports" seçilmelidir.

Gelen menüden sadece "File Average Metrics" ve "Project Metrics" seçilip, onaylanmalıdır.

Bu raporları saklama esnasında dosya isminin sonuna "02_01" formatında versiyon bilgisi eklenmelidir. Bu formatın "02" kısmı majör, "01" kısmı minör olarak yer almaktadır. Dosya isimlendirilmesi sırasında versiyon bilgisi eklenmediği takdirde kullanıcının versiyon bilgisini klavye ile girmesi gerekecektir.

YazOlc-Yardim aracı, bu raporlarda yer alan verilerin çeşitli grafiksel gösterimlerini, ayrıca bu verilerin ve grafiklerin Word dokümanı olarak raporlanmasını sağlamaktadır.

Kaynak Kod ve Karmaşıklık Ölçümleri

YazOlc-Yardim aracında yer alan üst mönüler kullanılarak istenilen ölçüm gerçekleştirilebilmektedir.

Şekil 2 – Kaynak Kod Ana Mönüsü

Ölçüm programı içerisinde belirlenen ölçüm tabloları çerçevesinde toplanan veriler:

- Dosya Sayısı
- Fonksiyon Sayısı
- SLOC (Kaynak Kod Satır Sayısı)
- Açıklama Satır Sayısı
- Aktif Olmayan Kod Satır Sayısı

Veri girişi için üst mönüden:

"Bilgi Girişi ► Dosyadan" seçilmelidir.

Ölçüm tabloları taban kabul edilerek kullanıcıdan istenen fakat zorunlu olmayan diğer bilgiler:

- Karmaşıklık Değeri (açılan içerik penceresinden alınabilir)
- Kaynak (Yeni / Yeniden Kullanım / COTS)
- Programlama Dili
- Teslim Durumu (Teslim Edilebilir / Edilemez)

Kullanıcıya sunulan grafiksel tablolar:

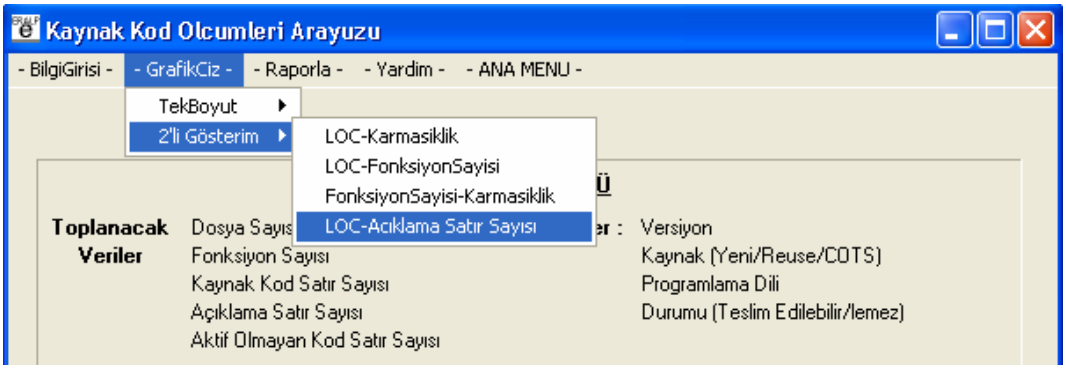
- SLOC ve Karmaşıklık Değeri Değişimi
- SLOC ve Fonksiyon Sayısı Değişimi
- Fonksiyon Sayısı ve Karmaşıklık Değeri Değişimi
- SLOC ve Açıklama Satır Sayısı Değişimi
- SLOC Değişimi

- Fonksiyon Sayısı Değişimi
- Karmaşıklık Değeri Değişimi
- Açıklama Oranı Değişimi
- Aktif Olmayan Satır Sayısı Değişimi

Bu grafikleri çizdirmek için üst mönüden:

"Grafik Çiz ► Tek Boyut ► …."

"Grafik Çiz ► 2'li Gösterim ► …." seçilmelidir.



Şekil 3 – Üst Mönüden Grafik Seçimi

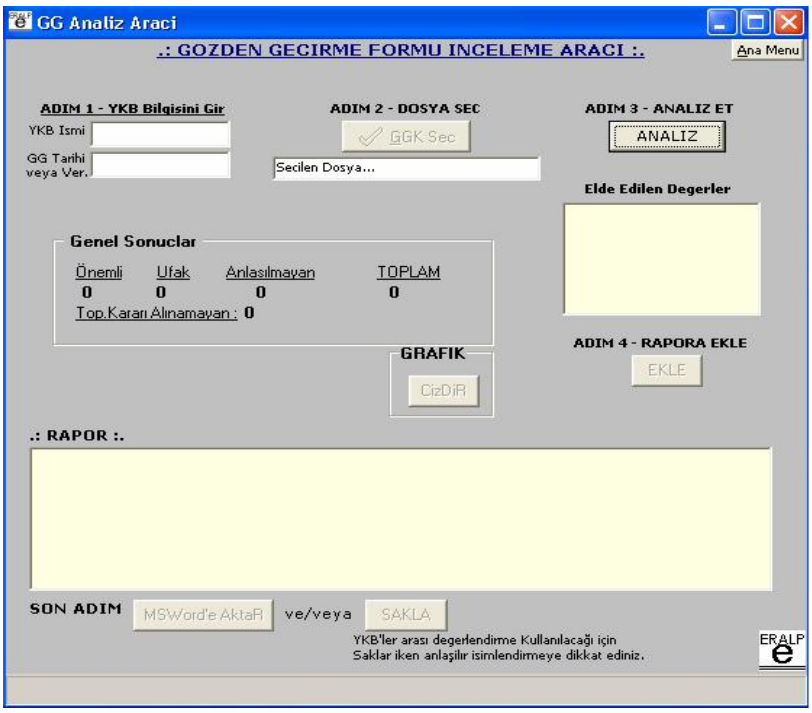
Raporlama: YazOlc-Yardim aracı kullanılarak elde edilen veriler ve grafikler Word dokümanına aktarılarak raporlanabilmektedir.

Ayrıca, verilerin dosyadan okunması yerine istenirse klavyeden girilmesine ve veri listesi üzerinde silme işlemine olanak sağlanmaktadır. Bunun için üst mönüden "Bilgi Girişi ► Manuel" seçilmelidir.

2. Gözden Geçirme Kısmı

YazOlc-Yardim aracının bu kısmı, gözden geçirme sürecinde katılımcılar tarafından doldurulan gözden geçirme formlarında yer alan verilerin

toplanması ve raporlanmasını sağlamak amacı ile tasarlanmıştır. Ek olarak grafiksel gösterim özelliğine sahiptir. Girdi olarak bu formların .txt (text) formatlarını kabul etmektedir.



Şekil 4 – Gözden Geçirme Ana Mönüsü

Gözden Geçirme Ölçümü

Ölçümü gerçekleştirmek için izlenecek adımlar araç içerisinde de numaralandığı gibi:

1. YKB bilgisini gir,
2. Dosya seç,
3. Analiz et,
4. Sonucu rapora ekle,
5. Bilgi girişinin devamı için 2. adıma geri dön,

6. Raporla

Ölçüm programı içerisinde belirlenen ölçüm tabloları çerçevesinde toplanan veriler:

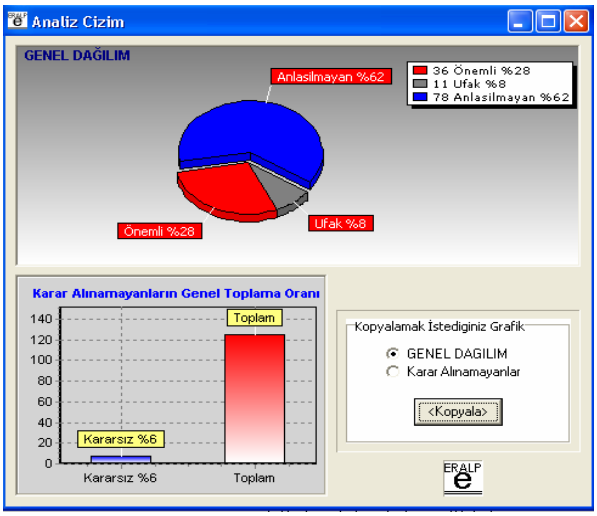
- Önemli madde sayısı
- Ufak madde sayısı
- Anlaşılmayan madde sayısı
- Toplam madde sayısı
- Toplantıda Karar Alınamayan madde sayısı

Ölçüm tabloları taban kabul edilerek kullanıcıdan istenen diğer bilgiler:

- YKB ismi
- Gözden Geçirme Tarihi veya Versiyon

Kullanıcıya sunulan grafiksel tablolar:

- Genel Dağılım Yüzdeleri (Önemli, Ufak, Anlaşılmayan)
- Karar Alınamayan Madde Sayısı Oranı

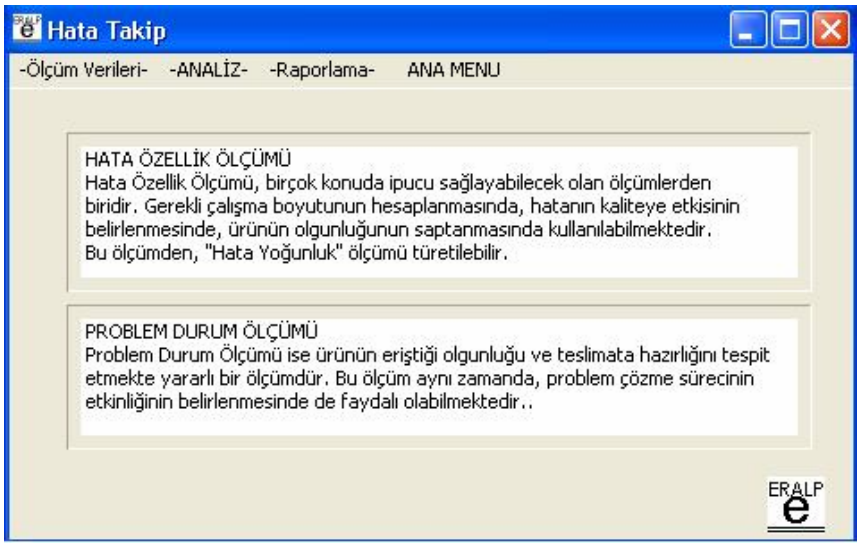


Şekil 5 – Gözden Geçirme Ölçüm Grafikleri

Raporlama: YazOlc-Yardim aracı kullanılarak gözden geçirme ölçümüne ait elde edilen veriler ve ilişkili grafikler Word dokümanına aktarılarak raporlanabilmektedir.

3. Hata Kısmı

Bu kısım 2 farklı ölçümü (Hata/Kusur ve Problem Durum Ölçümleri) kapsamaktadır. YazOlc-Yardim aracı bu bölümde girdi olarak Rational ClearDDTS aracının "Çözülmüş Problemlerin Ayrıntılı Listesi" (Detailed List of Resolved Problems) ve "Problemlere İlişkin Genel İstatistikler" (General Problem Statistics) rapor dosyalarını kullanmaktadır. Bu rapor dosyalarında yer alan verilerden ortalama değerler hesaplanmakta, ayrıca verilerin çeşitli grafiksel gösterimlerini kullanıcıya sunulmaktadır.



Şekil 6 – Hata Kısmı Ana Mönüsü

Hata/Kusur Ölçümü

YazOlc-Yardim aracında yer alan üst mönüler kullanılarak istenilen ölçüm gerçekleştirilebilmektedir.

Bu ölçüm için gerekli girdi "Problemlere İlişkin Genel İstatistikler" (General Problem Statistics) raporudur.

Bu dosyayı okutmak için üst mönüden:

"Ölçüm Verileri ► Hata Veri Dosyasından Oku" seçilmelidir.

Ölçüm programı içerisinde belirlenen ölçüm tabloları çerçevesinde toplanan veriler:

- Durumu
- Ciddiyeti
- Düzeltildiği safha
- Bulunduğu safha
- Sebep olduğu safha
- Nasıl çözüldüğü
- Nasıl bulunduğu

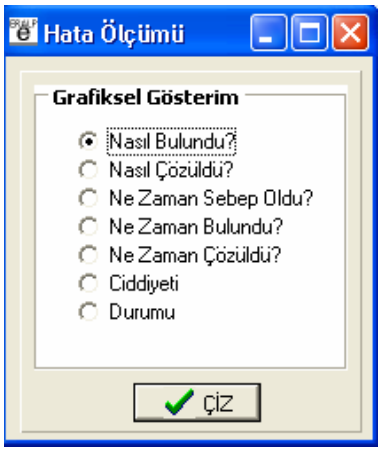
Kullanıcıya sunulan grafiksel tablolar:

- Durumu gösteren
- Ciddiyeti gösteren
- Düzeltildiği safha
- Bulunduğu safha

- Sebep olduğu safha
- Nasıl çözüldüğü
- Nasıl bulunduğu

Bu grafikleri çizdirmek için üst mönüden:

"ANALİZ ► Hata Ölçümü Grafik Analizi" seçilmelidir.



Şekil 7 – Hata Ölçüm Grafik Seçimi

Raporlama: YazOlc-Yardim aracı kullanılarak elde edilen veriler grafikler Word dokümanına aktarılarak raporlanabilmektedir. Bunun için üst mönüden:

"Raporlama ► Hata/Kusur Ölçümü" seçilmelidir.

Problem Durum Ölçümü

YazOlc-Yardim aracında yer alan üst mönüler kullanılarak istenilen ölçüm gerçekleştirilebilmektedir.

Bu ölçüm için gerekli girdi "Çözülmüş Problemlerin Ayrıntılı Listesi" (Detailed List of Resolved Problems) raporudur.

Bu dosyayı okutmak için üst mönüden:

"Ölçüm Verileri ► Problem Verileri Oku" seçilmelidir.

Ölçüm programı içerisinde belirlenen ölçüm tabloları çerçevesinde toplanan veriler, problemin:

- Girilme zamanı
- Atanma zamanı
- Açılma zamanı
- Çözülme zamanı

Toplanan bu veriler kullanılarak;

- Problemin ortalama kaç gün canlı kaldığı (girilmesinden çözülmesine kadar geçen zaman) bilgisi,
- Girildikten ortalama kaç gün sonra ilgili kişi tarafından "açık" durumuna getirildiği bilgisi,
- Atandıktan sonra ortalama kaç gün içerisinde ilgili kişi tarafından çözüldüğü bilgisi hesaplanılıyor.

Hesaplanan bu bilgiler ile birlikte kullanıcıya sunulan grafiksel tablolar:

- Tüm maddelerin kaç gün canlı kaldığına ilişkin grafik
- Tüm maddelerin kaç gün içinde açıldığına ilişkin grafik
- Tüm maddelerin kaç gün içerisinde çözüldüğüne ilişkin grafik

Bu grafikleri çizdirmek için üst mönüden:

"ANALİZ ► Problem Durum Ölçümü Analizi" seçilmelidir.

Raporlama: YazOlc-Yardim aracı kullanılarak elde edilen veriler ve grafikler Word dokümanına aktarılarak raporlanabilmektedir. Bunun için üst mönüden:

"Raporlama ► Problem Durum Ölçümü" seçilmelidir.

APPENDIX B

MEASUREMENT REPORT

An example of source file measurement report in text format is shown in table below.

Table 1 – Source File Measurement Report

KAYNAK KOD OLCUM RAPORU

Versiyon: 00_01
Elde Edilen Veriler:
.:. DosyaSayisi:18 .:. FonksiyonSayisi: 13
.:. KodSatirSayisi:3549 .:. AciklamaSatır: 1575
Kaynak (Yeni/Reuse/COTS): Yeni
Programlama Dili: Ansi C
Durumu (Teslim Edilebilir/Edilemez): Edilemez

Versiyon: 00_02
Elde Edilen Veriler:
.:. DosyaSayisi:18 .:. FonksiyonSayisi: 10
.:. KodSatirSayisi:3906 .:. AciklamaSatır: 1676
Kaynak (Yeni/Reuse/COTS): Yeni
Programlama Dili: Ansi C
Durumu (Teslim Edilebilir/Edilemez): Edilemez

Versiyon: 01_01
Elde Edilen Veriler:
.:. DosyaSayisi:18 .:. FonksiyonSayisi: 13
.:. KodSatirSayisi:3814 .:. AciklamaSatır: 1622
Kaynak (Yeni/Reuse/COTS): Yeni
Programlama Dili: Ansi C
Durumu (Teslim Edilebilir/Edilemez): Edilebilir

Versiyon: 01_02
Elde Edilen Veriler:
.:. DosyaSayisi:18 .:. FonksiyonSayisi: 14
.:. KodSatirSayisi:3993 .:. AciklamaSatır: 1674
Kaynak (Yeni/Reuse/COTS): Yeni
Programlama Dili: Ansi C
Durumu (Teslim Edilebilir/Edilemez): Edilebilir