DESIGN OF A SOFTWARE PLATFORM
FOR
THE QUALITY CONTROL OF MAIN BLOOD PRODUCTS

by

Sıtkı AKYON

M.D., Istanbul University, Medical Faculty of Cerrahpaşa, 1995

Submitted to the Institute of Biomedical Engineering

In partial fulfillment of the requirements

For the degree of

Master of Science

in

Biomedical Science

Boğaziçi University

September 2007

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Yekta ÜLGEN, for giving me the opportunity to work on this research topic, which was an exciting and challenging experience for me, in all aspects, his motivating approach and giving best examples of applications of engineering theory, in practice.

I would like to thank Prof. Dr. Ahmet ADEMOĞLU, for being a member of my thesis committee. I would like to thank Assoc. Prof. Dr. Albert GÜVENİŞ for being a member of my thesis committee.

This study is a part of quality control procedures and applications of Blood Banks of Türk Kızılayı. I am grateful to the Quality Management Department of Türk Kızılayı.

It was a pleasure to share the knowledge; I would like to thank valuable research assistants and the staff of Biomedical Engineering Institute for their support.

I'd like to thank to Güzeyya YILDIRIM, and Artuğ BAYRAKTAROĞLU for their efforts on Word Processing and dactylography. I thank Uğur BOSTANCI, my dear friend, for all the support he has given to me on answering all my questions in Computer Science. This study is dedicated to him.

ABSTRACT

DESIGN OF A SOFTWARE PLATFORM FOR THE QUALITY CONTROL OF MAIN BLOOD PRODUCTS

In modern blood banking services, blood banks and transfusion services, follow a standard operation procedure during preparation and the quality control of blood components. The Quality Management involves identification and selection of prospective blood donors, adequate collection of blood, preparation of blood components, quality laboratory testing and ensuring the safest and most appropriate use of blood/blood components: the objective is to ensure availability of high quality blood components for transfusion. A management model and a managing software is developed for the quality control procedures of main blood products: erythrocyte suspensions, thrombocyte suspensions, and fresh frozen plasma with reference to the Guide by European Council. The user can access detailed data for each of the prepared blood component; to prepare annual summations, and to manage QC processes effectively. It reduces the risk of producing defective components, by giving alarms to the QC Specialist. Unified Modeling Language is used as the Object-Oriented Modeling Design Platform and the software is developed on Eclips SDK, on a Java platform. Since data size is limited a simple memory-save function is used to a Java HashMap.

Keywords: Quality Control of Blood Components, Software for Quality Control Managing, UML Design of Quality Control of Blood Components

ÖZET

TEMEL KAN ÜRÜNLERİNİN KALİTE KONTROLÜ İÇİN BİR YAZILIM PLATFORMU TASARIMI

Modern kan bankacılığında, kan merkezleri ve transfüzyon servisleri, kan komponentlerinin hazırlanması sırasında ve kalite kontrolünde ortak ve standart bir yol izlerler. Kalite Yönetimi, uygun donörlerin tanımlanıp seçilmesini, yeterli kanın alınmasını, kan komponentlerinin üretilmesini, kalite laboratuarında yapılan testleri ve en güvenli ve en uygun kan ve kan ürünlerinin kullanılmasının garantilenmesini içerir: amaç, yüksek kaliteli kan ürünlerinin temininin sağlanmasıdır. Avrupa Konseyi Rehberi referans alınarak, ek solüsyonlu eritrosit süspansiyonu, trombosit süspansiyonu ve taze donmuş plazmanın kalite kontrol prosedürleri için bir yönetim modeli ve bir yönetim yazılımı geliştirilmiştir. Kullanıcı, yıllık toplamlar hazırlamak ve kalite kontrol süreçlerini yönetmek için, üretilmiş her ayrı kan komponenti hakkında detaylı bilgiye ulaşabilir. Kalite kontrol uzmanına alarm vererek, hatalı komponent üretimi riskini azaltacaktır. Obje Temelli Modelleme için, Birleşik Modelleme Dili dizayn platformu kullanılmış ve yazılım, Java platformunda, Eclips SDK üzerinde geliştirilmiştir. Veri miktarı küçük olduğu için bir Java HashMap üzerine basit hafıza kaydı kullanılmıştır.

Anahtar Sözcükler: Kan Ürünlerinin Kalite Kontrolü, Kalite Kontrol Takip Yazılımı, Kan Kalite Kontrolü UML tasarımı

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

QC	Quality Control
Htc	Haematocrit
Hb	Haemoglobin
PRP	Platelet Rich Plasma
CPD	Citrate-phosphate-dextrose
ES	Erythrocyte Suspension
ES-AD	Erythrocyte Suspension in Additive Solution
TS	Thrombocyte Suspension
FFP	Fresh Frozen Plasma
AS	Additive Solution
HIV	Human Immunodeficiency Virus
HBsAg	Hepatitis B surface Antigen
HCV	Hepatitis C Virus
CMV	CytoMegaloVirus
HTLV	Human T-cell lymphotropic virus
HLA	Human Leucocyte Antigen
HPA	Human Platelet Antigen
QA	Quality Assurance
SOPs	Standard Operating Procedures
QMS	Quality Management System
QCP	Quality Control Procedures
GMP	Good Manufacturing Practice
CPDA-1	Citrate-phosphate-dextrose-adenine
OOP	Object Oriented Programming
UML	Unified Modeling Language
GUI	Graphics User Interface
UI	User Interface

1. INTRODUCTION

A Quality System should ensure that no part of the transfusion chain is lacking in quality. Therefore, it seems to be scientifically justified that, in the Quality System perspective, all of the activities that have to be included in the Quality System must be based on validated and applicable methods [1].

It is also essential that the core elements of a common Quality System are selected so as to be applicable throughout the World, and especially for Turkey, throughout the European Community. This will facilitate benchmarking and other types of quality comparisons [2].

1.1. Objective

The main purpose in this project is to create a software model and develop an application and managing software for the quality control procedures of main blood products, such as erythrocyte suspensions, thrombocyte suspensions, and fresh frozen plasma.

The QC Management Software consists in applying the Guide to the Preparation, Use and Quality Assurance of Blood Products book by the European Council.

2. BLOOD BANK METHODOLOGY

Blood components are biological products derived from human blood and plasma. Having special features arising from the biological nature of the source material and, as such, the safety and efficacy of these products relies on the control of the source material at all stages of the manufacturing processes, storage, transport and issue [3].

On selecting individuals for blood and blood component donation, a blood bank must determine the health of the person in order to safeguard both their health and the health of the recipient. All donors should undergo a screening process to assess their suitability.

In the donor screening, a donor's appearance, medical history, general health, relevant lifestyle, and simple laboratory tests are used.

Hemoglobin or haematocrit levels should be determined by laboratory examinations in donor screening each time the donor attends to donate with an application form. Minimum values before donation for female donors is 125 g/l or 7.8 mmol/l (min. Hct = 0.38); and male donors is 135 g/l or 8.4 mmol/l (min. Hct = 0.4) [1].

After the screening tests are passed at the time of the blood donation, the blood container as well as the tubes of the samples collected for testing must be labelled for unique identification of the blood donation. A phlebotomy site is prepared for successful venopuncture. The quantity of blood donation in a standard donation is 450 ml ± %10 exclusive of anticoagulants [1].

The possibility of errors in labeling the blood containers and the blood samples can be minimized with good organization. Whole blood is collected into a three-part-bag containing an anticoagulant solution. The solution contains citrate (as anticoagulant) and cell nutrients such as glucose and adenine.

One of the vacutainer tubes is used for grouping (immunohematological tests), in Grouping Laboratory; the other test tube is used for other screening tests (Microbiological Tests), in Screening Laboratory. And collected blood is taken to Processing Laboratory.

2.1. Blood components

Although, whole blood can still be used in certain limited circumstances, the thrust of modern blood transfusion therapy is to use the specific component that is clinically indicated for the patient.

The components are those therapeutic constituents of blood that can be prepared by centrifugation, filtration and freezing; using blood bank methodology.

The first centrifugation steps will remove, more than half of these nutrients from the residual red cells, during this; the surrounding fluid is only a mixture of plasma and anticoagulant solution [1].

Leucocytes and Erythrocytes now can sediment more rapidly than platelets as they both have a bigger volume than platelets [4].

Table 2.1
The size, and density of the surrounding fluid of principal blood constituents [4].

Constituent	Mean Density (g/ml)	Mean Volume 10^{-15} litre
Plasma	1.026	
Thrombocytes	1.058	9
Monocytes	1.062	470
Lymphocytes	1.070	230
Neutrophils	1.082	450
Erythrocytes	1.100	87

In the second phase of the Centrifugation, most of the leucocytes and red cells therefore settle in the lower half of the bag, and the upper half contains platelet rich plasma (PRP).

Thrombocyte suspension (recovered) is the prepared from platelet-rich plasma. Platelets in PRP are sedimented by hard spin centrifugation; the supernatant platelet-poor plasma is removed leaving 50-70 ml of it with the platelets; finally the platelets are allowed to disaggregate and are then resuspended [3].

After the careful removing of the bag system from the centrifuge; the primary bag is placed into a plasma extraction system and the layers are transferred, one by one, into satellite bags within the closed system. Whole blood may be filtered for leucocyte depletion prior to high speed centrifugation. This procedure enables a separation into almost cell-free plasma and leucocyte and thrombocyte-depleted erythrocytes.

Erythrocyte Suspension In Additive Solution (ES-AD) is the component derived from whole blood by centrifugation and removal of plasma with subsequent addition to the red cells of an appropriate nutrient solution (CPD containing mannitol) [3].

Fresh frozen plasma (from whole blood) prepared either by a single high g centrifugation or two consecutive centrifugation steps (obtention of PRP by a low g centrifugation and plasma extraction after a high g centrifugation of the PRP)

Leucocyte depletion process needs careful validation. An appropriate method should be used for leucocyte counting after leucocyte depletion. The validation should be carried out by the blood establishment using the manufacturer's instructions against the requirements for leucocyte depletion and other quality aspects of the components including plasma for fractionation [3].

The isolation of some plasma proteins, most importantly Factor VIII, fibronectin and fibrinogen, can be achieved by making use of their reduced solubility at low temperature. In practice, this is done by, freezing the units of plasma, thawing and centrifugation at low temperature; this is called Cryoprecipitation. Freezing is a critical step

in the conservation of plasma Factor VIII. During freezing, pure ice is formed and the plasma solutes are concentrated in the remaining water [5].

Whole blood is the source material for blood component preparation. Figure below, is showing the derivation of blood components; from whole blood, with the steps of First Centrifugation, Second Phase of Centrifugation, Extraction, and Freezing. The Components, having QC mark are the components, mostly prepared in number, in blood banks of Turkey: ES-AD(#1), TS(#3), and FFP(#5).

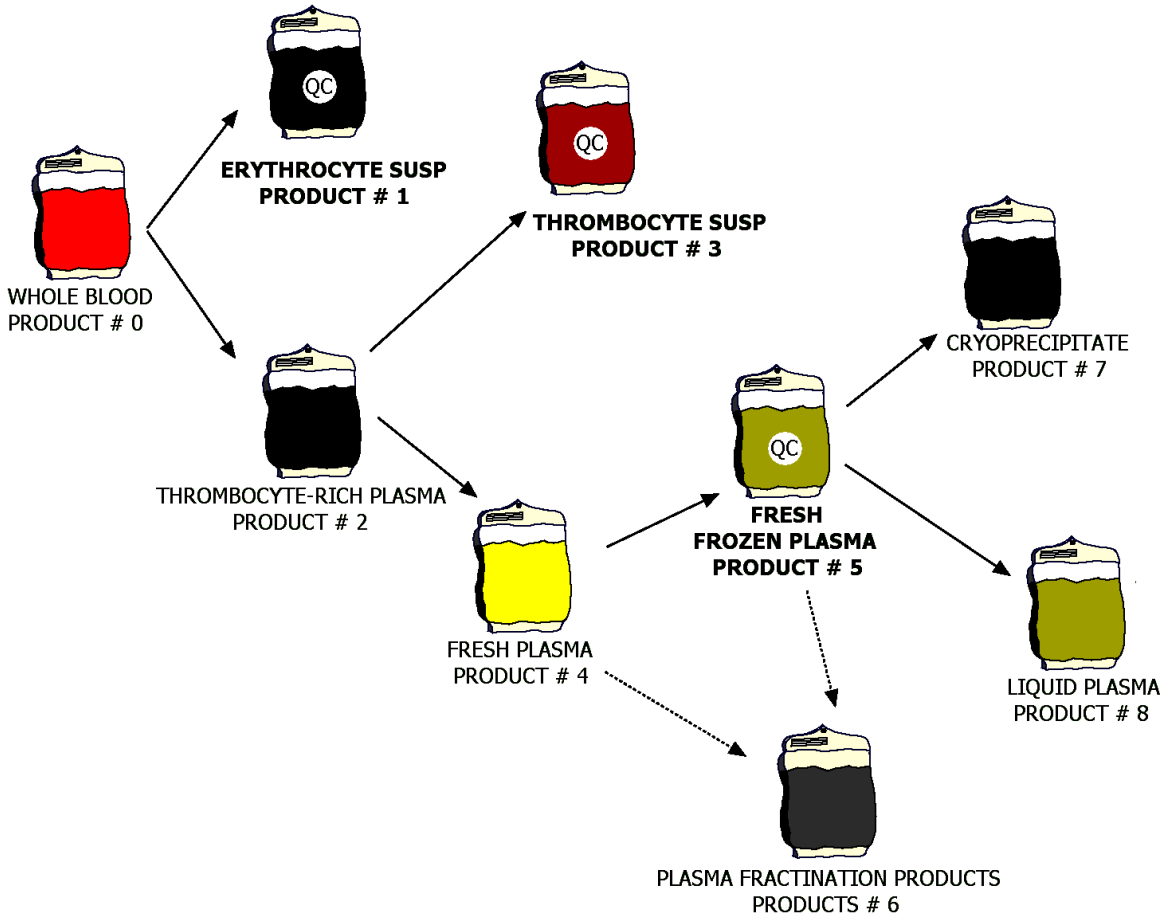


Figure 2.1 Derivatives of Wholeblood and Blood Components. Redrawn From [1].

The elimination of the supernatant of a given blood component is called volume reduction. Plasma depletion is the elimination of the major part of plasma with a procedure ensuring that the initial plasma protein concentration is reduced below a specified threshold, eg .5g/L

2.2. Blood Tests

Table 2.2. is the summary of the tests done in the laboratories and Storage Department of Blood Bank; and relationships with the QC Laboratory. These tests will be taken as "tests done outside the QC Laboratory during preparation" as described in the third chapter of Thesis.

Table 2.2
Summary of the tests performed in different laboratories and Storage Department of Blood Bank, and their relationships with QC Laboratory. Table is created by the thesis writer.

Laboratory	Definition	Tests	Future Relation with QC Lab; on QC of Components :
Hematological Routine Laboratory	Donor First Screening Tests	Haemoglobin; haematocrit; Blood Count;	No
Grouping Laboratory	Immunohaematological Tests, Grouping	ABO, RhD Grouping, (Forward and Reverse); Direct Coombs; Indirec Coombs; Anticore Definition; Phenotyping	**Yes:** ABO, RhD Grouping, (Forward and Reverse);
Screening Laboratory	Microbiological Tests	anti-HIV 1&2, HbsAg, anti-Hbc(when required), anti-HCV, Syphilis(when required), anti-CMV(when required), anti-HTLV I&II(when required), anti-CMV test(when required)	**Yes:** anti-HIV 1&2, HbsAg, anti-Hbc; anti-HCV, Syphilis, anti-CMV, anti-HTLV I&II, anti-CMV test
Processing Laboratory	Inspection during Processing	Volume Measurements; Leakage in the Extractor; Visual and Color Changes;	**Yes:** Volume Measurements; Leakage in the Extractor; Visual and Color Change;
HLA Laboratory	HLA Tests	HLA, HPA	**Yes:** HLA, HPA
Outside Laboratory	Tests not done in the Labs of Blood Bank	Factor VIIIc	**Yes:** Factor VIIIc
Storage Dept	-	Storage	**Yes:** Refrigerator Records; Storage Processes; Storage Effectiveness

2.3. Labeling of the Blood Components during Production

The label on the component ready for distribution should contain eye readable information necessary for safe transfusion, i.e. the unique identity number (preferably consisting of a code for the responsible blood collection organization, the year of donation and a serial number), the ABO and RhD blood group, the name of the blood component and essential information about the properties and handling of the blood component, the expiry date [3].

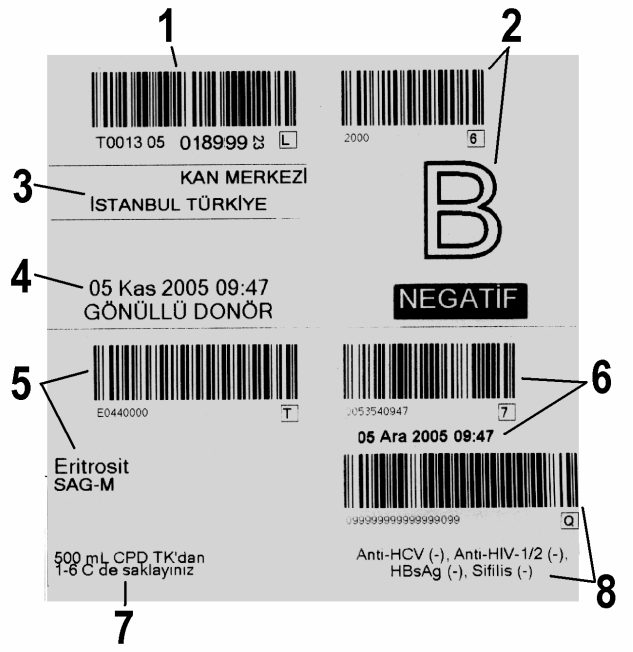


Figure 2.2 A Sample Blood Component Label Sticker of Türk Kızılayı.

This Sticker Label sample, on figure 3.1.on the component has following information: unique identity number (1) and the barcode serial of this number, ABO and RhD blood group (2) and barcode serial of the grouping, the name of the responsible blood collection organization (3), date of donation (4), the name of the blood component (5) and barcode serial of the type, expiry date (6) and barcode serial of the expiry date, storage conditions (7), screen test results information (8) necessary for safe transfusion, and barcode serial of the test results.

The labeling of blood components should comply with the relevant national legislation and international agreements, allowing full traceability of the blood component. This labeling gives traceability to the component itself in the storage department and to the QC test results of Blood Component, by the QC Specialist, and other QC personnel [6].

3. QUALITY CONTROL OF BLOOD PRODUCTS

To maintain public and professional confidence in the safety and efficacy of blood and its products, special care must be paid to all aspects of the quality of the blood components produced [3].

In blood banking and transfusion services, Quality Management involves identification and selection of prospective blood donors, adequate collection of blood, preparation of blood components, quality laboratory testing and ensuring the safest and most appropriate use of blood/blood components.

Quality management may consist of: Quality in procurement (donor, material, reagent); Quality in preparation (efficient and effective blood component preparation); Quality in design and development (improved techniques and procedures); Quality in supply (transportation and service). The Quality management system includes: Quality planning, quality assurance and quality control. And quality assurance (QA) deals with the maintenance of a system to ensure that the performance in a laboratory is of the required quality.

Each blood bank must have written Standard Operating Procedures (SOPs) for each procedure for preventing the errors [7], which may arise from verbal communication. These should provide a complete set of instructions to perform a certain task. They should also specify the way one should perform the assay in the laboratory within its constraints and limitations. The manufacturer's instructions should also be incorporated in the SOP [8].

3.1. The Quality Control System

Within any blood establishment there should be an independent unit with the responsibility of fulfilling Quality Assurance and Quality Control functions.

The quality of the blood components produced depends on the requirements or standards for the product, and the quality management systems (QMS), which enable the product to meet these requirements with confidence [3]. The application of the QMS to the Blood Components is named as Quality Control.

Requirements of a QC System, in the production of blood components, consists of a Quality Management System; QC Personnel (QC Specialists, Laboratory Personnel etc.) and QC organization, the premises of equipment and materials, an easy accessible documentation; known processes of collection, testing and blood processing, quality control proficiency testing, known processes of investigation of errors and accidents, known processes of validation of all processes, the retention of samples and disposal of rejected products, and known processes of self-assessment, internal audit and external audit. All the QC Processes must be complete, and up-to-date according to the New Edition Guides of QC [7]. QC personnel must be individuals functioning independent from other laboratories, and QC Laboratory Areas should be separate from the component preparation areas, and other laboratory areas.

3.1.1. Documentation

Detailed specification lists for the purchase of reagents and other materials used in the QC Laboratories, are required, and only materials from qualified suppliers that meet the documented requirements should be used. Manufacturers should provide a certificate of compliance for every material (blood collection systems, filters and test reagents). Documentation ensures that work is standardized and that there is traceability in all steps in the manufacture of blood components.

Only appropriate and authorized persons should approve documents. Documents should not be hand-written except for those parts where data have to be entered. Any alterations made on a hand-written record must be dated and signed. Documents relating to the selection of donors and the preparation of blood components must be retained according to local regulations in Turkey [9]. Data can also be stored in 'non-written' form, for instance on computer software etc. But the legal regulations must be taken into consideration. Users should only have access to those categories of data for which they are authorized.

3.1.2. Records

Records of the QC results are very important. A distinction should be made between records of results; which may require prompt or almost immediate correction, and records of results which can only be evaluated statistically or by summing up over a certain period [3]. It is essential that the recording system ensures a continuity of documentation of all procedures performed, from the blood donor to the recipient.

In addition to QC Test Results, performed in the QC Laboratories; QC Specialist should have the records of following, for future diagnostics of quality failure: Rejection or deferral of blood donors (numbers, reasons), donor reactions (numbers, sex, age, reaction category), unsatisfactory donations (numbers, category), positive tests for infectious markers (numbers, specific, false), discarded units of blood and blood components (numbers, categories, reasons), outdating of units of blood and blood components (for each category, the outdating as a percentage of the number of usable units), transfusion complications (numbers, category) including transfusion transmitted infection, external complaints (number, origin, category), clerical errors (numbers, category).

The supervisor must sign records of quality control procedures; and records of QC Procedures should also be kept for a period according to our national requirements. It is considered that the retention period should be at least five years [3, 9].

3.2. The Definition of the Quality in QC of Blood Components

The Quality of the Blood Component is the degree to which it fulfills the Standard Minimum Requirements until its expiry date. The Blood Component (the Product) is effective and reliable until its expiry date if the standard Production Process is validated and laboratory testings are performed and inspection results are normal. Otherwise, this group of Blood Components, in the same lot and storage depot, are not reliable and they cannot be used safely until its expiry date [10].

The QC Process takes place in the QC Laboratory. Mainly, QC Laboratory testing Equipment is used. And, it is done by the QC Specialists. The aim in the preparation of blood components is to produce "pure" components, but a very high degree of purity can be difficult and expensive to obtain and might not even be necessary in all instances [3].

But, it is necessary to declare the quality and to be able to make different types of preparations in order to give the clinicians a reasonable choice for patients with different transfusion demands. The purpose of product control is to help the blood bank maintain a high and consistent quality of the prepared product. In this way, the clinical outcome will improve, confidence in component therapy will increase, and the introduction of an adequate component therapy program will be facilitated.

In QC application, the critical control point is the question: “Does this component (product) meet the minimum criteria of quality requirement until its expiry date?” The evaluation criteria for this consists firstly of retrospective procedure controls during the Production Period with such questions as: “During preparation, are all the standard procedures performed?”, “During preparation, are all the standard laboratory tests performed?”, “Are the results of standard laboratory tests according to indicated standard values?”, and “Do the storage conditions comply with the quality criteria?”. And secondly, evaluation consists of the condition of the produced component found in the storage department at that instant with such questions: “ Does the component meet the criteria of quality requirement on the day of sampling?”, “Does the component meet the criteria of quality requirement until the expiry date?”.

The sample size of the QC is 1 of every 100 produced components in a month with a minimum of 4 components [3]. For example, if there are 100 components/per month, then the number of samples is 4; if there are 500 components/per month, then the number of samples is 5; if there are 501 components/per month, then the number of samples is 6. And, sampling frequency is usually once a week, according to the number of produced components.

The first one of every produced 100 components is selected as a sample. For example, if there are 501 components/per month; then the 1st, 101st, 201st, 301st, 401st, and 501st of the components are taken as samples [11].

The test and inspection validations during the production are made and controlled by the personnel of Procedure Laboratory. And, laboratory tests are made by the QC specialist.

Finally, this QC procedure shows the working effectiveness of the Process Laboratory Personnel and if the production meets the GMP standard. It shows us if the storage is good. QC procedure shows if the product (component) is good and can be stored until its expiry date; and the product shows same good quality until expiry date.

3.3. The Main Quality Control Process

This flow chart, drawn by the author, as being an application of the Quality Requirements of Basic Blood Components, stated in the Guide to the Preparation, Use, and Quality Assurance of Blood Components Book by European Council, is a recommendation and a correction of the applications of the QC system of some blood banks in Turkey. It is prepared by the theoretical analysis of the requirements and the inspection of the applications done.

Process is started by the calculation of the number of samples required for QC, according to the number of produced components of same kind, and according to the minimum QC sample requirement of that component. This calculation is stated in the QC requirements of each product, in Appendix sections.

Then QC Specialist collects the required number of samples of indicated type, from the Storage Department, with a request form.

The “collection of other information on QC” is the procedures (in Figure 3.1), which have been done outside the QC Laboratory, during the preparation of the component. The

laboratory units, and the tests they perform during the preparation procedures of blood products, are given on Table 2.2. on the second chapter of thesis [3, 12, 13].

The Main Quality Control Process of the Blood Components is shown in the figure below :

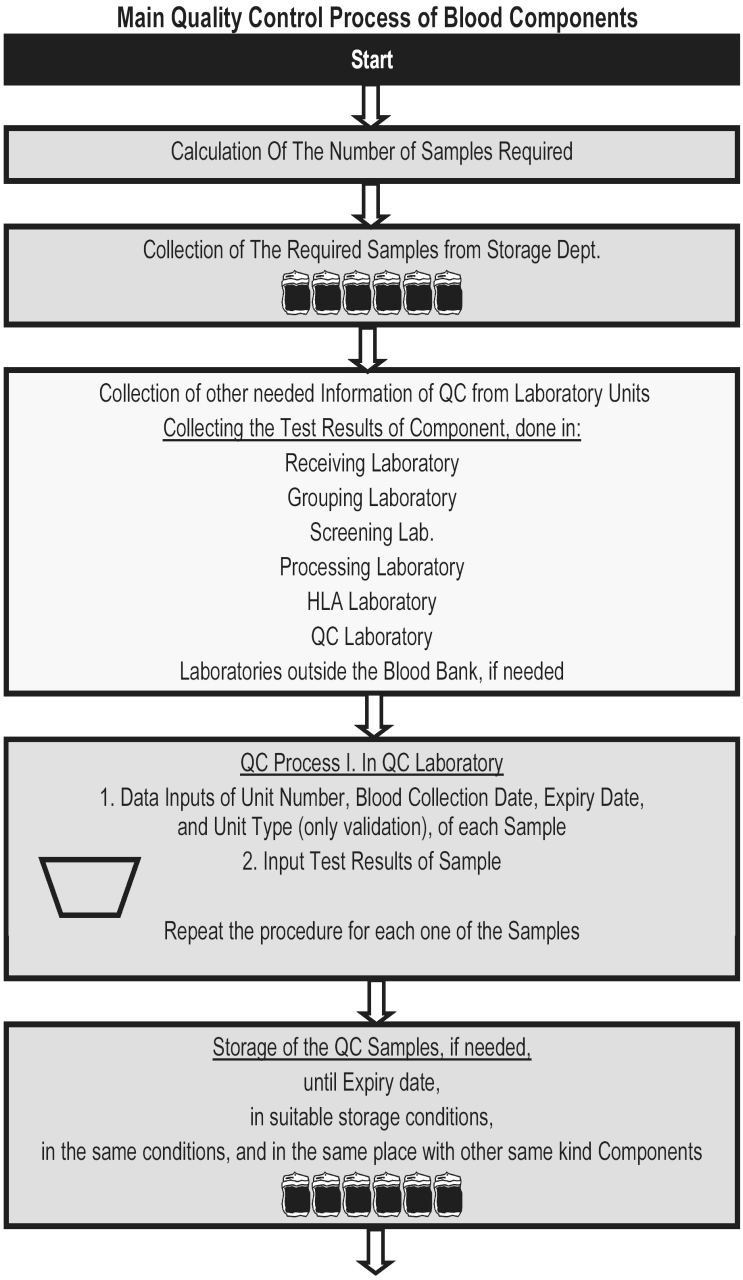


Figure 3.1 The Main Quality Control Process of Blood Components.

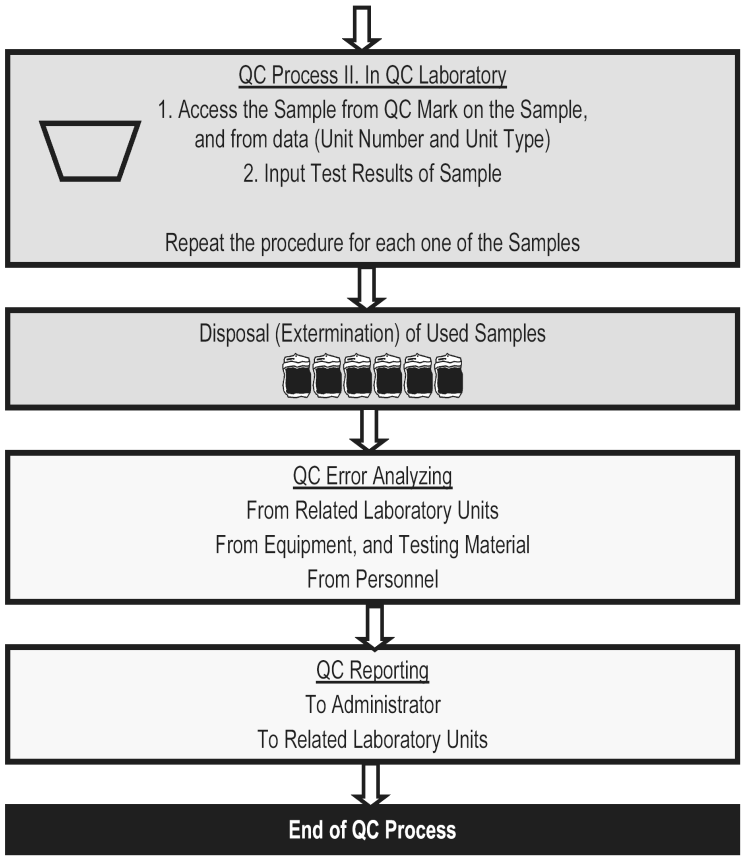


Figure 3.1 The Main Quality Control Process of Blood Components.

For the storage of the QC samples until expiration date in optimal storage conditions, in the same conditions, and in the same place along with the other components of same kind, the QC specialist sends the samples to the storage dept, to be stored there, until the second QC testing, those usually are performed on expiry date.

If a second QC process is needed in QC Laboratory, procedure starts after accessing the same QC Samples, from a "QC" mark on those samples, and collection of the samples from storage department by QC specialist. This time, QC specialist, will only match the unit numbers of the samples of the same kind; and will input the secondary test results of those samples; and repeat the procedure for each one of the Samples.

If needed, for the disposal of the samples with a suitable method with the other components of same kind on the expiration date, the QC specialist sends the samples to the storage department. And QC specialist gets information on this disposal, with a report.

Then, QC specialist searches for the error sources; in case of the QC defects. He gets information on the wrong procedures of related laboratory units, on equipment, on testing reagents, and investigates personnel mistakes.

At the end, QC specialist gives reports and shares the findings with QMS administrator, prints reports, if needed legally, sends information to related laboratory units to warn them about the errors, and records the data to use later, for statistical annual reports.

And the developed QC management software, which is the main project of this thesis, controls each one of the steps explained above and saves the records of QC data on every step.

The whole blood, collected from the donor, is a source material for blood component preparation [5]. Much of the quality control tests necessary to ensure the safety and efficacy of whole blood are performed at the time of the blood collection. In addition to the measures carried out at the time of collection, as stated in the second chapter of this thesis, the parameters listed in the quality control requirements of whole blood, which is given in Appendix A.1. must also be checked. And the minimum QC requirements of the separate blood components, derived from whole blood, will have their own requirements for their specific test results; with the additions of quality control requirements of whole blood.

The detailed flow charts of the QC processes of each one of the blood components (ES-AS, TS and FFP) which are drawn by the author as being an application of the Quality Requirements, are given in Appendix B.1, B.2 and B.3.

3.3.1. Storage of Blood Components during QC Process

Storage conditions for blood components must be designed to preserve optimal viability and function during the QC storage period according to their Storage requirements, those can be found in QC Guides.

In the main storage dept. of Blood Banks; in the Refrigerators, separate spaces should be reserved for the units kept separately awaiting completion of QC testing. The

space for each of these components should be clearly indicated. The temperature within the unit should be recorded continuously. QC Samples must be easy to find, without touching.

3.3.2. Main QC Report Prints

A QC report must consist records of laboratory test results of QC samples, approvals of the tests, main information and explanation on the causes of QC defects, QC requirements and sampling frequency of that component (Units / Month). And as well, it must have the name of the blood bank, date and period it is belonging to and the name of the QC specialist, and administrators.

In addition to the monthly report, an annual report must consist the sums of numbers of passed and failed test results of QC samples, the percentage of quality defects in total number of samples, between the specified dates.

3.4 The Quality Control Process of Erythrocyte Suspension In Additive Solution

The number of samples for the QC of ES-AD component is 4 units/month according to its minimum QC requirements. The QC Requirements table can be found in Appendix A.2 and the QC process of ES-AD in detail is shown in Appendix B.1.

The process is started by the QC Specialist collecting the required 4 samples of ES-AD, from the Storage Department with a request form.

In the First QC Process taking place in the QC Laboratory, the QC Specialist will input the First Day Haematocrit and Hemoglobin test Results of the Sample that must be taken at the beginning of the storage period.

For a period of time of 42 days from the storage of the QC Samples until the expiration date under optimal conditions, along with other ES-AS Components, the second QC Process starts. This time, the QC Specialist will input the 42nd Day Haematocrit level, the

42nd Day Hemoglobin level, and the 42nd Day Hemolysis test Results of the Sample that must be taken at the Expiry period of ES-AS.

The Monthly Report of ES-AS must have the following parameter names, parameter qc requirements, and test results of each of the samples for that parameter in addition to the report content stated in the Main Quality Control Reporting Subject of this thesis according to the QC Requirements: Haematocrit ratio on 1st day, Haemoglobin level on 1st day, Haematocrit ratio on 42nd day, Haemoglobin level on 42nd day and Hemolysis ratio on 42nd day of storage. The examples of the QC Monthly Report, prepared by the author, are given in the Appendix E.1.

3.5 The Quality Control Process of Thrombocyte Suspension (Recovered)

The minimum QC requirements of TS are given in Appendix A.3. The number of samples for the QC of TS Component is 1% of all produced units per month, with a minimum sample number of 10 according to these requirements.

For this component, the process is started by the QC Specialist, collecting the data of the number of the produced components on that month, and calculating the number of the required TS samples, according to the sampling frequency indicated in the QC guides.

In the first QC Process, QC specialist will input the Thrombocyte Count, and Residual Leucocyte Counts. And as a secondary QC process after the storage of the QC Samples until their expiration date, on the 4th day only pH value is measured, and recorded. The platelets should be stored in agitators which should enable satisfactory mixing in the bag as well as gas exchange through the wall of the bag; avoid folding of the bags; have a set speed to avoid foaming [1] during storage.

The complete QC Process of TS is given in Appendix B.2.

In addition to the standard report content, Thrombocyte Count on the 1st day, Residual Leucocyte Count on the 1st day, and pH Value on the 4th day of storage are found on the reports. The examples of the QC Report, prepared by the author, are given in Appendix E.2.

3.6. The Quality Control Process of Fresh Frozen Plasma Suspension

According to the QC requirements of FFP Component, which are given in Appendix A.4, the number of samples is 1% of all produced units / month, with a minimum sampling of 4. So for the starting of the process, by the QC Specialist, the number of the produced components on that month must be known.

The Complete QC Process of FFP is as shown on the diagram in Appendix B.3. For this component, there is no secondary QC Process. And Factor VIII measurements are performed outside the QC laboratory.

In addition to the standard report content, Residual Erythrocyte Count on the 1st day, Residual Leucocyte Count on the 1st day and Residual Thrombocyte Count on the 1st day of storage must be found on the reports. The examples of the QC Report, prepared by the author, are given in Appendix E.3.

3.7. Defects in Quality

Main Sources of Quality Defects are standard routine laboratories, blood processing laboratories, storage department, computers and electromechanical devices, and personnel (human error).

Component bag problems, centrifuge equipment, and problems of extractor usage during preparation can be the main quality defect causes [1] arising from production. The causes of quality defect, arising from the physical condition of the component, are its storage and transportation conditions. And lastly tests having results that do not pass the

quality criteria, can arise from the testing equipment, such as laboratory blood count equipments or others.

In case of the presence of quality defects, the components that have not yet been released for distribution must be kept in quarantine and should not be released. Quarantine continues until all the required secondary quality controls and laboratory tests have been completed, laboratory results meet the established requirements, and decisions are made for the usage or disposal of the components.

The Storage problems are almost always ended by disposal of the Components.

However, if the laboratory values are below the requirement but are in the scientifical range, the blood can be used, if needed greatly, but the expiry period is much less than normal as is shown in the figure below:

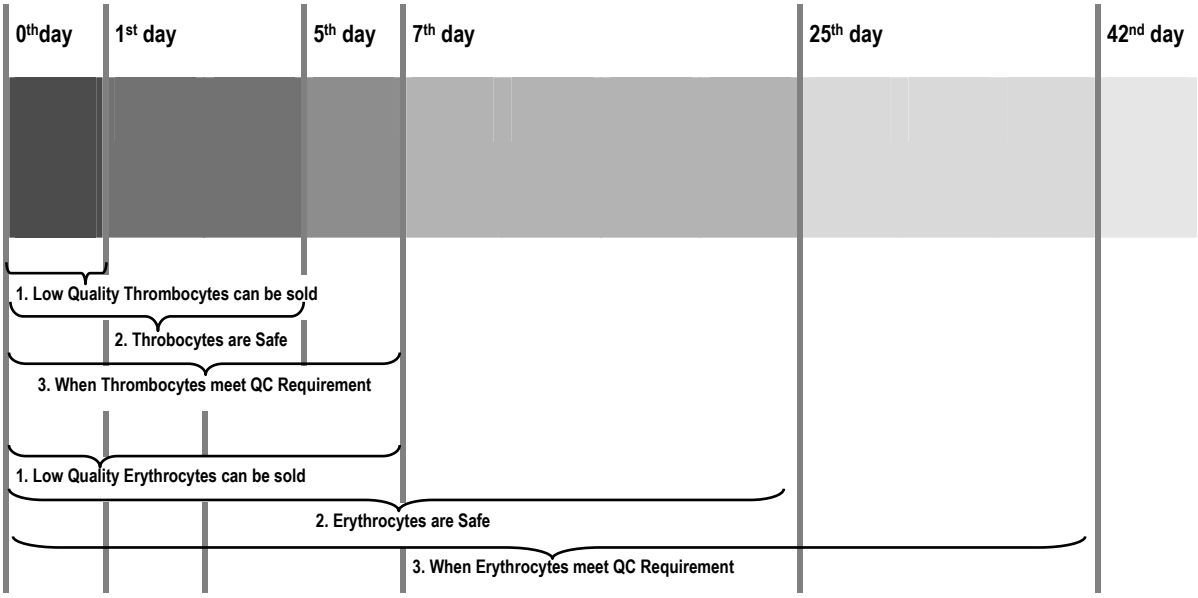


Figure 3.2 Usage and Expiry Period of TS, and ES Preparations, in different Conditions: 1.Low Quality, 2.Good Stored but not Quality Controlled, and 3.Good Stored, and Quality Controlled

For example, if Erythrocyte Preparations are not well-preserved; ATP and 2,3 DPG levels of the cells are decreased, and Potassium level is increased [14, 15]; A pyruvate + Inosine + Phosphate + Adenine containing solution is given to Component; and this preparation can be used for transfusion, in a period of 24 hours [1].

Or for example, If we have no fresh blood (less than 5 days), for new borns, with IgA deficiency who have histories of allergic transfusion reactions, transfusions can be done with washed erythrocyte cells, in a period of 6 hours [1].

On these kinds of transfusions, clinical users should be informed of the properties of all components. These transfusions, which are not done with reliable blood components, are not recommended in Turkey; because of lack of Haemovigilance (Tracebility of the Blood, until transfusion). So, in order to institute an adequate scheme of component therapy, all products must be carefully defined and minimum requirements set.

The duty of the Quality Control Specialist is giving the QC Report of the month; not to recommend disposition of the Components. As the reports are sent to related departments of the Blood Bank, the investigative and corrective process begins.

If, the number of the disposed components are more than 1 % of all prepared components of the same kind; a three-month-retrospective investigation also begins [1].

All complaints, production records from donation, reasons of disposition, transfusion reactions and other information, about defective blood components must be documented, carefully investigated, and should be dealt with as quickly as possible; Written effective procedures must exist for recalling defective blood components or blood components suspected of being defective. These written procedures must encompass any look back procedures, which may be necessary. The procedures should be communicated to the facilities, where the blood components are used [3].

Preventive and corrective actions should also be documented and assessed for effectiveness after an appropriate period. And needed education must be given to the Personnel.

4. MODELING THE SYSTEM

The blood bank or transfusion services should have a computer software to manage their QC processes: validation, data inputting and report printing, later risk analysis [16].

This Software is a database, archiving records of QC, a text editor for printing and designing reports, a mathematical data analyzer for statistical analyzing. It has functions to collect data from barcode readers or via RS232 serial port, and functions to share data via internet or internal network. So, the development method of the software; the modeling type of the system, the platform to develop the software and the testing procedures included in the software code are selected according to these functions [17].

4.1. The Waterfall Method of Development

The Waterfall development Method is used in this project. This is the advised model of development for the blood QC Software [1] by most of the authorities. The waterfall model takes its name, from the fact that it views software developments as a set of phases that a development team goes down in a cascading fashion, like water going down a waterfall.

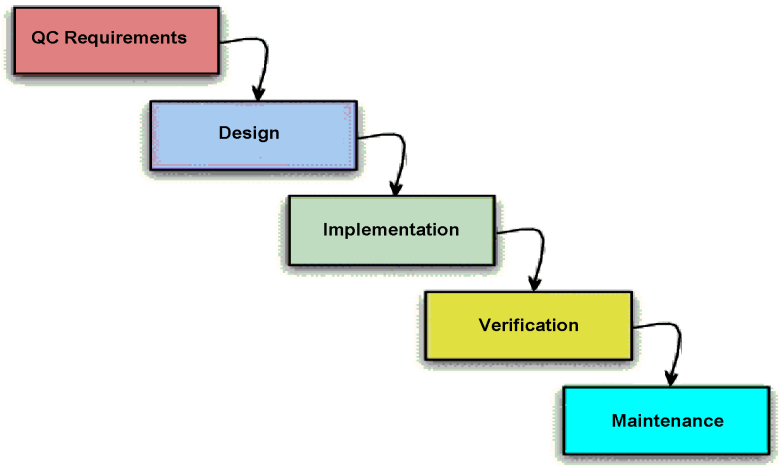


Figure 4.1 Waterfall Development for QC Software

In the waterfall model, modeling starts at the first phase, and move on to the next one as soon as the current phase is complete. The number of phases in this QC waterfall is five.

Firstly, for the establishing of the QC requirements, QC guides are found, read, inspections in the blood banks, verbal communication with the QC specialists are performed, and the analysis of the procedures are done. The conflicts and the solutions are established, between QC requirements, and processes in practice. The detailed process charts are drawn, which some of them are given in this thesis.

In design phase, based on the QC requirement documents, produced in the preceding phase, creation of design documents, which will act as the blueprints for the application, are built. Since, if the requirements documents specify what the application will do, “the design documents” specify how the application will do it. This was the most important modeling phase, in this project.

On the implementation phase, development of software application using the design documents has been done.

Verification is done with test procedures inside the code, and main application is tested to make sure that it meets the requirements and is free of errors.

Maintenance of the QC Software is not done. This software must be used in a blood bank, for a pilot application. During that time, it may be modified or updated to meet new QC requirement needs. And the errors, which are not detected during the verification phase can be corrected.

The use of the QC software in a blood bank is critical. It must be fully validated to ensure that, it meets the predetermined specifications for its functions, to preserve correct data integrity. And to ensure its use to be properly integrated into that centre's operating procedures.

4.2. Modeling

In this QC software development, modeling is used to provide structure for problem solving, to experiment to explore multiple solutions, furish abstractions to manage complexity, reduce development time, and manage the risk of mistakes.

Modeling behavior has the advantages to let domain experts specify outward view (what) so that developers can construct inside view (how); and let developers approach an element and understand it; it has basis for testing [18].

4.2.1. Object Oriented Programming (OOP)

Object Oriented Programming (OOP) is a programming language model organized around "objects" rather than "actions", and “data” rather than “logic”. So, in OOP, the programming challenge is how to define the data; not how write the logic.

Object-oriented programming takes the view that what we really care about are the objects we want to manipulate rather than the logic required to manipulate them. It’s just the opposite of the procedural languages, that is a flowchart, a logical procedure, that takes input data, processes it, and produces output data.

The first step in OOP Model of this QC Software was to identify all the objects that we want to manipulate and how they relate to each other, an exercise often known as data modeling. After the identification of an object, one can generalize it as a class of objects, and define the kind of data it contains and any logic sequences that can manipulate it.

Each distinct logic sequence is known as a method. A real instance of a class is called an "object" or, an "instance of a class." The object or class instance is what we run in our computers. Its methods provide computer instructions and the class object characteristics provide relevant data. We can communicate with objects - and they communicate with each other - with well-defined interfaces called messages [19].

The concepts and rules used in object-oriented programming provide important benefits. The concept of a data “class” makes it possible to define subclasses of data objects that share some or all of the main class characteristics. Called inheritance, this property of OOP forces a more thorough data analysis, reduces development time, and ensures more accurate coding.

Since a class defines only the data it needs to be concerned with, when an instance of that class (an object) is run, the code will not be able to accidentally access other program data. This characteristic of data hiding provides greater system security and avoids unintended data corruption.

The definition of a class is reusable not only by the program, for which it is initially created, but also by other object-oriented programs (and, for this reason, can be more easily distributed for use in blood QC management software) [20]. The concept of data classes allows a programmer to create any new data type that is not already defined in the language itself.

An object has a public interface that other objects can use to communicate with it. But the object can maintain private information and methods that can be changed at any time without affecting the other objects that depend on it.

An object's behavior is expressed through its methods, so (aside from direct variable access) message passing supports all possible interactions between objects. Objects don't need to be in the same process or even on the same machine to send and receive messages back and forth to each other.

The main objects and their relations and communications, which are created in this software are explained 5th chapter of this thesis.

4.2.2. Unified Modeling Language (UML)

Unified Modeling Language (UML) is an effective type of modeling complex software systems that specifies the functional requirements of system in an object-oriented manner, and structural modeling specifies a skeleton that can be refined and extended with additional structure and behavior [18].

During UML modeling, developer identifies “the actors” that interact with the element; organizes actors by identifying general and more specialized roles; and for each actor, considers the primary ways it interacts with the element; considers exceptional ways of interaction; and organizes these behaviors as use cases. Every behavior is one usecase [18].

In a usecase diagram, “Actor” is an actor, who or what uses the system, representing a role, and communicates with the system by sending and receiving messages. Actors are in control and initiate actions.

Actors can be ranked, Primary and secondary actors [18]. And the flow of messages between actors and the use case depends on conditions and exceptions. We must describe which entities are modified and used; when is the usecase considered to be finished and what kind of value is delivered to the actor [18]. A well-structured usecase shows “the single identifiable behaviour” of the system, “common behaviours” by using inclusions, “the variants” by using extension [19].

For the OOP-UML modeling the system context, the following steps are applied to QC Processes: Identification of the actors that surround the system; which groups require help from the system; which groups are needed to execute the system; which groups interact with external hardware; which groups perform secondary functions for administration and maintenance (1); organization the actors (generalization relationships) when needed stereotype actors (2); and putting them in a use case and connect to use case [18].

For the model of the QC management software, UML modeling analysis, made by Product Demo of Borland Together Software, which is a visual modeling platform, is used.

4.3. The Development Platform of the Software: Java

As the software platform, Java is selected. Since, java programs are compiled into machine-independent bytecodes, they run consistently on any Java platform; and we can avoid platform dependencies [21].

Java is the most popular object-oriented language, with great modularity. The source code for an object can be written and maintained independently of the source code for other objects. Also, an object can be easily passed around in the system.

In java, created Classes have reusability; in the Subclasses specialized behaviors from the basis of common elements provided by the superclass if found. Through the use of this “inheritance”, a developer can reuse the code in the superclass many times.

A developer can implement superclasses called “abstract classes” that define "generic" behaviors. The abstract superclass defines and may partially implement the behavior but much of the class is undefined and unimplemented.

4.3.1. Eclipse SDK On Java

Eclipse is a component-based platform used as a workbench on java, and Eclipse-based applications are highly modular. The one, which is used for this project is “Eclipse SDK”. It consists of a mixture of plug-ins, itself which together make it what it is; other Eclipse-based applications may share some of those plug-ins, but usually bring along their own set of plug-ins that differentiate them [21].

Elements of Eclipse Workbench consist of the main application menu bar, toolbar, an editor area (into which editors may be opened), and several views for some functional

area or user activity, such as Java development, resource management, and so on, depending on what the application provides.

4.3.2. Window Builder On Eclipse

Trial version of WindowBuilderPro software is used to design the user friendly GUI of the QC management software. It is a Plug-in of Eclipse SDK which is working on Java.

5. SOFTWARE FOR QC MANAGEMENT

After the selection of modeling and development platforms and their plug-ins, management model is designed and the managing application software is developed for the QC procedures of main blood products: Erythrocyte Suspensions, Thrombocyte Suspensions, and Fresh Frozen Plasma.

The software is helpful to input, store, and report the quality control data, on each step of the QC Process, which are given in Appendix B.1., B.2. and B.3. User can access and edit the test results of each blood components samples of different kind, even if the data of that sample is limited with its unit number. Barcode readers can be used for this purpose.

Alarm function avoids QC Specialist to forget QC tasks; and this reduces the risk of producing defective components.

5.1. General information on the software

By our source code, with the “methods” of the object “Form” and use of the methods of other objects, it is possible to talk about the creation of a folder of a “concept product”, of a specified product group, and of a selected number of samples, which has a specific number of “Parameters”. User can designate values to those parameters.

The implementation of a “component sample” is characterized by product type, product name, number of quality control forms per unit of time, quality control frequency, number of samples, blood product expiry period, and the parameters which has the integers: Scientific Minimum Value, Scientific Maximum Value, Quality Minimum Value, Quality Maximum Value, Acceptable Qualifying Percentage and Sample Testing Period.

Software has main Constants as shown in the figure 5.1.

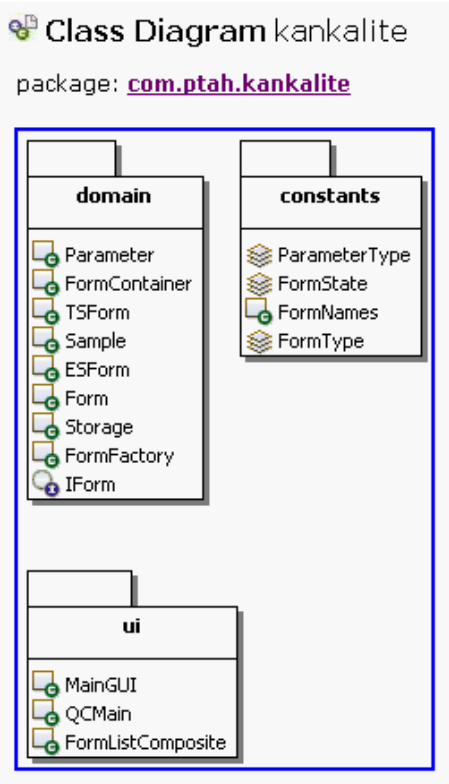


Figure 5.1 Objects, Constants, and UI of Software.

The software is based on a database of "monthly quality control reports", but not on a database of "quality control samples". The name "form" is used as: "a blank data sheet, containing information about QC samples, test results and explanations on the causes of QC defects of that month".

So "a form", is a monthly report when it is displayed on the screen or printed. And again "a form" is an annual report, when the data designated on its parameter integers are summed. For printing function, a simple plug-in of Java Eclipse SDK is used.

Software checks the date and the related forms which has TR or SR status (to give alarm) at the beginning of that session, and checks the default values.

The Source code of the Software is added to Appendix H. And the software written on a CD is added to Thesis as an Appendix.

5.2. General Design of the Model

For better understanding of the computing mechanism of this QC software, firstly the model of the domain code (com.ptah.kankalite.domain) must be analyzed .

"The domain" is the skeleton which is very well defined and refined. Until the end of the modeling process, implementations of the "objects" in the software are postponed to make the blueprints of those objects defined and flawless.

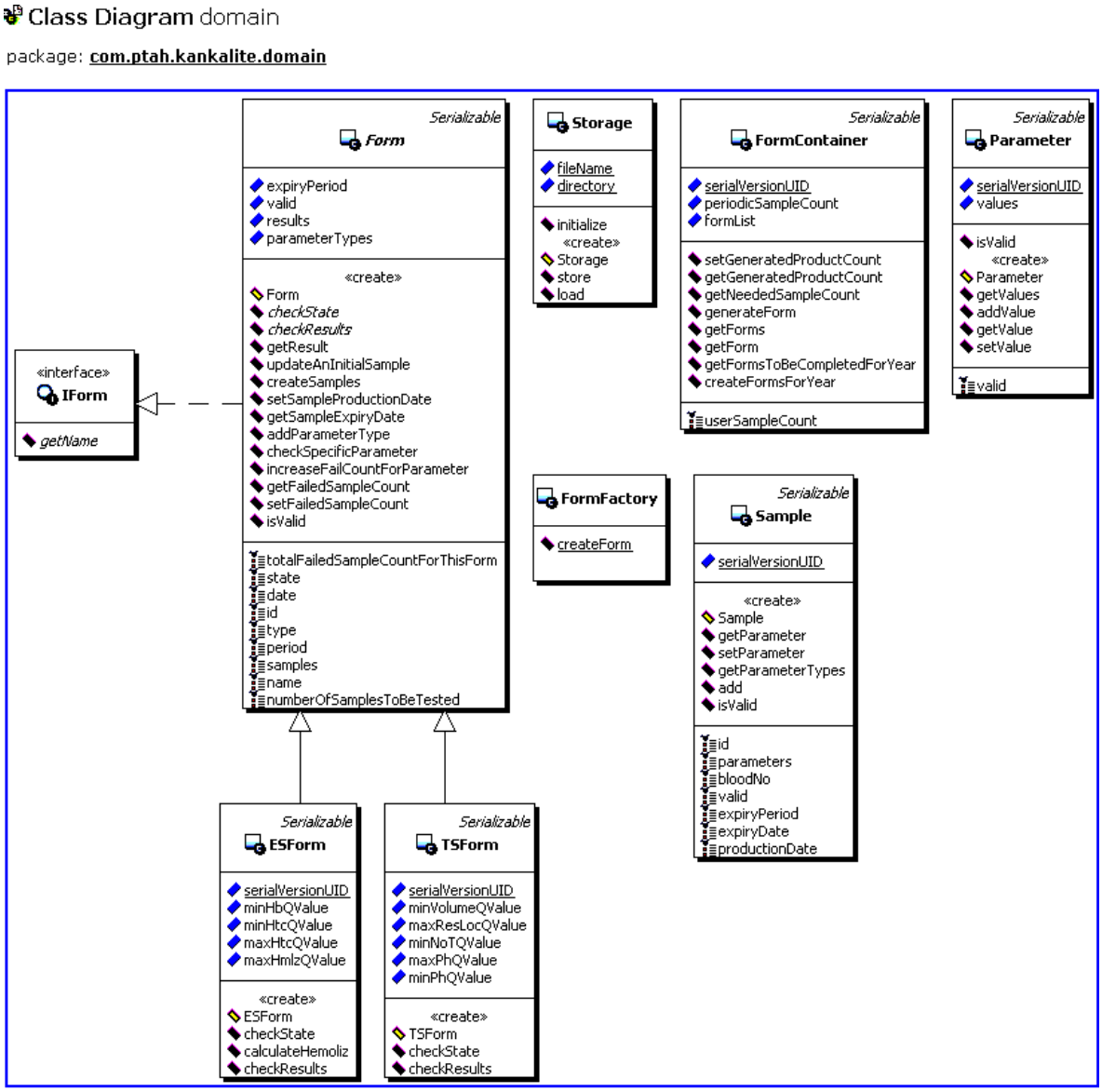


Figure 5.2 The basic UML diagram of the Domain of QC Management Software

5.3. Design of the Classes

Mainly, UML design of the domain gives us four classes and seven objects, which are the instantiations of them.

The Major Classes are “FormContainer”, “Form”, “Sample” and “Parameter”.

Mainly, FormContainer contain Forms. A Form uses samples to contain parameters. Every object “knows” only the function of itself, but not the functions of other objects, depending on the OOP basics.

5.3.1.The Class: FormContainer

The Class “FormContainer” is the main starter of the software. FormContainer is the object, which contains all of the separate Monthly Report Forms. We need two parameters to access the form: FormType and the FormDate.

The number of the qc samples of one month can be a lot; but, in every month of the year, there is only one Form. FormContainer can be expressed mathematically as having 12 x 3 = 36 Form objects (12 of each type of ES, TS, FFP). It has the hashmap of Forms and is just like a big document folder with 36 files in it and has no information written inside about the files themselves.

The Object “FormContainer” has a field named FormList. It is a hashmap containing another hashmap, which is a matrix having an index and data. On the first hashmap, we use formType as an index and the hashmap gives us another hashmap containing the forms of that type. And we give the formDate as the index of a new hashmap and the hashmap gives us the form data that we want as can be seen on Figure 5.3. below.

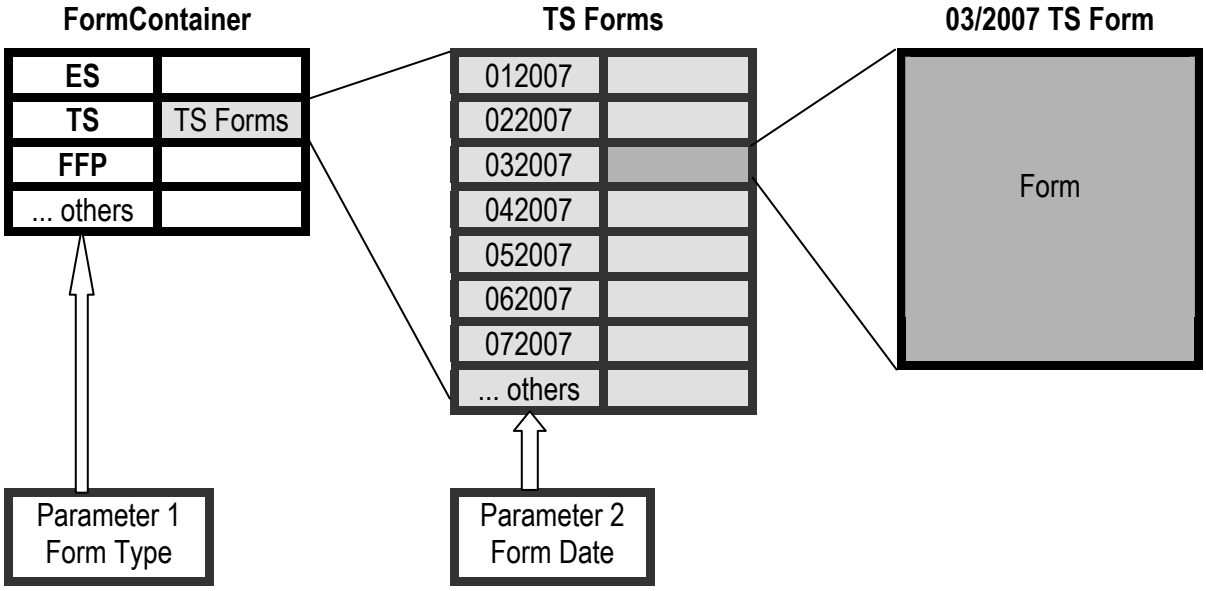


Figure 5.3 Hashmap of FormContainer

FormContainer has its own methods shown on table 5.1.

Table 5.1
Field and method summary of class FormContainer

Field Summary	
int	**userSampleCount**
private Map<FormType,Map<String,Form>>	**formList**
private Map<FormType,Map<String,Integer>>	**periodicSampleCount**
private final static long	**serialVersionUID**
private int	**userSampleCount**

Method Summary	
public void	**createFormsForYear**(FormType type, String year)
public Form	**generateForm**(FormType formType, String period)
public Form	**getForm**(FormType formType, String date)
public Map	**getForms**(FormType formType)
public Map	**getFormsToBeCompletedForYear**(FormType type)
public int	**getGeneratedProductCount**(FormType formType, String date)
public int	**getNeededSampleCount**(FormType formType, String date)
public void	**setGeneratedProductCount**(FormType formType, String date, Integer count)
public void	**setUserSampleCount**(Integer userSampleCount)

FormContainer uses a static factory class: FormFactory. FormFactory, with the parameters formType and formPeriod, creates new objects of Form. But the only instantiation of FormContainer is formContainer hashmap.

The integer we obtain from getNeededSampleCount method is number of needed QC samples. It is calculated with algorithms that are written on the QC Processes of the

blood component. For Example, for the ES Form, the default of this integer number is 4 in case the number of produced ES Components is less than 400 in the blood bank.

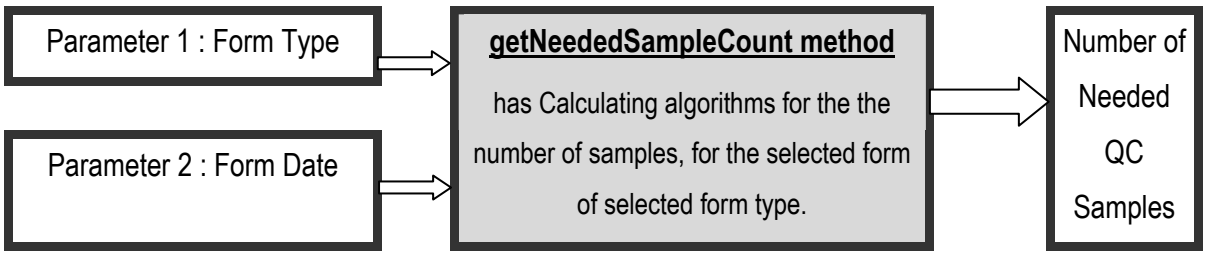


Figure 5.4 GetNeededSampleCount method of the FormContainer

The method "setGeneratedProductCount" sets a number for a form of chosen type.

The method GenerateForm, generates a particular form for a specific month. The input parameters are formType and date (string). With these parameters, it gives a form object. The number of the samples, is the number of needed QC samples for that month and for that type of component. By using the GUI of the software, we input the data of the samples (test results etc.) afterwards.

GetForm method of FormContainer can get the form in the container having the same type and date given to the method as indices of the interlinked hashmaps.

CreateFormsForYear method takes formType and year and creates 12 forms for the year. Parameter "Year" always has 4 digits; this method adds 2 extra digits before this 4 digits to represent each month.

5.3.2. The Class: Form

The object Form is the most valuable object of the list attribute of formContainer. And it is also the most valuable object other than the object "formContainer". In object Form, there are some attributes like the form type, form date, and the samples contained in the form. Class Form has the information of formType of itself (component type).

This object controls its own “state”. There are three states of form object: “Sample absent”, “awaiting test results”, and “done”.

The properties of the class Form are inherited to its extents: TS, ES, and FFP Objects, which have some of the properties of same kind with some differences between, like parameters, number of samples per month, etc. (Extents are not the exact instantiations).

Table 5.2
Method summary of the class “Form”

Method Summary	
public void	**addParameterType**(ParameterType type)
public abstract void	**checkResults**()
protected void	**checkSpecificParameter**(ParameterType type, Integer min, Integer max, Float minQValue,
public abstract void	**checkState**()
public void	**createSamples**()
public Date	**getDate**()
public int	**getFailedSampleCount**()
public long	**getId**()
public String	**getName**()
public int	**getNumberOfSamplesToBeTested**()
public String	**getPeriod**()
public int	**getResult**(ParameterType type)
public Date	**getSampleExpiryDate**(Integer i)
public Map	**getSamples**()
public FormState	**getState**()
public int	**getTotalFailedSampleCountForThisForm**()
public FormType	**getType**()
private void	**increaseFailCountForParameter**(ParameterType type, Parameter parameter)
public boolean	**is¥alid**()
public void	**setDate**(Date date)
public void	**setFailedSampleCount**(Integer failedSampleCount)
public void	**setNumberOfSamplesToBeTested**(Integer numberOfSamplesToBeTested)
public void	**setPeriod**(String period)
public void	**setSampleProductionDate**(Integer i, Date date)
public void	**setState**(FormState state)
public void	**setType**(FormType type)
public void	**updateAnInitialSample**()

At the time of creation of the object form, the default value of its state is “sample absent”, because no one of the test results of QC samples has been input. If the test results of QC samples are entered, but there are still some more absent test results left (to be input at the end of storage period of the component), the state of the form changes to “test awaiting state” from “sample absent”. If even one of the sample data inputs are not written on the form, the state continues to be “sample absent”.

At the end of storage period of the blood component, when the data of new test results are input, the state changes to “done” from “test awaiting”. If the form has a form state of “done”, this form will be saved to the main database of formContainer. And, the only reason for a form to appear on the QC status window of the GUI of the software, (which is on the left side of the main GUI), is having a form state of “sample absent” or “test awaiting”. The forms, which have states of “done” are not shown on the listbox on the QC status window of the GUI.

If the storage period of samples on a form are close or have already passed, on the alarm listbox of the QC status window of the GUI, a date and an alarm code “TR” appears (test result input is requested).

Table 5.3
Field summary of the class “Form”

Field Summary	
FormType	**type**
String	**period**
Map<Integer,Sample>	**samples**
long	**id**
Date	**date**
FormState	**state**
int	**totalFailedSampleCountForThisForm**
int	**numberOfSamplesToBeTested**
String	**name**
private Date	**date**
protected int	**expiryPeriod**
protected long	**id**
protected String	**name**
private int	**numberOfSamplesToBeTested**
List<ParameterType>	**parameterTypes**
private String	**period**
protected Map<ParameterType,Integer>	**results**
private Map<Integer,Sample>	**samples**
private FormState	**state**
protected int	**totalFailedSampleCountForThisForm**
private FormType	**type**
private boolean	**valid**

When the new forms of the year are created, the forms belonging to the months that must have samples and test results, have an alarm code of “SR” (Sample inputs are requested). The forms belonging to the months after the present day of the usage of the

software have no SR or TR alarm codes. During that session of usage of the software, a new form created, has always a code of “NF” (new form created). And, if the state of the form changes to “done” in that session, this form changes to a QC status code of “C” (completed).

In the object Form, there is a parameter named “numberOfSamplesToBeTested”. This parameter is set by the object formContainer.

The object Form has three different types of lists: two of them are hashmaps and the third is a plain list. The plain list parameter is called “property types”. The values of this parameter are the types of object parameters, that are called “parameter types” such as haematocrit, hemolysis, haemoglobin, pH value, platelet count, etc.

The class Form has the Hashmap of samples which has the results of QC tests. It has a method of calculation of the number of Samples, according to the Produced Component type; and the number of production, and it has the list of parameters, to set to the object “Sample”.

It has a method of sample creation, a method for asking for a specific parameter; and has some methods of differentiating the “passed” and “failed” QC samples according to the QC test results, and a method of giving information, when asked.

Every extended form object, (for example ES), has the information on how to control the minimum and the maximum values of parametric results of tests.

5.3.3. The Class : Sample

The class “sample” has the ability to contain the secondary test results of the QC parameters to be checked at the end of the storage period, as well as the QC parameters to be checked at the beginning of the process, such as haematocrit value. This is done with a hashmap having a name “parameters”. So, in summary, FormContainer is the object

containing forms; forms contain samples; samples contain parameters; and parameters contain minimum and maximum values of the parameters of QC testing.

Table 5.4
Method and field summary of the class "Sample"

Field Summary	
Map<ParameterType,Parameter>	**parameters**
int	**id**
boolean	**valid**
int	**expiryPeriod**
int	**bloodNo**
Date	**productionDate**
Date	**expiryDate**
private int	**bloodNo**
private Date	**expiryDate**
private int	**expiryPeriod**
private int	**id**
private Map<ParameterType,Parameter>	**parameters**
private Date	**productionDate**
private final static long	**serialVersionUID**
private boolean	**valid**

Method Summary	
public void	**add**(ParameterType type, Integer index, Float value)
public int	**getBloodNo**()
public Date	**getExpiryDate**()
public int	**getExpiryPeriod**()
public int	**getId**()
public Parameter	**getParameter**(ParameterType type)
public Map	**getParameters**()

The object Sample, has a Unit Number, an Expiry Date, an Expiry period, a Production Date; and has a hashmap of Parameters. The object Sample" has no information on these parameters, just contains them. This condition gives us the ability to create a sample with any number of any parameters. Object can call Parameter "set", and "get" methods and can check expiry date and unit number of a sample.

5.3.4. The Class: Parameter

The Instantiation (implementation of a blueprint of an object) of this class has an hashmap, a multi dimensional matrix, which gives it the chance to store any number of parameter checks of any parameter. For example ES Hematocrit level is controlled two times, firstly on the 1st, and secondly on the 42nd day of sampling. This object could even

create a parameter check of hematocrit 12 different times during its storage period, instead of 2. This is an important feature to be flexible in future developments [21] of the QC management software.

This class has methods for controlling the validity of the results, and has methods of giving any value that has been set to a parameter (test results) to the user, who uses GUI of the software. And the user can set any value to a parameter from GUI (inputting of the test results).

Table 5.5
Method Summary of The class "Parameter"

Method Summary	
public void	**add**(ParameterType type, Integer index, Float value)
public int	**getBloodNo**()
public Date	**getExpiryDate**()
public int	**getExpiryPeriod**()
public int	**getId**()
public Parameter	**getParameter**(ParameterType type)
public Map	**getParameters**()
public Set	**getParameterTypes**()
public Date	**getProductionDate**()
public boolean	**isValid**()
public void	**setBloodNo**(Integer bloodNo)
public void	**setExpiryDate**(Date expiryDate)
public void	**setExpiryPeriod**(Integer expiryPeriod)
public void	**setId**(Integer id)
public void	**setParameter**(ParameterType type, Parameter parameter)
public void	**setParameters**(Map parameters)
public void	**setProductionDate**(Date productionDate)
public void	**setValid**(Boolean valid)

6. THE USE AND GUI DESIGN OF THE SOFTWARE

The QC management software has functions of creating monthly sample record forms of ES-AD, TS, and FFP components, inputting or editing the data of the samples and test results of first and second QC processes to the forms.

It has GUI designs to find the forms related with the specific samples which stored until their expiry date and to print reports, including the approvals of QC Processes performed outside the QC Laboratory.

All of the elements of menu and the windows on the GUI are designed to make the management of the monthly “forms” which contain samples easy and practical. There are menu objects, and shortcut buttons for direct access to Monthly Component Databases, Samples, and Reports.

Almost all of the default values are written on a text file which is used during the starting of the software. So the most of the default parameters are only changed by only firmware update in this version of the software. But the design of the model, allows us to make connections with the GUI setup menu to access the defaults.

On the GUI, there are two bars of buttons: the “Menu Bar”, and the “ShortCut Buttons Bar”, and two main windows: the “QC Status Window” (consisting of three sub windows: “QC Form List”, “QC Alarm List”, and “Alarm Definitions”), and the “Component Form Entry”, window (consisting of three sub windows: “Sample Data Input”, “QC Form General Data Input”, and “QC Form Report General View”).

Main GUI design of the QC Management Software is shown on Figure 6.1.

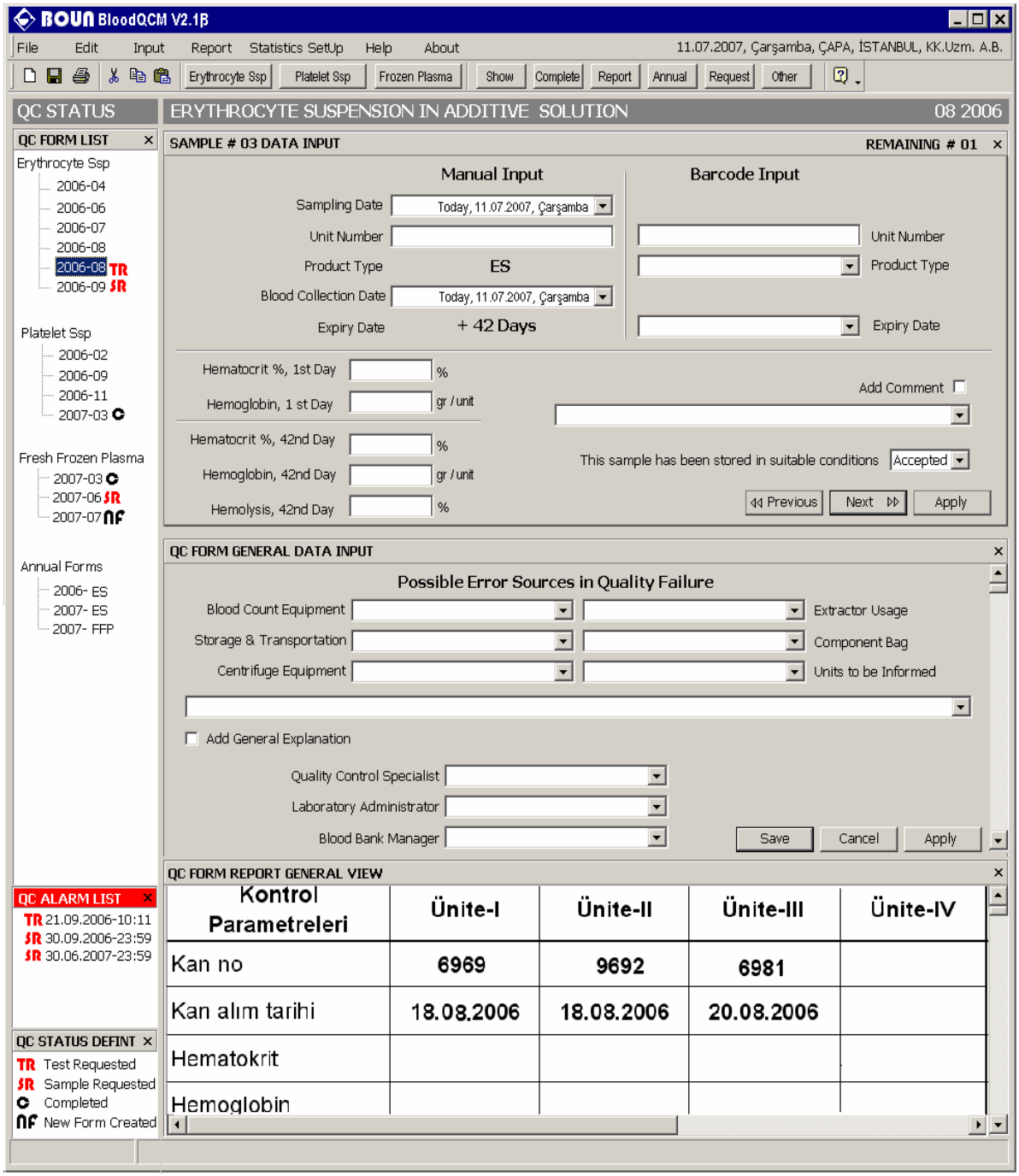


Figure 6.1 Main Gui design of the QC Management Software.

6.1. General Menu of the Software

General menu access on the software is done with menu tree, on the Menu Bar of the software GUI. Menu bar is shown on the figure 6.2. The Menu tree and its functions are given on Appendix C.

Figure 6.2 The Menu Bar.

Some of the buttons gives a "N/A" when pressed, as it is shown in the menu tree in Appendix C. These are the buttons with the functions, which has been modeled to be added to domain code of the software, but have no place in the GUI design of this version.

6.1.1. Set Up Menu

On the setup menu, there are different kinds of user setups. Default values of minimum-maximum scientific and quality requirement values of parameters of testing results are among them. The software allows the data entry of each parameter, between these scientific minimum and maximum values. The criteria, for a parameter tested to be marked as "quality passed" is the QC requirement of that parameter. For this, the value must be between Quality Min, and Quality Max values on the default list [1, 14]. Default unit list, and defaults table of scientific and QC min-max values used in this software are given in Appendix D.

Some of the main functions is put on the setup menu to avoid the unauthorized personnel to make changes in the database. Creation of the set of forms of the Year is a typical example of this. This procedure is performed once a year to create blank forms of a selected year, or the year of that QC session.

6.2. Form and Sample Data Entry of the Software

Data entries to the forms and samples are done with the menu tree, or the ShortCut Buttons of ES, TS, and FFP. Each of them, when clicked, opens its own window on the data entry side (right side) of the GUI. Each of the "component form entry" window, has its own parameters result entry boxes, on their "Sample Data Input" subwindow.

During the data entry, which is performed from the "Sample Data Input" sub window, the results and unit numbers can be seen real-time on "QC Form Report General View" sub window.

6.2.1. ES-AD ShortCut Button

This button opens the "sample data input" subwindow (Figure 6.3), or tab of the "ES component form entry" window and lets the user to input data of samples of ES (to the last ES form he has been in).

To change the ES form, user must use the "direct form access" button, or the "QC Form list".

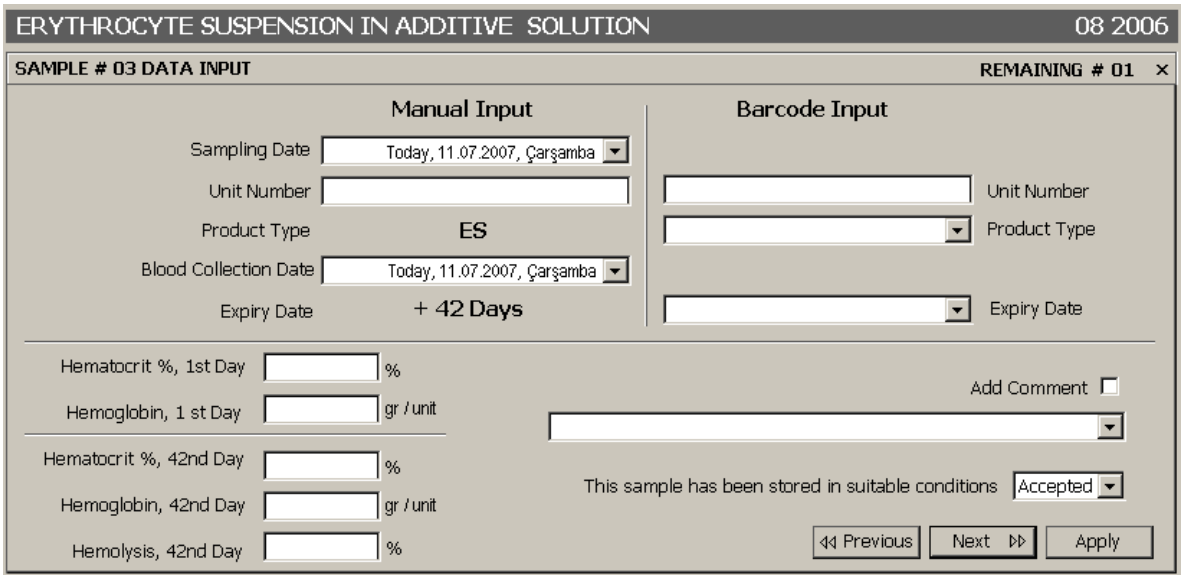


Figure 6.3 Sample Data Input sub window, of the ES Data Entry Window.

6.2.2. FFP ShortCut Button

This button opens the “sample data input” subwindow (Figure 6.4), or tab of the “FFP component form entry” window and lets the user to input data of samples of FFP (to the last FFP form he has been in).

To change the FFP form, user must use the “direct form access” button, or the “QC Form list”.

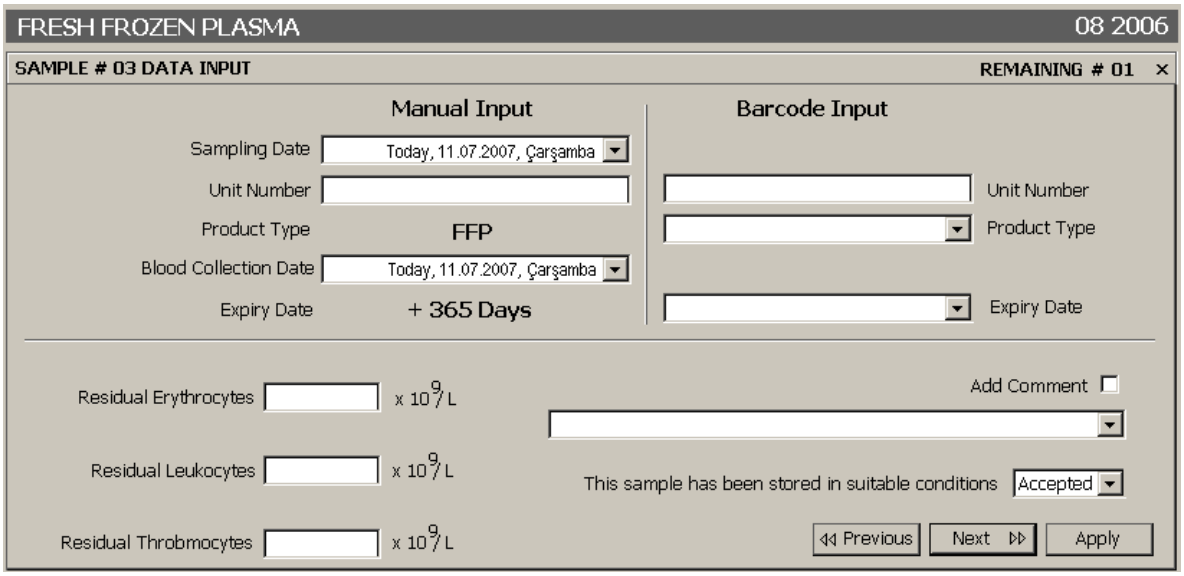


Figure 6.4 Sample Data Input sub window, of the FFP Data Entry Window.

6.2.3. TS ShortCut Button

This button opens the “sample data input” subwindow (Figure 6.5), or tab of the “TS component form entry” window and lets the user to input data of samples of TS (to the last TS form he has been in).

To change the TS form, user must use the “direct form access” button, or the “QC Form list”.

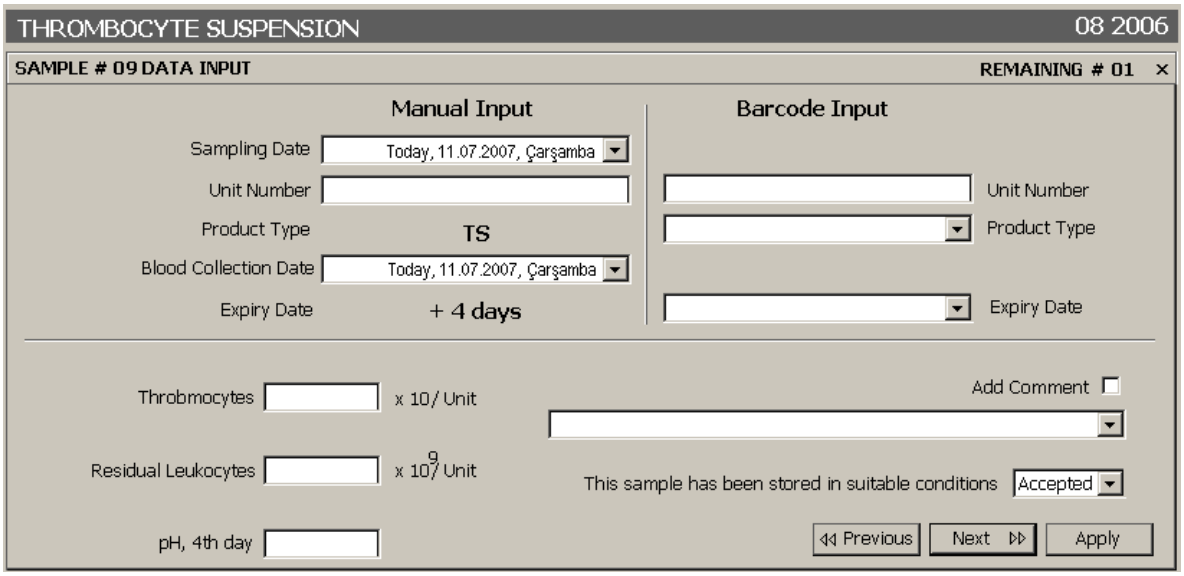


Figure 6.5 Sample Data Input sub window, of the TS Data Entry Window.

6.2.4. An example of a data entry procedure of TS samples

Before the data entry of the samples, the user must input the number of the produced components, on that month for the calculation of the number of samples to be used in QC Process of that component (TS). QC specialist enters the value, via the tab of "Number of Samples" of TS Component Form, seen on the Figure 6.6. The Number of samples can be limited by the user, to a limited value, instead of 1% of produced components.

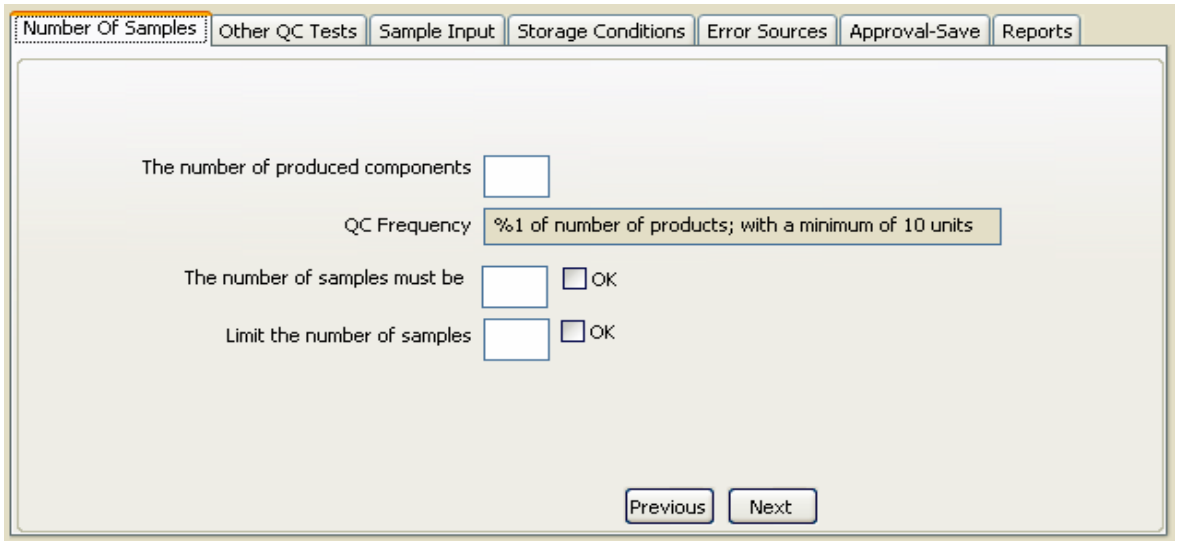


Figure 6.6 Number of samples input tab, of the TS Data Entry Window.

Then with the “Next Button”, “Other QC Tests” tab comes (Figure 6.7.). This is the tab of the GUI, for the approvals of QC tests, which are performed outside the QC Laboratory, during preparation phase of that component. As well, with the “Storage Conditions tab”, approval of “suitable storage” by the QC Specialist is done. For these approvals, QC specialist uses the records of process laboratory and other laboratories, and refrigerator degree measurement records of storage department. The name of QC Specialist can be selected from a combo-box of GUI easily.

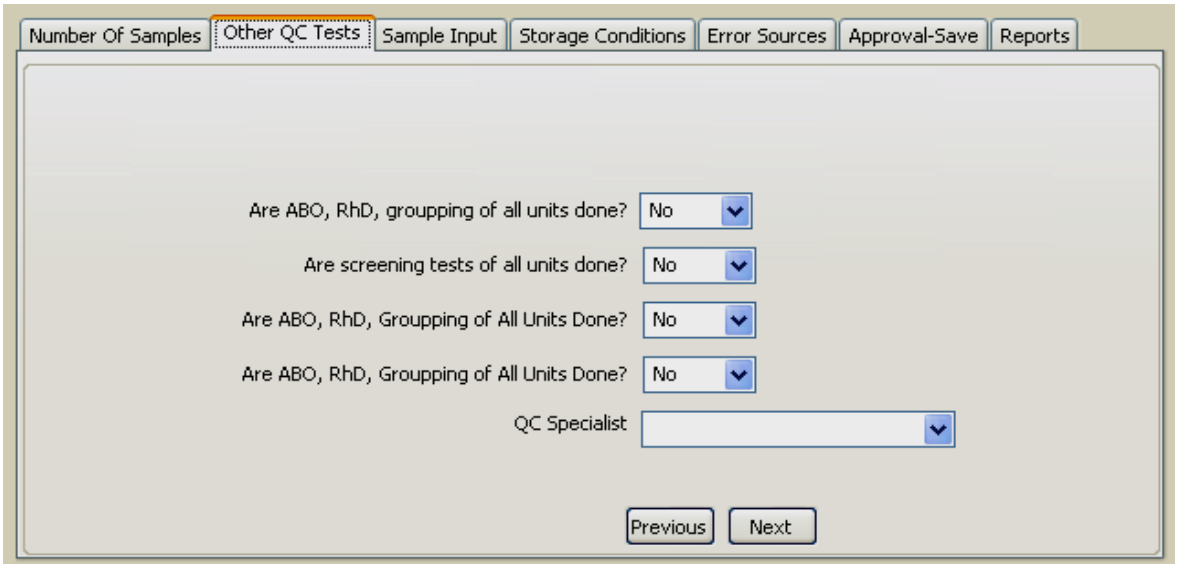


Figure 6.7 Other QC Tests Approval sub window, of the TS Data Entry Window.

With the “Sample Data Input” subwindow of the “TS Form Entry” window, QC specialist enters the test results of selected samples, one by one. The Entries are shown on the “QC Form General View” subwindow, shown below Figure 6.8.:

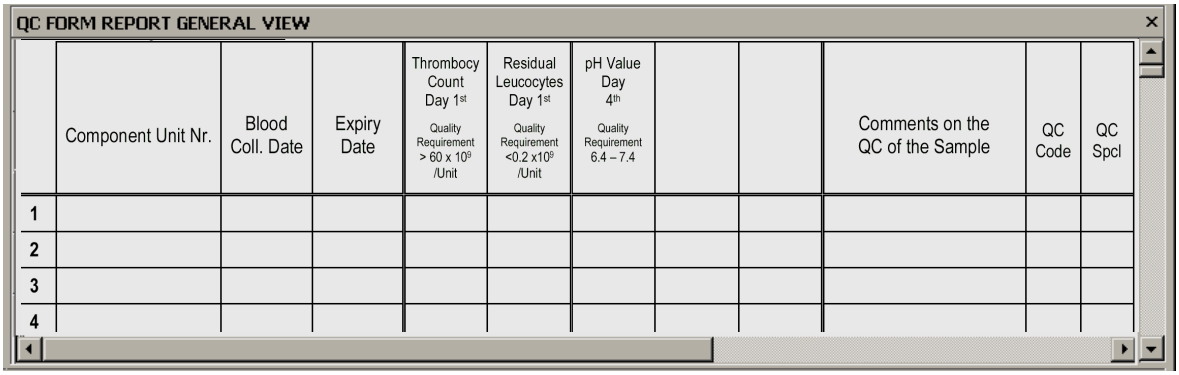


Figure 6.8 QC Form Report General View sub window, of the TS Data Entry Window.

After that, error sources, in Quality defects tab (or window) comes (Figure 6.9.), and then Administrative Approvals (Figure 6.10.), and Save Functions take place.

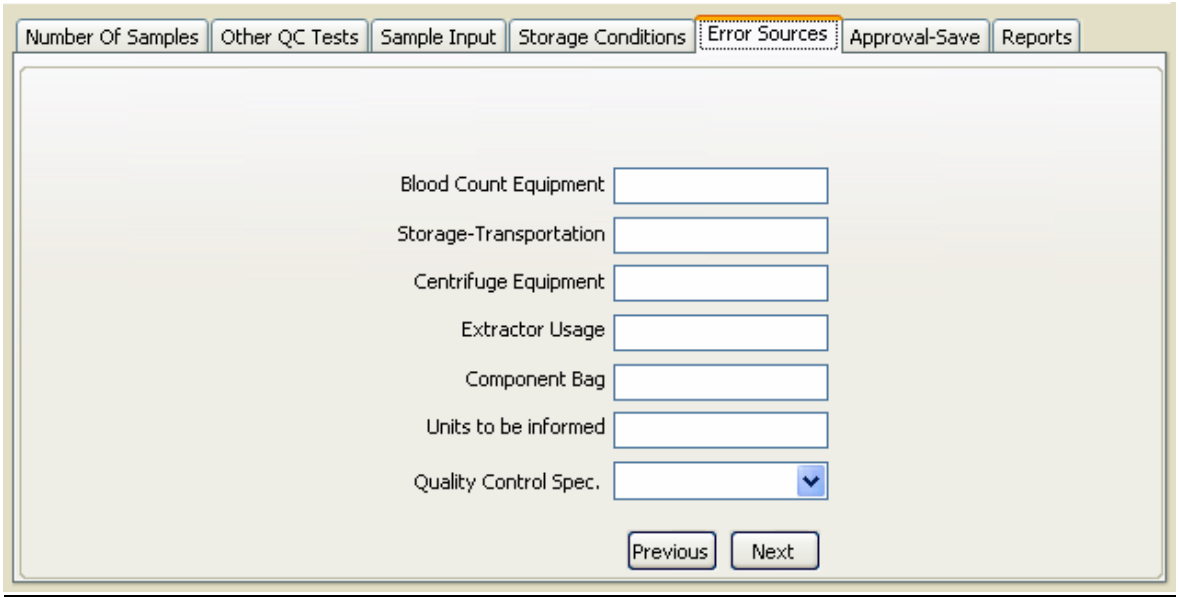


Figure 6.9 QC Form Error Sources Input sub window, of the TS Component Data Entry Window.

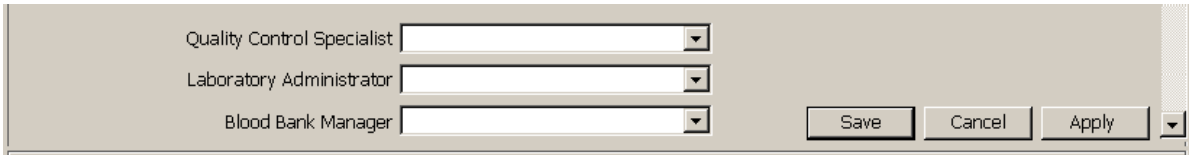


Figure 6.10 QC Form Administrative Approval sub window, of the TS Data Entry Window.

With these procedures, the entry phase of a QC process is completed. If another session of entry is needed, QC specialist can access the form, via form date of the QC form or using the unit number of the sample, to be tested on the second QC Process. After finding the specific form, specialist can edit or finish the inputting phase.

After this procedure, the form status, becomes “C”, on QC Status window. And, next day, the date of that form will disappear from that window.

6.3. General Form and Sample Access of the Software

The Access to the Forms and Samples saved in the hashmap of FormContainer, can be done with the menu tree, or ShortCut Buttons. Short Cut Buttons are shown on the Figure 6.11:

Figure 6.11 ShortCut Buttons

6.3.1. Direct Form Access Button

Direct Form Access Button has two functions. One of them is finding the form from "form type" & "form date"; and other function is finding the form from the QC alarm list. This Function is used for finding the QC Forms of a selected year and month. When pressed, a window appears with the input areas of form type form date. By entering the data, the form of selected component type comes as the editing window, to allow the data entry or to edit the sample results.

The Direct Form Access window is shown on Figure 6.12. below:

Figure 6.12 The Direct Form Access Window for ES

New Created forms of that year appear on the QC Form list subwindow of the QC Status window. To Access a form by using the QC Form List, is limited with the forms of the year, and is limited with the forms, which does not have adequate number of samples or which does not have adequate number of QC test results of these samples. By pressing the form of selected date, user can go to a specific form.

The forms have QC status types, as it was stated in the modeling explanations, like TR (test required), SR (sample required), NF (new created form), and C (completed-approved form). The status code is written next to the form date on the QC Status window. And the definitions are given in another window. These QC Status are put on the QC Form List, to indicate the forms, which need QC sample data or QC test results. QC Status Window, and its sub windows are shown as Figure 6.13. below:

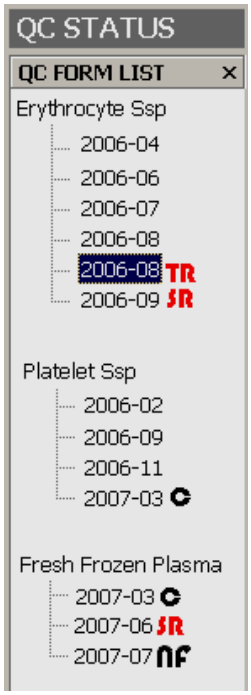


Figure 6.13 QC Status Window and and its sub windows.

With the alarm function, not the form type or form date, but the “sample data inputting date” or “test result inputting date” and hour is given to QC Specialist.

If the software session is on the same day of the sample data or test input date just after the starting of the software, the alarm window (in red) appears. The Alarm window can appear 0 to 6 days before the event according to the user setup. Default value is 1 day.

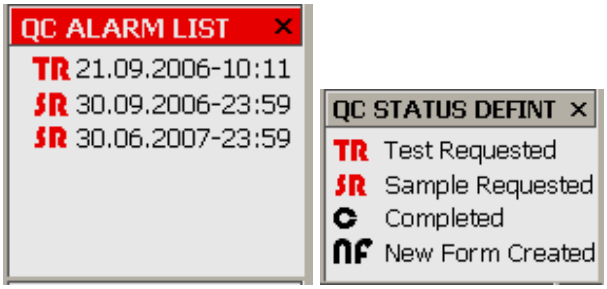


Figure 6.14 Alarm and Definition Windows

6.3.2. Direct Sample Access Button

Direct Sample Access Button has two functions. One of them is to find the form, which is the related with a specific Sample, from its “unit number” and “component type”; and the other function is find that from of that specific sample from a selection list.

When the Direct Sample access button is clicked, a window appears with the entry areas of component type and unit number. By entering these data, the form, which contains that sample comes as window, allowing the QC specialist to enter or edit data, or test result of that sample, and of other samples on that form.

This Function helps QC specialist to access the records of the component samples, which are taken from storage department for the second part of QC Process. The Direct access can be done, by using simple barcode entries (the component type and unit number).

The barcode reading results table of label samples of different components used in blood banks in Turkey, is given in Appendix G. In the development and GUI design phases, these readings were used in the standardization of the component and date recognition.

The Direct Form Access Window is shown below Figure 6.15.

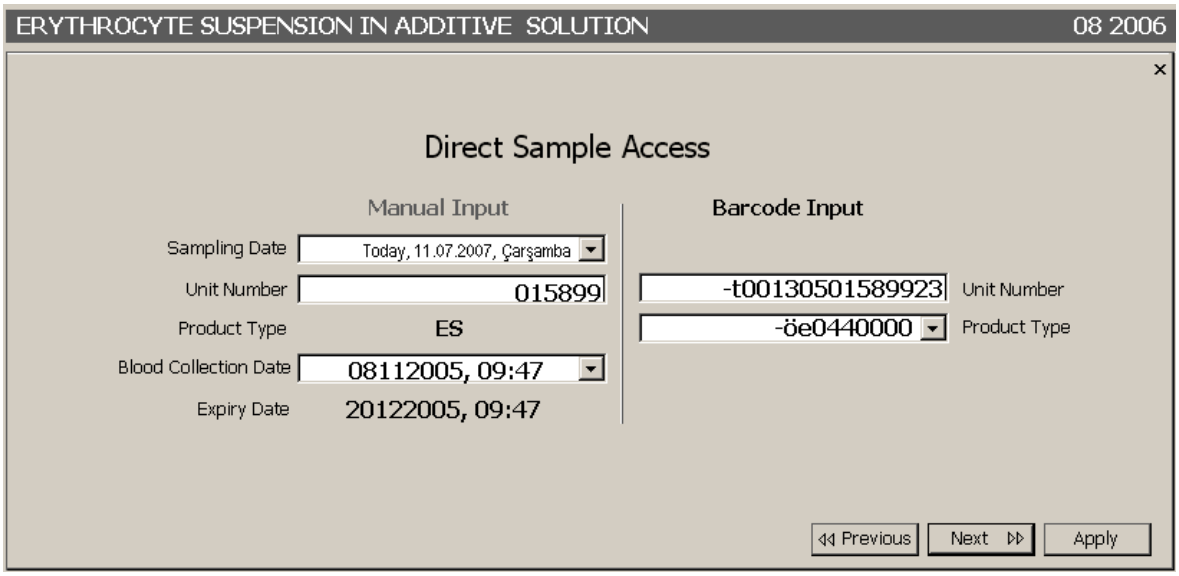


Figure 6.15 The Direct Sample Access Window for Barcode Input.

6.4. Printing Reports

User can print monthly reports, monthly component request forms and annual summations. Email and Share Buttons, which are put on the menu, are for future improvement of the software.

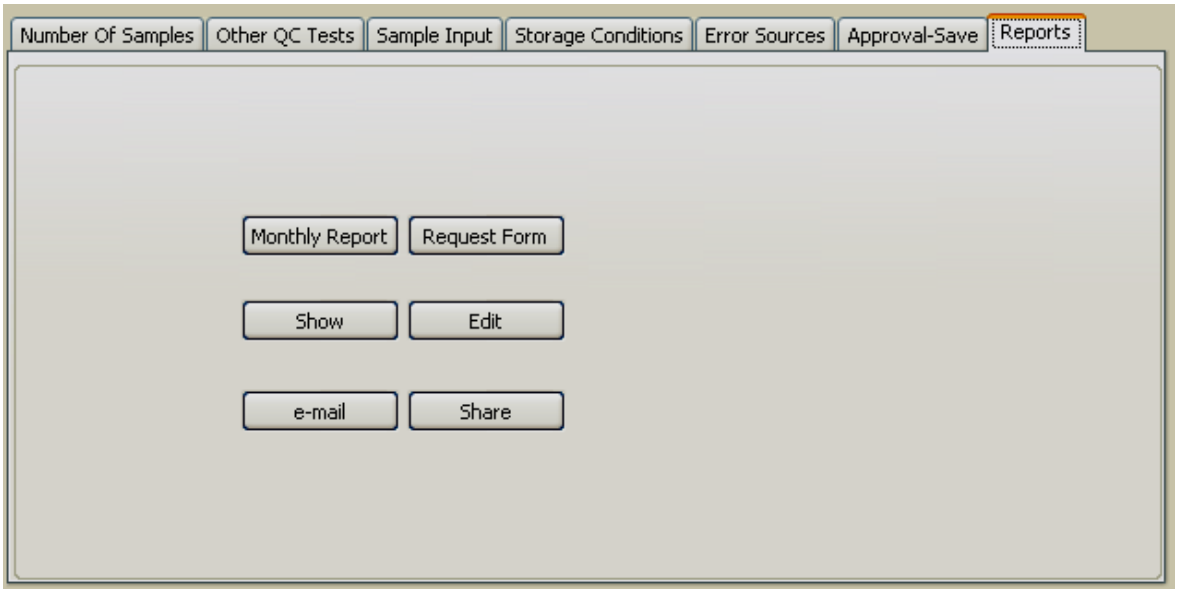


Figure 6.16 Report tab of the TS Component Data Entry Window.

6.4.1. Monthly Report

A “Report” is a collection and an interpretation of data and approvals of QC. As mentioned before, a report has: general information on the Report, the QC requirements, QC sampling frequencies, test results of samples, approvals, explanations on the error sources. The report samples are given in Appendix E.1, E.2 and E.3.

6.4.2. Request Form

Request Form can be printed automatically from the “Report Tab” of the “Component Form Entry” Window. It has the list of the transportation dates of the components, from storage department to QC laboratories, and vice versa. All of these transportations must have an administrative approval and the initials of that administrator. The request form sample is given in Appendix F.

6.4.3. Annual Summations

The user selects two dates (limited with 12 months) to prepare an annual sum table of the QC results. The Annual Report Samples are put on Appendix E.4, and E.4.1.

7. DISCUSSION AND CONCLUSION

The Management Software of QC of blood products offers laboratories “robotic-like” accuracy with a human touch. With these, QC Laboratories and preparation processes can be automated to improve accuracy and reliability [22].

The quality assurance program should use the electronic data processing systems that affect product quality by affecting the QC Process by accelerating the improvement of the laboratory's performance and services.

According to the experience that this project gave, it is obvious that, development of a software, can be done with the collaboration of the QC Specialist and the developer under the supervision of an expert in Biomedical Science possessing an analytical mind.

And future projects in this area in Biomedical Science should be done on the analysis of QC processes taking place in the Blood Banks and QC Requirements in preparation of Components to make changes in the applications and to develop safer QC processes, the validation of the QC software of the same kind in Blood Banks and QC Laboratories to develop safer and more reliable software and analysis of the requirements of the country for a hemovigilance system for the traceability of the blood components and for adding a QC approval system into it [10].

7.1. The Necessity and Benefits of QC of Blood Products

It is inevitable, even in the best blood banks and in the best of laboratories, that some materials will fail some of the tests and a strict protocol should be drawn up showing action to be taken in such an eventuality.

All staff in the blood bank service should be trained to accept quality control as a welcome and necessary part of everyday work. It is useful to cultivate a positive attitude

towards the detection and correction of errors though the emphasis is on the prevention of problems and the production of blood components [3].

In addition, some research has shown that computerized software automation of blood banks and their QC procedures reduce the cost of Components, the time period of preparation, the wastage percent of blood products [1], the error rates in the procedures of preparation and QC of the preparations [23], and, as stated before, by reducing the risk associated with producing defective components and ensuring a greater productivity through reduced manufacturing downtime, QC Procedures improve the productive life of expensive equipment of Blood Banks.

In Turkey, the data of error rates was unavailable. And, the show of improved QC performance and low error incidence rates would require a very large prospective study [5, 23].

Haemovigilance networks should embody operational linkages between hospitals, and blood banks. There is a need to coordinate efforts at QC Laboratory, Hospital, Blood Bank and National Authority levels, and to develop effective collaboration with experts, developers, and institutions working for QA of Components and Blood Safety at different levels.

7.2. Future work on development of QC managing software

This project is limited by the QC of components ES-AD, TS, and the FFP. And, it is useful, practical, and adequate for the Blood Banks of Turkey for the usage of the software to be started immediately. In a blood bank which has different QC automations, (e.g. like ones in Europe) the software modules of QC of ABO Grouping, QC of FactorVIIIc tests, QC of test kits (reagents), and QC approvals of refrigerator temperature stability, which are done several times a day should also be developed.

In future projects, the development of the software can be done as explained below:

Software development can be done to manage not only QC of Components ES-AD, TS, and the FFP, but also all kinds of blood components, including validations of the QC laboratory itself: ABO Grouping, TestKits, Storage, Equipment. The modeling of the software allows us to do so and adds different kinds of Component Forms and Parameters.

It is not advisable to link the data of the QC results with the data of main managing system of the blood bank directly because of the possibility of unauthorized data entry and problems during development and validation of the software [5]. But, the QC Software can use an internal network to go a specific data storage area to get specific information or a text file without interrupting the main system and without interference regarding the blood components, test results, etc. So, the QC Specialist will be able to trace the component from a donation to a recipient.

The results and "done" approvals of laboratory procedures that take place outside the QC Laboratory can also be transferred automatically to the QC Managing Software via the internal network of the blood bank. This will save time, decrease the number of bureaucratic procedures, and allow the QC Specialist to follow the procedures in real time. This module can also exercise administrative control over the QC procedures. This way, the records of QC procedures can be signed by the supervisor [11] and this can allow the printing of documents or reports according to the laws of Türkiye.

The computer of the QC Lab can communicate with the Blood Counting Equipment and other test Equipment via the RS232 serial port with one of the protocols of: ASTM (E 1381), HL7, ASC X12, UN/EDIFACT, which are usually used in laboratory equipment [1]. Even if the protocol is not apparent, the transferred data can be copied to a text file and the QC Managing software picks up the information from this text file. For this procedure, Windows Software of Hiperterminal.exe or the Activex Communication Tools of Visual Languages can be used [1]. This way, the automation of the QC System will be complete.

The QC Software can allow us to create a datasharing link via the Internet between the centre and its branches. The results are collated and accuracy scores can be determined; the results can be communicated to all participating laboratories (in coded or uncoded form according to local agreements) in order to enable each laboratory to compare its own

quality standard [2] with that of a large number of other laboratories including the reference centre selected by the Blood Bank Administration. With this software module, international performance testing and statistical evaluations can also be done.

APPENDIX A. QUALITY REQUIREMENTS

A.1 Quality Requirements of Whole Blood

Table A.1
Table of Quality Requirements of Whole Blood [3].

	Parameter to be checked	Quality requirement	Frequency of control	Control executed by
1	ABO, Rh D	Grouping	All units	grouping lab
2	anti-HIV 1&2	Negative by approved screening test	All units	screening lab
3	HBsAg	Negative by approved screening test	All units	screening lab
4	anti-HBc (when required)	Negative by approved screening test	All units	screening lab
5	anti-HCV	Negative by approved screening test	All units	screening lab
6	Syphilis (when required)	Negative by screening test	All units	screening lab
7	anti-CMV (when required)	Negative by screening test	As required	screening lab
8	anti-HTLV I&II (when required)	Negative by screening test	All units	screening lab
9	Volume	450 ml ± 10% volume excluding anticoagulant	1% of all units with a min. of 4 units/month	processing lab
10	Haemoglobin	Minimum 45 g/unit	4 units per month	QC lab
11	Haemolysis at the end of storage	<0.8% of red cell mass	4 units per month	QC lab[21]

A.2 Quality Requirements of ES-AS

Table A.2
Table of Quality Requirements of ES-AS [3].

	Parameter to be checked	Quality requirement	Frequency of control	Control executed by
1	ABO, Rh D	Grouping	All units	grouping lab
2	anti-HIV 1&2	Negative by approved screening test	All units	screening lab
3	HBsAg	Negative by approved screening test	All units	screening lab
4	anti-HBc (when required)	Negative by approved screening test	All units	screening lab
5	anti-HCV	Negative by approved screening test	All units	screening lab
6	Syphilis (when required)	Negative by screening test	All units	screening lab
7	Anti-CMV (when required)	Negative by screening test	As required	screening lab
8	anti-HTLV I&II (when required)	Negative by screening test	All units	screening lab
9	Volume	to be defined for the system used	1% of all units	Processing lab
10	Hct	0.50 - 0.70	4 units per month	QC lab
11	Haemoglobin	minimum 45 g/unit	4 units per month	QC lab
12	Haemolysis at the end of storage	<0.8% of red cell mass	4 units per month	QC lab[21]

A.3 Quality Requirements of TS

Table A.3
Table of Quality Requirements of TS [3].

	Parameter to be checked	Quality requirement (specification)	Frequency of control	Control executed by
1	ABO, Rh D	Grouping	All units	grouping lab
2	anti-HIV 1&2	Negative by approved screening test	All units	screening lab
3	HBsAg	Negative by approved screening test	All units	screening lab
4	anti-HBc (when required)	Negative by approved screening test	All units	screening lab
5	anti-HCV	Negative by approved screening test	All units	screening lab
6	Syphilis (when required)	Negative by screening test	All units	screening lab
7	anti-CMV (when required)	Negative by screening test	As required	screening lab
8	anti-HTLV I&II (when required)	Negative by screening test	All units	screening lab
9	HLA or HPA (when required)	Typing	as required	HLA lab
10	Volume	> 40 ml	all units	Processing lab
11	Platelet Count*	> 60×10^9/single unit equivalent	1% of all units with a minimum of 10 units per month	QC lab
12a	Residual leucocytes* Before leucocyte depletion a. prepared from PRP b. prepared from buffy-coat	< 0.2×10^9/single unit equivalent < 0.05×10^9/single unit equivalent	1% of all units with a minimum of 10 units per month	QC lab
12b	Residual leucocytes** After leucocytes depletion	< 0.2×10^6/single unit equivalent	1% of all units with a minimum of 10 units per month	QC lab
13	pH measured*** (+22 °C) at the end of the recommended shelf life	6.4 to 7.4	1% of all units with a minimum of 4 units per month	QC lab

* These requirements shall be deemed to have been met if 75% of the units tested fall within the values indicated.
** These requirements shall be deemed to have been met if 90% of the units tested fall within the values indicated.
*** Measurement of the pH in a closed system is preferable to prevent CO2 escape. Measurements may be made at another temperature and converted by calculation for reporting pH at +22 °C.

A.4 Quality Requirements of FFP

Table A.4
Table of Quality Requirements of FFP [3].

	Parameter to be checked	Quality requirement (specification)	Frequency of control	Control executed by
1	ABO, Rh D	Grouping	All units	grouping lab
2	anti-HIV 1&2	Negative by approved screening test	All units	screening lab
3	HBsAg	Negative by approved screening test	All units	screening lab
4	anti-HBc (when required)	Negative by approved screening test	All units	screening lab
5	anti-HCV	Negative by approved screening test	All units	screening lab
6	Syphilis (when required)	Negative by screening test	All units	screening lab
7	anti-HTLV I&II (when required)	Negative by screening test	All units	screening lab
8	Volume	stated volume ± 10%	all units	processing lab
9	Factor VIIIc	≥ 70 IU per 100 ml	every two months. a) pool of 6 units of mixed blood groups during first month of storage. b) pool of 6 units of mixed blood groups during last month of storage.	QC lab
10	Factor VIIIc	Average (after freezing and thawing): ≥ 70 % of the value of the freshly collected plasma unit	Every 3 month 10 units in the first month of storage**.	QC lab
11	Residual cells*	red cells: < 6.0 × 109/l leucocytes: < 0.1 × 109/l platelets: < 50 × 109/l	1% of all units with a minimum of 4 units/month	QC lab
12	Leakage	no leakage at any part of container e.g. visual inspection after pressure in a plasma extractor, before freezing and after thawing	all units	processing and receiving laboratory
13	Visual changes	no abnormal colour or visible clots	all units	"

* Cell counting performed before freezing. Low levels can be achieved if specific cellular depletions are included in the protocol.
** The exact number of units to be tested could be determined by statistical process control

APPENDIX B. QC PROCESSES

B.1 QC Process of ES-AD Component

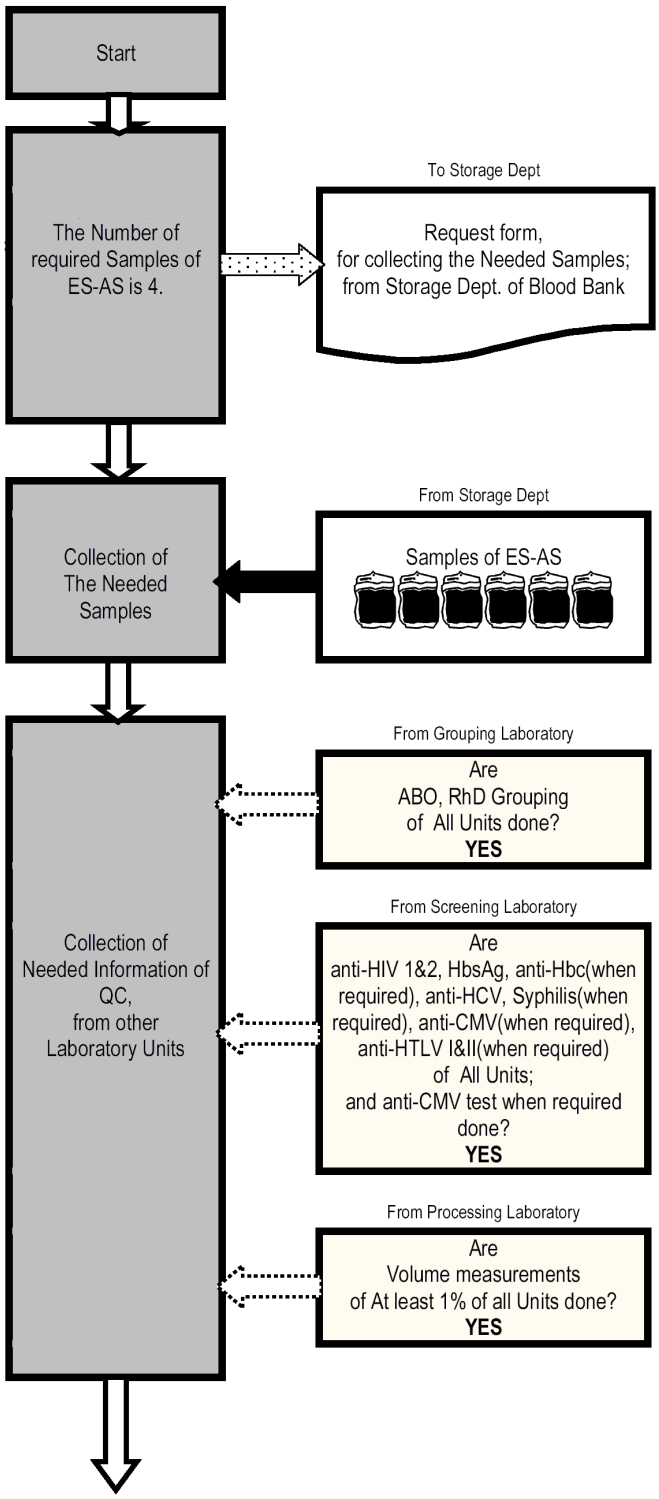


Figure B.1 Diagram of QC Process of ES-AD Component

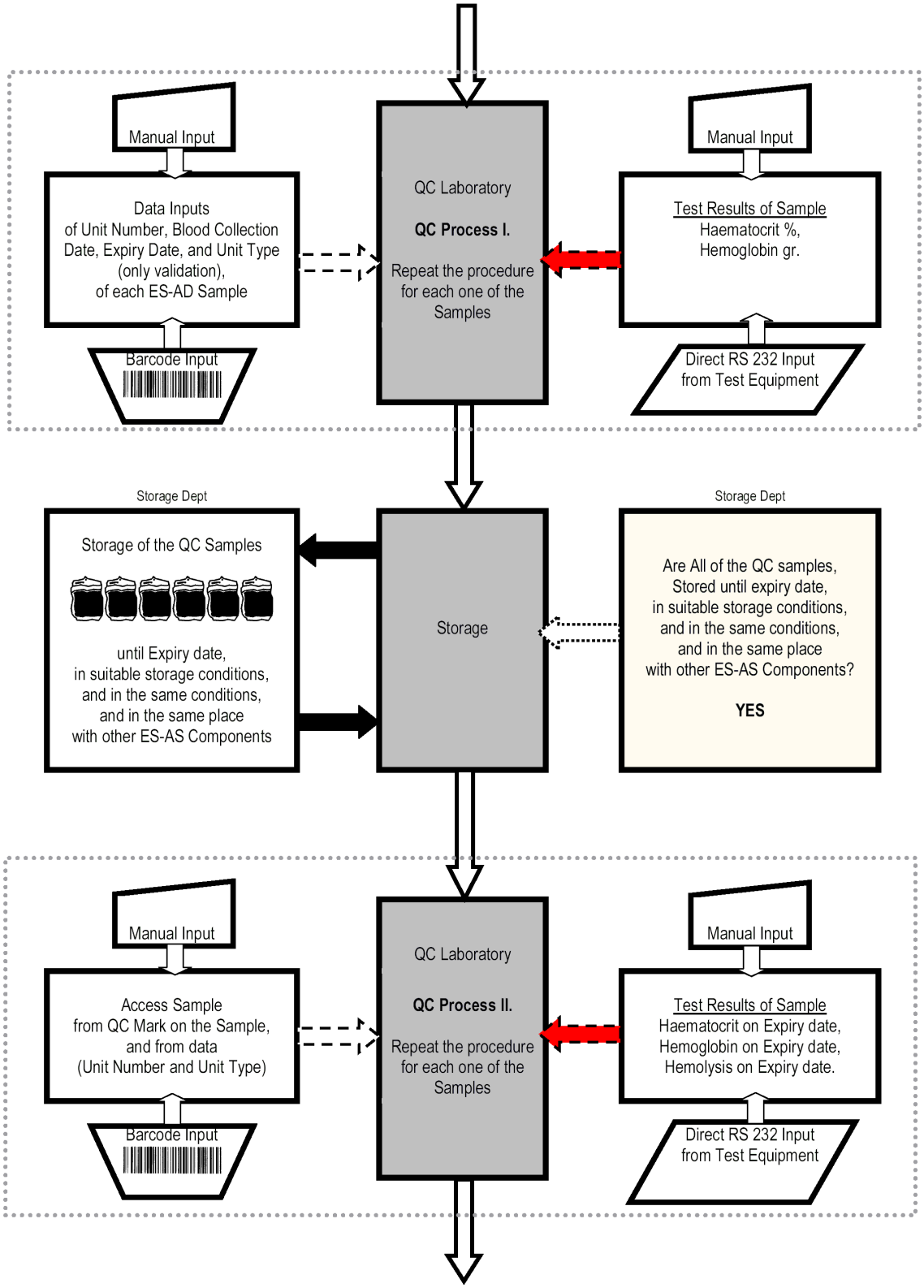


Figure B.1 Diagram of QC Process of ES-AD Component

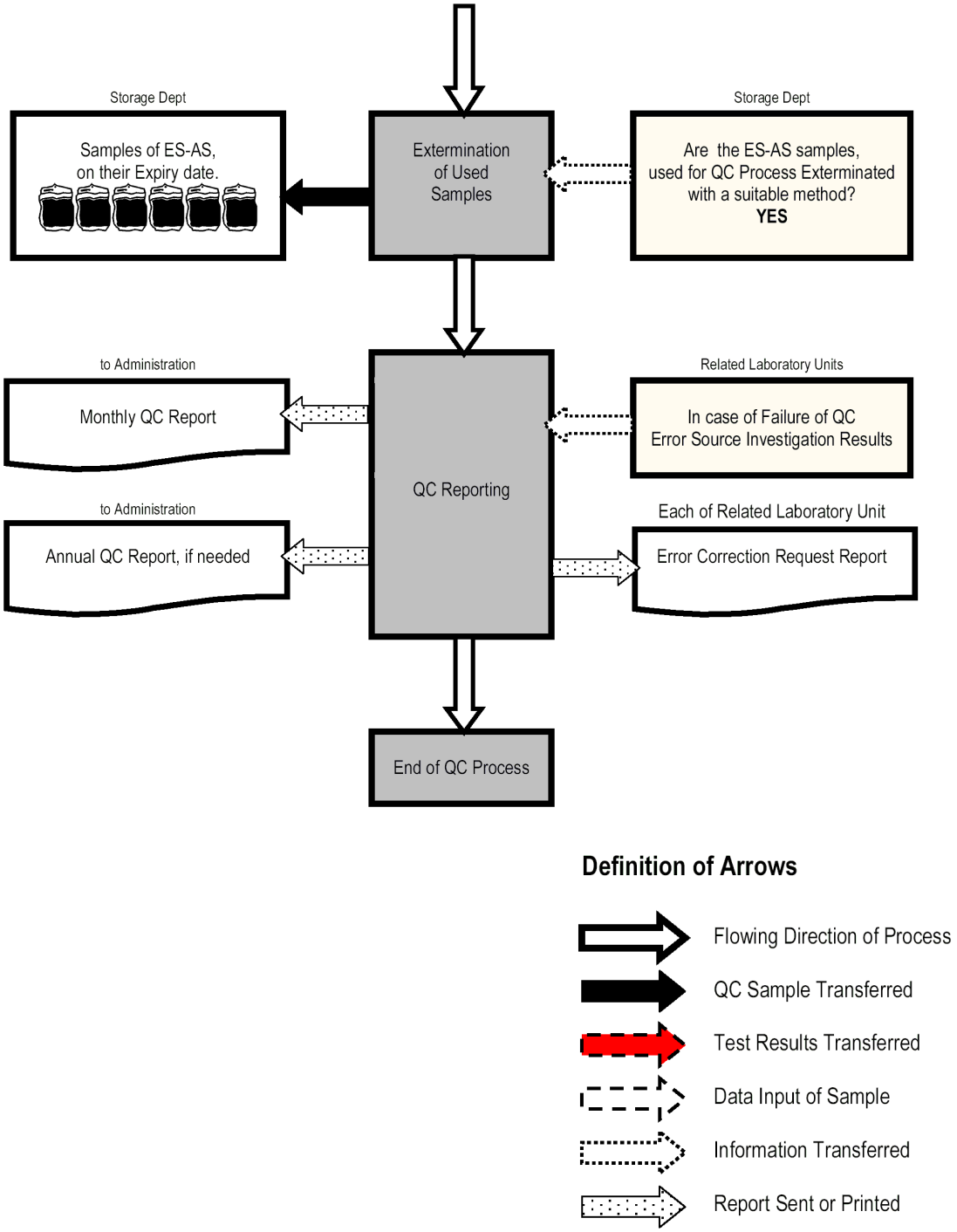


Figure B.1 Diagram of QC Process of ES-AD Component

B.2 QC Process of TS Component

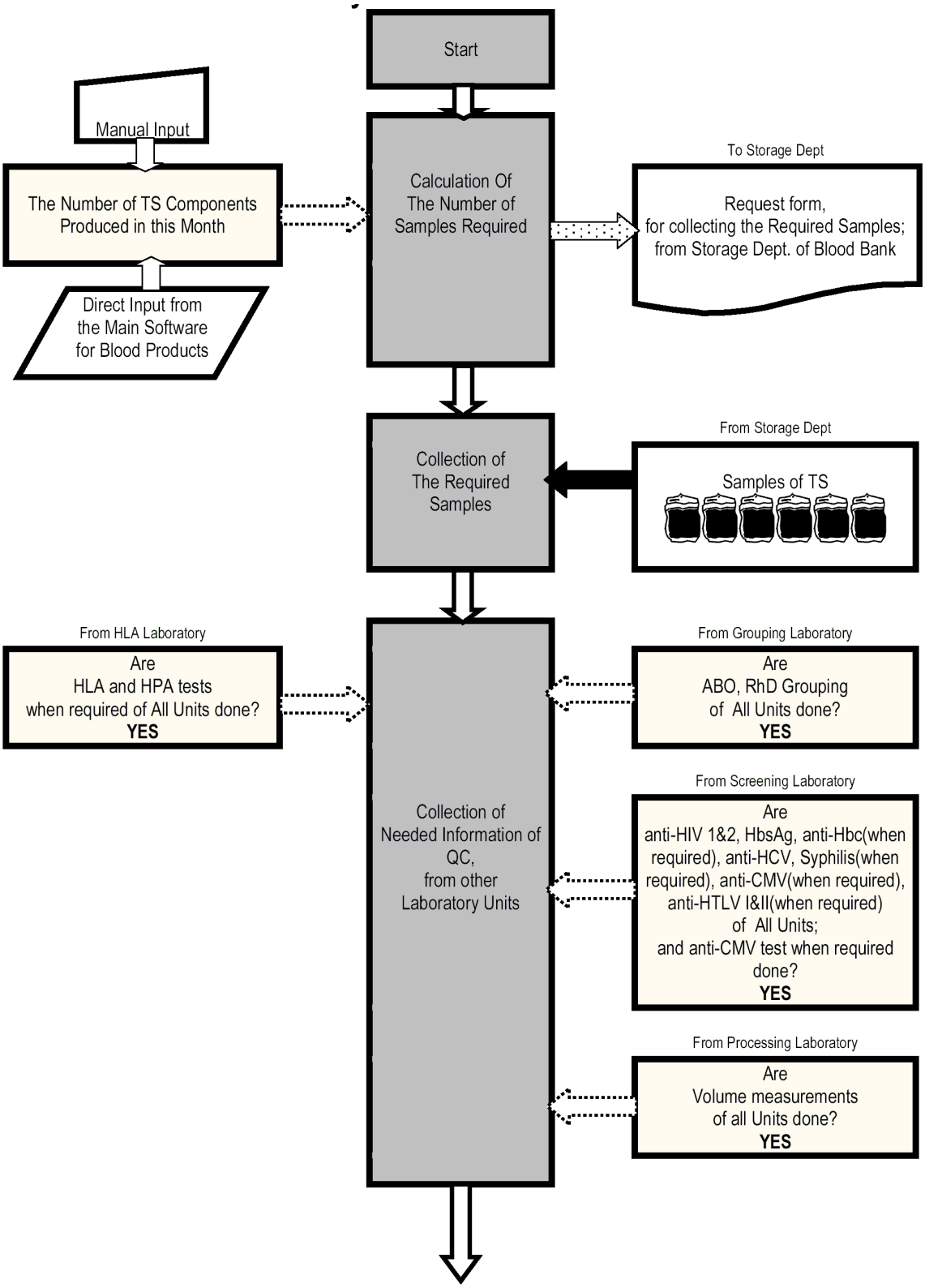


Figure B.2 Diagram of QC Process of TS Component

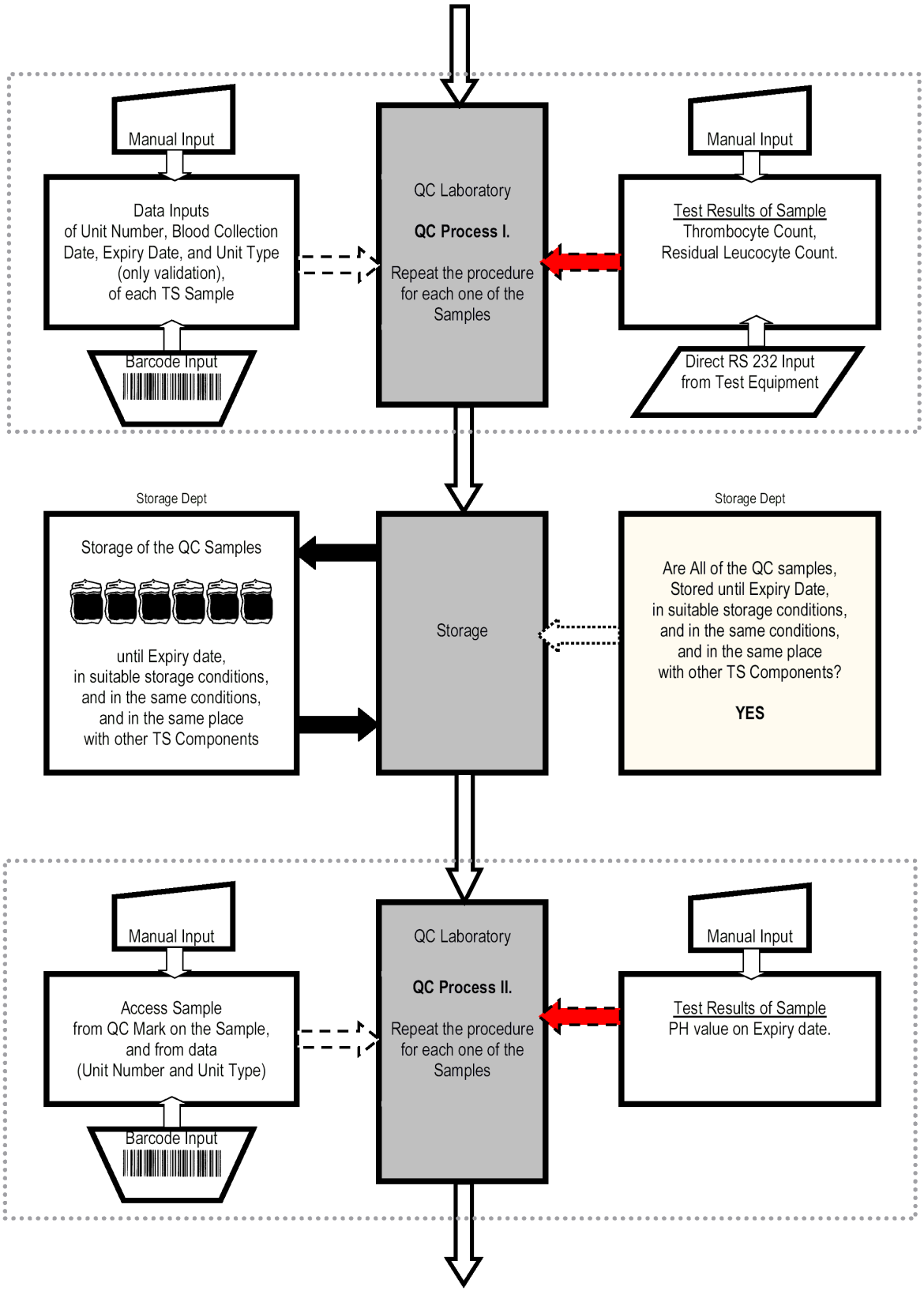


Figure B.2 Diagram of QC Process of TS Component

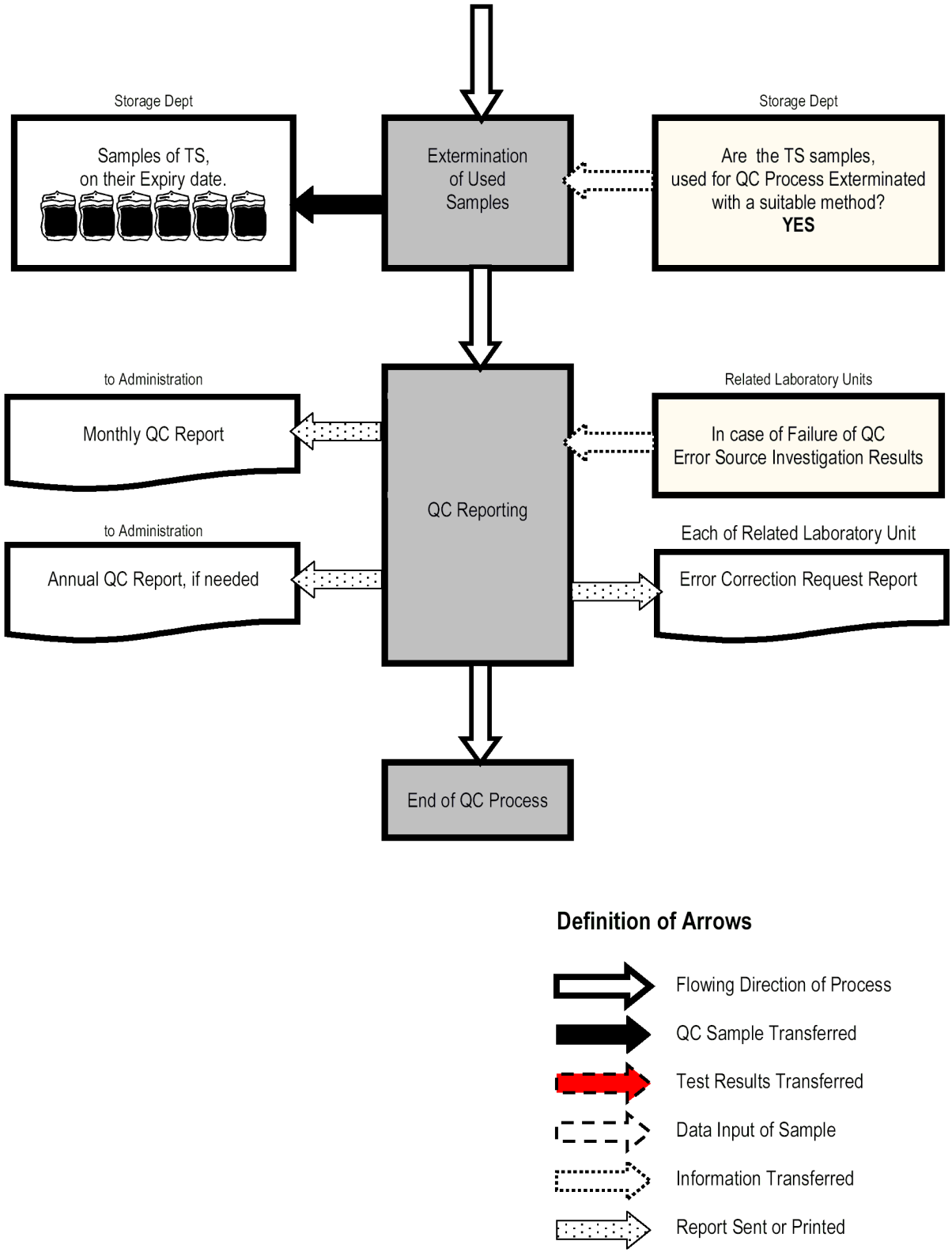


Figure B.2 Diagram of QC Process of TS Component

B.3 QC Process of FFP Component

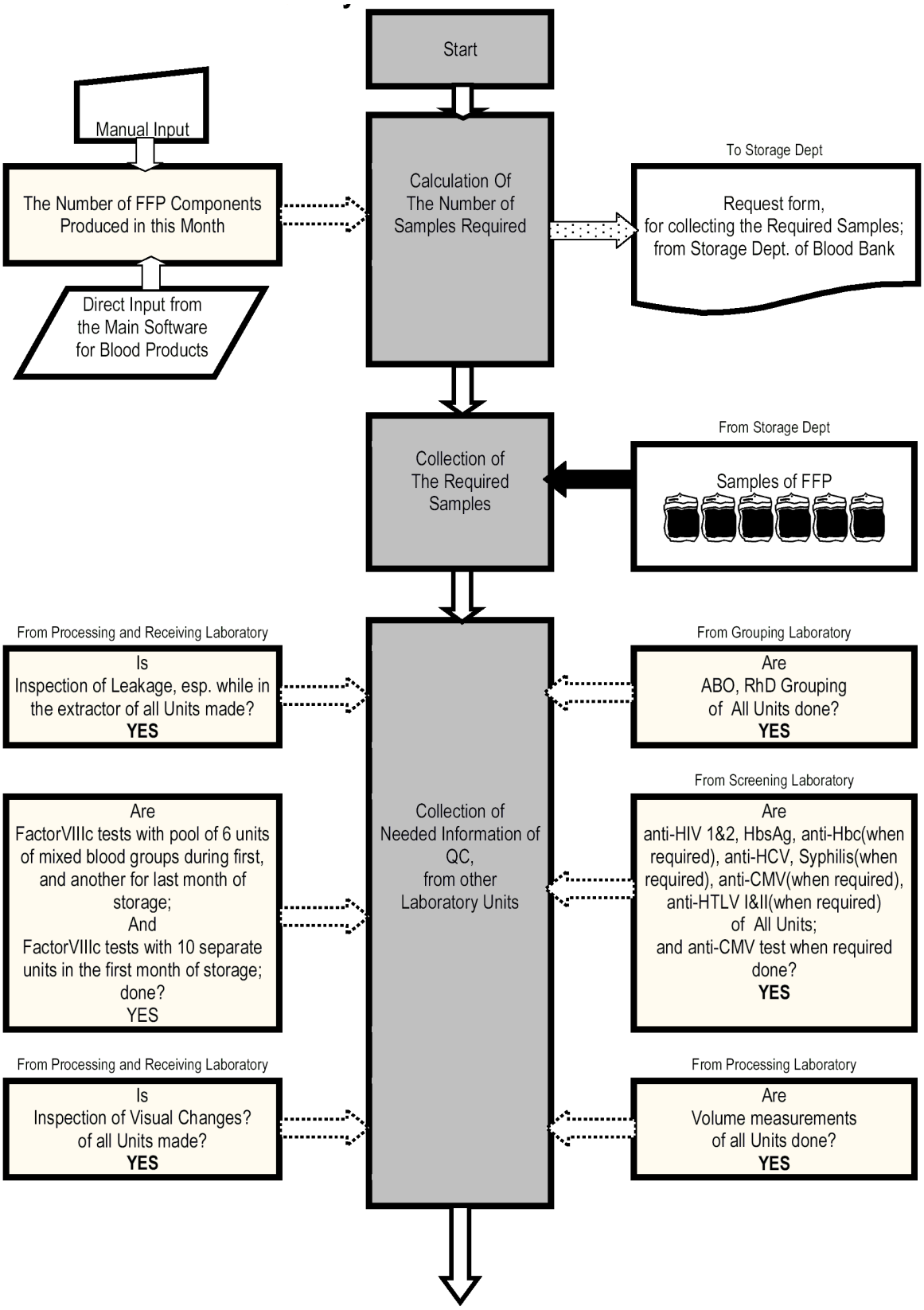


Figure B.3 Diagram of QC Process of FFP Component

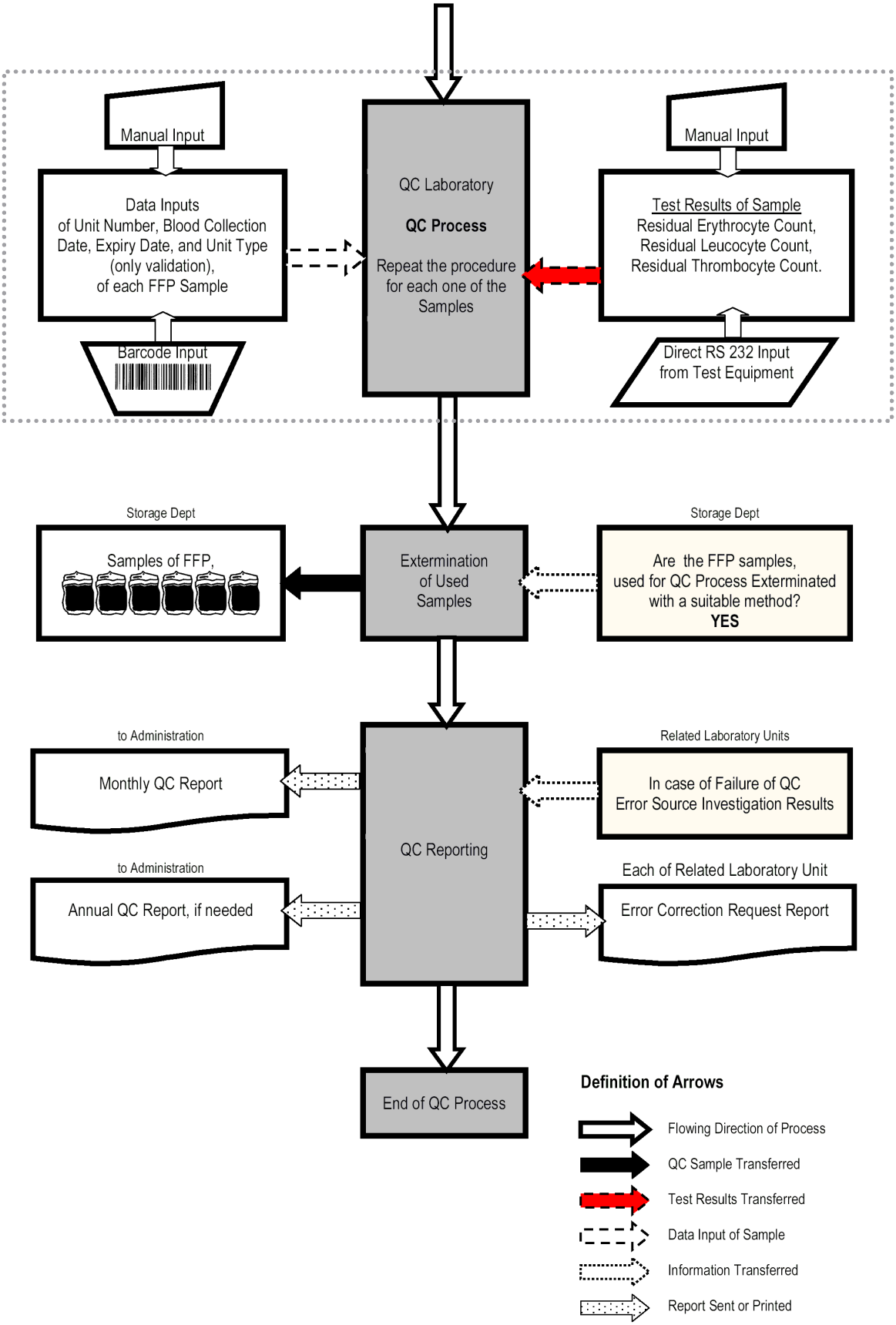


Figure B.3 Diagram of QC Process of FFP Component

APPENDIX C. THE MENU TREE OF THE SOFTWARE

Table C.1
Menu tree of the Software.
* Default Values . ** Some of the buttons which are modeled to be added to domain code of the software, have no place in the GUI design of this version. They are given as "N/A" in the menu tree list below.

Tree	**Function** (Function is stated at the left column; unless explained)
File	
Open	Opens a database file of another source
Save	Saves the database file to another source
Save& Exit	
Find Form	Shortcut to Direct Form Access
Find Sample	Shortcut to Direct Sample Access
Print Report	Prints Selected Monthly Report
Print Annual Report	Prints Selected Annual Report
Print Request Form	Prints Selected Request Form
Edit	
Undo	
Redo	
Cut	
Copy	
Paste	
Select All	
Input	
Form	
ES	Shortcut to ES Sample Input
TS	Shortcut to TS Sample Input
FFP	Shortcut to FFP Sample Input
Other N/A**	
Sample	Shortcut to Direct Sample Access
Summation Formula N/A**	N/A**
Statistical Formula N/A**	N/A**
Report	
Monthly	
ES	Shortcut to ES Report Output Menu
TS	Shortcut to TS Report Output Menu
FFP	Shortcut to FFP Report Output Menu
Other N/A***	
Annual Summation	Shortcut to Annual Summation Menu
Annual Statistics N/A**	N/A**
Request Form	

ES	Shortcut to ES Request Form Output Menu
TS	Shortcut to TS Request Form Output Menu
FFP	Shortcut to FFP Request Form Output Menu
Statistics N/A	**N/A****
SetUp	
Create Forms of the Year	*Creates New Forms of the Year*
*Language N/A**	***N/A****
QC Setup	
QC Frequency	N/A**
QC Criteria	
QC Min-Max Scientific Values	
QC Used Parameter Units	N/A**
Error Sources N/A**	N/A**
Reports Setup	
Monthly Report	
Show QC Requirements*	Puts QC Requirements area to the report
Show QC Frequency*	Puts QC Frequency area to the report
Show QC Samples*	Puts QC Sample Results area to the report
Show QC Error Sources*	Puts QC Error Investigations area to the report
Show QC Explanations*	Puts QC Explanation area to the report
Show QC Approvals*	Puts QC Approval area to the report
Annual Summation Report	N/A**
Statistical Report	N/A**
E-Mail Setup	N/A**
Barcode Deffaults	
Manual *	Selects Manual Side of the Input Screens
Barcode Reader	Selects Barcode Side of the Input Screens
Administratives List	
Add Name	
Delete Name	
Password N/A**	N/A**
Add Password	
Change Password	N/A**
Return to Deffaut Values	
Help	
User Manual	
Help Index	
On Restrictions of B Version	Explanations on the B Version
About	
Copyright	
Register N/A**	N/A**

APPENDIX D. DEFAULT VALUES OF THE SOFTWARE

Table D.1
Default unit list of different parameters used in the GUI of the software, next to the test results.

#	Unit String
1	%
2	gr/unit
3	10^9/L
4	10^9/unit
5	Ph
6	(Blank)

Table D.2
Default values of parameters of QC requirements and scientific values recorded in the software.

Definition of the Parameters to be Checked	#	Testing Day	Scientific Min Value	Scientific Max Value	Quality Min Value	Quality Max Value	Unit	Qualifiying Percantage required
Erythrocyte Suspension Haematocrit % 1st Day	1	0	30	90	50	70	%	100
Erythrocyte Suspension Haemoglobin 1st Day	2	0	10	120	45	100	gr/unit	100
Erythrocyte Suspension Haematocrit % 42nd Day	3	42	30	100	55	80	%	100
Erythrocyte Suspension Haemoglobin 42nd Day	4	42	10	120	45	100	gr/unit	100
Erythrocyte Suspension Haemolysis % 42nd Day	5	42	0	40	0	0.8	%	100
Fresh Frozen Plasma Residual Erythrocyte, 1st Day	6	0	0	30	0	6.0	10^9/L	100
Fresh Frozen Plasma Residual Leucocytes	7	0	0	30	0	0.1	10^9/L	100
Fresh Frozen Plasma Residual Thrombocyte	8	0	0	250	0	50	10^9/L	100
Thrombocyte Suspension Residual Leucocytes	9	0	0	30	0	0.2	10^9/unit	100
Thrombocyte Suspension Thrombocyte	10	0	30	200	60	130	10^9/unit	75
Thrombocyte Suspension Ph Degree, 4th Day	11	4	4	10	6.4	7.4	PH	100

APPENDIX E. QC REPORT SAMPLES

E.1 Monthly ES in AD QC Report

MONTHLY QUALITY CONTROL REPORT

BOUN BloodQCM V2.1B
Reporting Module

BLOOD BANK : ISTANBUL

MONTH :

COMPONENT : ERYTHROCYTE SUSP. IN ADD. SOL.

DATE :

QC Frequency	4 Units / Month
Total Number of Components Produced in this Month :	-
Nr. of Samples according to 1 % of Total Nr. of Comp. Produced in this Month :	-
Instead of 1% of Total Nr. of Comp., The number of Samples is limited to	4
The number of Samples of this Month is :	4
ABO and RhD Grouping of All Units are done.	YES
All of the Screening tests required are done.	YES
Volume measurements of at least 1% of all Units are done	YES
Volume measurements of all Units are done	-
HLA and HPA tests when required of All Units are done.	-
Inspection of Visual Changes and Leakage in the Extractor of all Units is made.	-
Factor VIIIc tests with pooled or separate units are done by qc.	-
All of these Samples have been stored, under suitable conditions.	YES
And with the same conditions, as the other same type Comp; before and during testing.	YES
All of these samples, have been exterminated with a suitable method.	YES

QC Laboratory Test Results

	Component Unit Nr.	Blood Coll. Date	Expiry Date	Htc % Day 1st Quality Requirement 50 – 70 %	Hb Day 1st Quality Requirement > 45 gr./ unit	Htc % Day 42nd Quality Requirement 50 – 70 %	Hb Day 42nd Quality Requirement > 45 gr./ unit	Hemolysis Day 42nd Quality Requirement <0.8% mEryt	Comments on the QC of the Sample	QC Code	QC Spcl
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											

Possible Error Sources

Laboratory Blood Count Equipment :		
Transportation & Storage Conditions :		
Centrifuge Equipment :		
Extractor Usage :		
Component Bag :		
Laboratory Units to be Informed :		

Other Explanations :

	Quality Control Specialist	Laboratory Administrator	Blood Bank Manager
Name			
Signature			

Figure E.1 Monthly ES in AD QC Report

E.2 Monthly TS QC Report

MONTHLY QUALITY CONTROL REPORT

BOUN BloodQCM V2.1B
Reporting Module

BLOOD BANK : ISTANBUL

MONTH :

COMPONENT : THROMBOCYTE SUSP

DATE :

QC Frequency	**% 1 of all units with a Minimum of 10 Units / Month**
Total Number of Components Produced in this Month :	
Nr. of Samples according to 1 % of Total Nr. of Comp. Produced in this Month :	
Instead of 1% of Total Nr. of Comp., The number of Samples is limited to	
The number of Samples of this Month is :	**10**
ABO and RhD Grouping of All Units are done.	**YES**
All of the Screening tests required are done.	**YES**
Volume measurements of at least 1% of all Units are done	**-**
Volume measurements of all Units are done	**YES**
HLA and HPA tests when required of All Units are done.	**YES**
Inspection of Visual Changes and Leakage in the Extractor of all Units is made.	**-**
Factor VIIIc tests with pooled or separate units are done by qc.	**-**
All of these Samples have been stored, under suitable conditions.	**YES**
And with the same conditions, as the other same type Comp; before and during testing.	**YES**
All of these samples, have been exterminated with a suitable method.	**YES**

QC Laboratory Test Results

	Component Unit Nr.	Blood Coll. Date	Expiry Date	Thrombocy Count Day 1st Quality Requirement > 60 x 10^9 /Unit	Residual Leucocytes Day 1st Quality Requirement <0.2 x10^6 /Unit	pH Value Day 4th Quality Requirement 6.4 – 7.4			Comments on the QC of the Sample	QC Code	QC Spcl
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											

Possible Error Sources

Laboratory Blood Count Equipment :		
Transportation & Storage Conditions :		
Centrifuge Equipment :		
Extractor Usage :		
Component Bag :		
Laboratory Units to be Informed :		

Other Explanations :

	Quality Control Specialist	Laboratory Administrator	Blood Bank Manager
Name			
Signature			

Figure E.2 Monthly TS QC Report

E.3 Monthly FFP QC Report

MONTHLY QUALITY CONTROL REPORT

BOUN BloodQCM V2.1β
Reporting Module

BLOOD BANK : ISTANBUL

MONTH :

COMPONENT : FRESH FROZEN PLASMA

DATE :

QC Frequency	**% 1 of all units with a Minimum of 4 Units / Month**
Total Number of Components Produced in this Month :	
Nr. of Samples according to 1 % of Total Nr. of Comp. Produced in this Month :	
Instead of 1% of Total Nr. of Comp., The number of Samples is limited to	
The number of Samples of this Month is :	**4**
ABO and RhD Grouping of All Units are done.	**YES**
All of the Screening tests required are done.	**YES**
Volume measurements of at least 1% of all Units are done	**-**
Volume measurements of all Units are done	**YES**
HLA and HPA tests when required of All Units are done.	**-**
Inspection of Visual Changes and Leakage in the Extractor of all Units is made.	**YES**
Factor VIIIc tests with pooled or separate units are done by qc.	**YES**
All of these Samples have been stored, under suitable conditions.	**YES**
And with the same conditions, as the other same type Comp; before and during testing.	**YES**
All of these samples, have been exterminated with a suitable method.	**-**

QC Laboratory Test Results

	Component Unit Nr.	Blood Coll. Date	Expiry Date	-	-	Residual Erythrocyt Day 1st Quality Requirement < 6.0 x 10^9/L	Residual Leucocytes Day 1st Quality Requirement < 0.1 x 10^9/L	Residual Thromboc Day 1st Quality Requirement < 50 x 10^9/L	Comments on the QC of the Sample	QC Code	QC Spcl
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											

Possible Error Sources

Laboratory Blood Count Equipment :		
Transportation & Storage Conditions :		
Centrifuge Equipment :		
Extractor Usage :		
Component Bag :		
Laboratory Units to be Informed :		

Other Explanations :

	Quality Control Specialist	Laboratory Administrator	Blood Bank Manager
Name			
Signature			

Figure E.3 Monthly FFP QC Report

E.4 Annual QC Results Summation Report Form

ANNUAL QUALITY CONTROL SUMMATION REPORT

BOUN BloodQCM V2.1B
Reporting Module

BLOOD BANK : ISTANBUL

YEAR :

COMPONENT : THROMBOCYTE SUSP

DATE :

QC Frequency: **% 1 of all units with a Minimum of 10 Units / Month**

Total Number of Components Produced in this Year :	
The number of Samples of this Year :	120
The number of Unqualified Samples of this Year :	
ABO and RhD Grouping of All Units are done.	YES
All of the Screening tests required are done.	YES
Volume measurements of at least 1% of all Units are done	-
Volume measurements of all Units are done	YES
HLA and HPA tests when required of All Units are done.	YES
Inspection of Visual Changes and Leakage in the Extractor of all Units is made.	-
Factor VIIIc tests with pooled or separate units are done by qc.	-
All of these Samples have been stored, under suitable conditions.	YES
And with the same conditions, as the other same type Comp; before and during testing.	YES
All of these samples, have been exterminated with a suitable method.	YES

Quality Control Annual Summation and Percentage Table of Thrombocyte Suspension Component according to QC Laboratory Test Results of the Year

		MONTHS												Σ	% INCIDENCE
		JAN	FEB	MAR	APR	MAY	JUN	JUL	AGU	SEP	OCT	NOV	DEC		
Number of the Samples															
Nr. of Unqualifed Samples															
Percentage Of Unqualifed Samples															
Parameter : **Thrombocyte Count**	**Number** of Unqualified Results														
	Percentage of Unqualified Results, to the Number of Test Results of that Month														
Parameter : **Residual Leucocytes**	**Number** of Unqualified Results														
	Percentage of Unqualified Results, to the Number of Test Results of that Month														
Parameter : **Ph Degree**	**Number** of Unqualified Results														
	Percentage of Unqualified Results, to the Number of Test Results of that Month														
QC Status Code		QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	

Explanations Required:

Quality Control Specialist | **Laboratory Administrator** | **Blood Bank Manager**

Name

Signature

Figure E.4 Annual QC Results Summation Report Form

E.4.1 Annual QC Results Summation Report, TS Sample

ANNUAL QUALITY CONTROL SUMMATION REPORT

BOUN BloodQCM V2.1B
Reporting Module

BLOOD BANK : ISTANBUL

YEAR :

COMPONENT : THROMBOCYTE SUSP

DATE :

QC Frequency: **% 1 of all units with a Minimum of 10 Units / Month**

Total Number of Components Produced in this Year :	1240
The number of Samples of this Year :	155
The number of Unqualified Samples of this Year :	15
ABO and RhD Grouping of All Units are done.	YES
All of the Screening tests required are done.	YES
Volume measurements of at least 1% of all Units are done	-
Volume measurements of all Units are done	YES
HLA and HPA tests when required of All Units are done.	YES
Inspection of Visual Changes and Leakage in the Extractor of all Units is made.	-
Factor VIIIc tests with pooled or separate units are done by qc.	-
All of these Samples have been stored, under suitable conditions.	YES
And with the same conditions, as the other same type Comp; before and during testing.	YES
All of these samples, have been exterminated with a suitable method.	YES

Quality Control Annual Summation and Percentage Table of Thrombocyte Suspension Component according to QC Laboratory Test Results of the Year

		MONTHS												Σ	% INCIDENCE
		JAN	FEB	MAR	APR	MAY	JUN	JUL	AGU	SEP	OCT	NOV	DEC		
Number of the Samples		10	10	10	20	20	20	10	10	15	10	10	10	155	-
Nr. of Unqualifed Samples		1	0	0	1	4	3	3	0	0	1	1	1	15	
Percentage Of Unqualifed Samples		%10	%0	%0	%5	%20	%15	%30	%0	%0	%10	%10	%10	-	9.68
Parameter : Thrombocyte Count	Number of Unqualified Results	1	0	0	1	0	0	3	0	0	0	0	0	2	
	Percentage of Unqualified Results, to the Number of Test Results of that Month	%10	%0	%0	%5	%0	%0	%30	%0	%0	%0	%0	%0		3.23
Parameter : Residual Leucocytes	Number of Unqualified Results	1	0	0	1	4	3	0	0	0	1	1	2	13	
	Percentage of Unqualified Results, to the Number of Test Results of that Month	%10	%0	%0	%5	%20	%15	%0	%0	%0	%10	%10	%20		8.39
Parameter : Ph Degree	Number of Unqualified Results	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Percentage of Unqualified Results, to the Number of Test Results of that Month	%0	%0	%0	%0	%0	%0	%0	%0	%0	%0	%0	%0		0.00
QC Status Code		QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC< 75%	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	QC OK.	

Explanations Required:

Quality Control Specialist	Laboratory Administrator	Blood Bank Manager
Name		
Signature		

Figure E.5 Annual QC Results Summation Report, TS Sample

APPENDIX F. COMPONENT REQUEST FORM SAMPLE

QUALITY CONTROL
SAMPLE COMPONENT REQUEST FORM

BOUN BloodQCM V2.1β
Reporting Module

BLOOD BANK : ISTANBUL

COMPONENT :

MONTH :

DATE :

For the Monthly Quality Control Processes, that will take place in QC Lab;
The Blood Components written below requested, from the Storage Dept of Blood Bank.
The Components are transported to related Unit of Blood Bank, on dates indicated.

Unit Numbers and Transportation Data of the Components are as follows :

Nr.	Comp. Type	Comp. Unit Nr.	Blood Coll.Date	Exp. Date	1st Transportation From Storage to QC Lab			2nd Transportation From QC Lab to Storage			3rd Transportation From Storage to QC Lab			4th Transportation From QC Lab to Storage			Exterminate	Other
					Date	Storage Specit	QC Specialist	Date	Storage Specit	QC Specialist	Date	Storage Specit	QC Specialist	Date	Storage Specit	QC Specialist		
1																		
2																		
3																		
4																		
5																		
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
17																		
18																		
19																		
20																		

Explanations If Required :

	Quality Control Specialist	Storage Department Administrator	Blood Bank Manager
Name			
Signature			

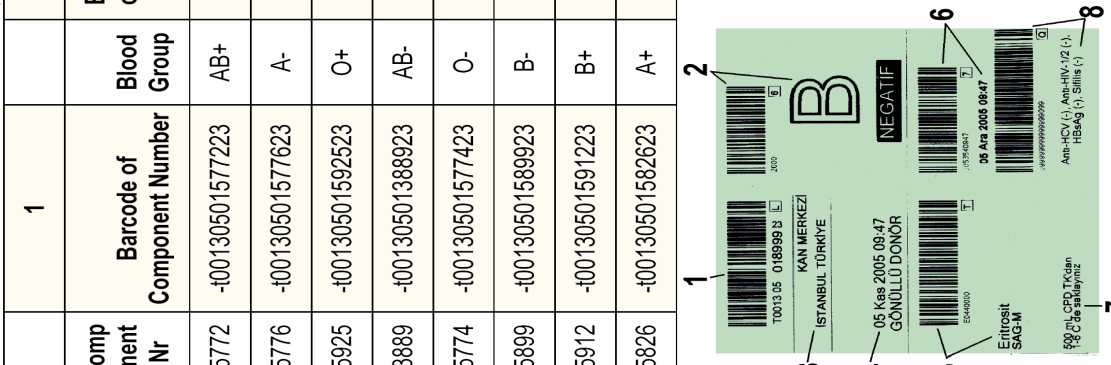
Figure F.1 Component Request Form Sample

APPENDIX G. SAMPLE BARCODE READINGS

Table G.1

Sample barcode readings from Blood Labels of Türk Kızılayı. (Data acquisition by a barcode reader from eight samples of blood component labels. Localizations of the barcodes on the label, which are shown in numbers in the list, defined on the label sample below.)

	1		2		5			6	8
Component Nr	Barcode of Component Number	Blood Group	Barcode of Blood Group	Component Type	Barcode of Component Type	Blood Collection Date, and Time	Expiry Date, and Time	Barcode of Expiry Date	Barcode of Test Results: Passed
15772	-t00130501577223	AB+	-%8400	FFP	-ëe0701000	07112005, 22:26	07112006, 22:26	/ç0063112226	-,099999999999999099
15776	-t00130501577623	A-	-%0600	TS	-ëe2824000	07112005, 23:59	12112005, 23:59	/ç0053162359	-,099999999999999099
15925	-t00130501592523	O+	-%5100	ES	-ëe0440000	04112005, 12:51	16122005, 12:51	/ç0053501251	-,099999999999999099
13889	-t00130501388923	AB-	-%2800	FFP	-ëe0701000	30102005, 16:00	30102006, 16:00	/ç0063031600	-,099999999999999099
15774	-t00130501577423	O-	-%9500	TS	-ëe2824000	07112005, 23:13	12112005, 23:13	/ç0053162313	-,099999999999999099
15899	-t00130501589923	B-	-%2000	ES	-ëe0440000	08112005, 09:47	20122005, 09:47	/ç0053540947	-,099999999999999099
15912	-t00130501591223	B+	-%7300	ES	-ëe0440000	08112005, 11:33	20122005, 11:33	/ç0053541133	-,099999999999999099
15826	-t00130501582623	A+	-%6200	ES	-ëe0440000	08112005, 08:30	20122005, 08:30	/ç0053540830	-,099999999999999099



APPENDIX H. SOFTWARE SOURCE CODE

H.1 Source Code of Domain

FormContainer

```
package com.ptah.kankalite.domain;

import java.io.Serializable;
import java.util.HashMap;
import java.util.Map;
import java.util.TreeMap;

import com.ptah.kankalite.constants.FormState;
import com.ptah.kankalite.constants.FormType;

public class FormContainer implements Serializable {

    // TODO : Bilimsel min, bilimsel max değerlerini kontrol et
    /*
     * Her formdaki sample sayısını kullanıcı tarafından forse etmek
için 0'dan farklı bir değer verilir.
     */
    private int userSampleCount = 0;
    private static final long serialVersionUID = 5718965482615025476L;

    // String - 022007, 112007, etc
    // Integer - 3,5,6, etc
    // Verilen periyotta her formdan kac tane alinmasi gerektigini tutar
    private Map<FormType, Map<String, Integer>> periodicSampleCount =
new TreeMap<FormType, Map<String, Integer>>();

    /*
         * FormContainer'daki formların listesi formList objesinde
tutulur (Belirli bir formType'a ait Map'ler şeklinde)
         * Bu Map'lerin key'i dönem bilgisini gösteren String, data'sı
ise formun kendisidir.
         */
    private Map<FormType, Map<String, Form>> formList = new
TreeMap<FormType, Map<String, Form>>();

    /*
         * Belirli bir form tipi için verilen döneme ait ürün sayısı set
edilir. Bu method çağrıldığında ilgili döneme
         * ait yeni bir form objesi yaratır ve formList'e ekler.
         */
    public void setGeneratedProductCount(FormType formType, String date,
int count) {
      Map<String, Integer> periodicSpecificFormValues;
      periodicSpecificFormValues = periodicSampleCount.get(formType);
      if (periodicSpecificFormValues == null) {
          periodicSpecificFormValues = new HashMap<String, Integer>();
      }
      periodicSpecificFormValues.put(date, count);
      periodicSampleCount.put(formType, periodicSpecificFormValues);
      Form form = generateForm(formType, date);
      Map<String, Form> formsWithSameType = formList.get(formType);
      /*
         * Eğer böyle bir Map henüz yaratılmadıysa, şimdi yaratılmalı
         */
      if (formsWithSameType == null) {
          formsWithSameType = new HashMap<String, Form>();
```

```
      }
      formsWithSameType.put(date, form);
      formList.put(formType, formsWithSameType);
    }

    /*
         * Belirli bir sure icersindeki (022007, etc) üretilmiş olan
ürün sayısı
         */
    public int getGeneratedProductCount(FormType formType, String date)
{
      int result = 0;
      Map<String, Integer> list = periodicSampleCount.get(formType);
      if (list != null) {
          try {
            result = list.get(date);
          } catch (NullPointerException e) {
            result = -1;
          }
      } else {
          result = -1;
      }
      return result;
    }

    public int getNeededSampleCount(FormType formType, String date) {
      if (userSampleCount > 0) {
          return userSampleCount;
      }
      int numberOfSamplesToBeTested = -1;
      int count = getGeneratedProductCount(formType, date);
      if (formType == FormType.ES) {
          if (count < 401) {
            numberOfSamplesToBeTested = 4;
          } else {
            numberOfSamplesToBeTested = Math.round((float) (count + 49) /
100);
          }
      } else if (formType == FormType.TS) {
          //numberOfSamplesToBeTested = Math.round((float) count *
0.01f);
          numberOfSamplesToBeTested = Math.round((float) (count + 49) /
100);
          if (numberOfSamplesToBeTested < 10) {
            numberOfSamplesToBeTested = 10;
          }
      } else if (formType == FormType.TDP) {
          if (count < 401) {
            numberOfSamplesToBeTested = 4;
          } else {
            numberOfSamplesToBeTested = Math.round((float) (count + 49) /
100);
          }
      }
      return numberOfSamplesToBeTested;
    }

    /*
         * Verilmiş olan ay için form üreteci (O periyot için üretilmiş
ürünlerin sayısına bakarak kaç sample alınması
         * gerektiğini de bildirir.
         */
    public Form generateForm(FormType formType, String period) {
      int numberOfSamplesToBeTested = getNeededSampleCount(formType,
period);

      Map<String, Form> formForSpecificPeriod = new HashMap<String,
Form>();
```

```
        Form form = FormFactory.createForm(formType, period);
        form.setNumberOfSamplesToBeTested(numberOfSamplesToBeTested);
        formForSpecificPeriod.put(period, form);

        form.createSamples();
        return form;
      }

      public Map<String, Form> getForms(FormType formType) {
        return formList.get(formType);
      }

      /*
           * Spesifik bir formu getirir - Basitlik olsun diye konuldu
           */
      public Form getForm(FormType formType, String date) {
        Form forms = formList.get(formType).get(date);
        return forms;
      }

      public void setUserSampleCount(int userSampleCount) {
        this.userSampleCount = userSampleCount;
      }

      public Map<String, Form> getFormsToBeCompletedForYear(FormType type)
{
        Map<String, Form> forms = new TreeMap<String, Form>();
        Map<String, Form> currentForms = formList.get(type);
        for (String form : currentForms.keySet()) {
            FormState state = currentForms.get(form).getState();
            if ((state == FormState.SAMPLE_ABSENT) || (state ==
FormState.TESTS_WAITING)) {
              forms.put(form, currentForms.get(form));
            }
        }
        return forms;
      }

      public void createFormsForYear(FormType type, String year) {
        Map<String, Form> currentForms = formList.get(type);
        if (currentForms == null) {
            currentForms = new TreeMap<String, Form>();
        }
        for (int i = 1; i <= 12; i++) {
            String index = "" + i + year;
            // Başında 0 olmayan String'lerin başına 0 koyuyoruz
            if (index.length() == 5) {
              index = "0" + index;
            }
            if (currentForms.get(index) == null) {
              Form form = generateForm(type, index);
              currentForms.put(index, form);
            }
        }
        formList.put(type, currentForms);
      }

      public Map<FormType, Map<String, Form>> getFormList() {
        return formList;
      }

      public void setFormList(Map<FormType, Map<String, Form>> formList) {
        this.formList = formList;
      }

      public Sample getSpecificSample(FormType formType, int bloodNo) {
        Sample result=null;
        Map<String, Form> forms=getForms(formType);
```

```
      for (String period:forms.keySet()) {
          Form tempForm=getForm(formType, period);
          Map<Integer, Sample> samples = tempForm.getSamples();
          for (Sample sample:samples.values()) {
            if (sample.getBloodNo()==bloodNo) {
                result=sample;
            }
          }
      }
      return null;
    }
}
```

FormFactory

```
package com.ptah.kankalite.domain;

import com.ptah.kankalite.constants.FormType;

public class FormFactory {
//     Factory method olarak kullanılıyor - Sadece FormContainer
tarafindan kullanildigindan emin olmak lazim
      protected static Form createForm(FormType formType, String period)
{
            Form form=null;
            if (formType == FormType.ES) {
                  form = new ESForm(period);
            }
            if (formType == FormType.TS) {
                  form = new TSForm(period);
            }
            if (formType == FormType.TDP) {
                  form = new TDPForm(period);
            }
            return form;
      }
}
```

Form

```
package com.ptah.kankalite.domain;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Group;

import com.ptah.kankalite.constants.FormState;
import com.ptah.kankalite.constants.FormType;
import com.ptah.kankalite.constants.ParameterType;

public abstract class Form implements IForm, Serializable {
    protected long id;
    protected FormType type;
```

```
    protected Date date;
    protected String name;
    protected int expiryPeriod;
    protected int numberOfSamplesToBeTested;
    protected FormState state;
    protected int totalFailedSampleCountForThisForm;
    protected boolean valid;
    protected String period;
    protected String condition;
    protected int leastSampleCount;

    private boolean sampleCountCheck;
    private boolean storageBeforeQCCheck;
    private boolean storageDuringQCCheck;
    private boolean storageExterminationCheck;
    private String qcSpecialist;
    private boolean screeningCheck;
    private boolean groupingCheck;

    private String esComponentBag;
    private String esUnitsToBeInformed;
    private String esExtractorUsage;
    private String esCentrifuge;
    private String esStorage;
    private String esBloodCountEquipment;
    //transient Button sampleUpdateButton;
    private int noOfProducedComponents;

    protected Map<ParameterType, Integer> results = new
TreeMap<ParameterType, Integer>();

    private Map<Integer, Sample> samples = new TreeMap<Integer,
Sample>();

    List<ParameterType> parameterTypes = new ArrayList<ParameterType>();

    public Form() {
      state = FormState.SAMPLE_ABSENT;
      totalFailedSampleCountForThisForm = 0;
    }

    public abstract void checkState();

    public abstract void checkResults();

    public abstract void showParameterEntries(Group parameterEntry);

    public abstract void extractParametersFromGUI(Sample currentSample);

    public abstract void updateGUIFromParameters(Sample currentSample);

    public int getResult(ParameterType type) {
      return results.get(type);
    }

    /*
         * checkState() burada çağırılıyor - Bütün validity
checking'leri checkState içersinde yapılıyor.
         */
    public FormState getState() {
      checkState();
      return state;
    }

    public void setState(FormState state) {
      this.state = state;
    }
```

```
    public FormType getType() {
      return type;
    }

    public void setType(FormType type) {
      this.type = type;
    }

    public String getName() {
      return name;
    }

    public long getId() {
      return id;
    }

    public void setNumberOfSamplesToBeTested(int
numberOfSamplesToBeTested) {
      this.numberOfSamplesToBeTested = numberOfSamplesToBeTested;
      // createSamples();
    }

    public int getNumberOfSamplesToBeTested() {
      return numberOfSamplesToBeTested;
    }

    public Map<Integer, Sample> getSamples() {
      return samples;
    }

    public void updateAnInitialSample() {

    }

    /*
         * Generic sample parametrelerini set etmek icin en uygun yer
burasi
         */
    public void createSamples() {
      createSamples(0);
    }

    public void createSamples(int startPoint) {
      for (int i = startPoint; i < numberOfSamplesToBeTested; i++) {
          Sample sample = new Sample(i);
          Map<ParameterType, Parameter> sampleParam = new
HashMap<ParameterType, Parameter>();
          for (ParameterType type : parameterTypes) {
            sampleParam.put(type, new Parameter(0, -1.0f));
          }
          sample.setParameters(sampleParam);
          // sample.setProductionDate(new
Date(System.currentTimeMillis()));
          samples.put(sample.getId(), sample);
          sample.setExpiryPeriod(expiryPeriod);
      }
    }

    /*
         * i. örneğin production date'ini set eder
         */
    public void setSampleProductionDate(int i, Date date) {
      Sample aSample = samples.get(i);
      aSample.setProductionDate(date);
      Calendar calendar = Calendar.getInstance();
      calendar.setTime(date);
      calendar.add(Calendar.DAY_OF_MONTH, this.expiryPeriod);
      aSample.setExpiryDate(calendar.getTime());
```

```
    }

    public Date getSampleExpiryDate(int i) {
      return samples.get(i).getExpiryDate();
    }

    public void addParameterType(ParameterType type) {
      parameterTypes.add(type);
    }

    public Date getDate() {
      return date;
    }

    public void setDate(Date date) {
      this.date = date;
    }

    /*
         * Verilen parametrenin min ve max değerlerinin kalite
değerlerinin arasında olması durumu kontrol eder.
         * Değerlendirmeye girmesi istenmeyen değerler NaN olarak
verilir. minSuccessRate ise örneklerin % kaçının
         * geçerli olmasının yeterli olacağını belirtir. Sonucu
"results" isimli hashMap'de saklar.
         */
    protected void checkSpecificParameter(ParameterType type, int min,
int max, float minQValue, float maxQValue,
          float minSuccessRate) {
      results.put(type, 0);
      int actualSampleCount = getSamples().keySet().size();
      int failedSampleCountForSpecificType = 0;
      for (Sample sample : getSamples().values()) {
          Parameter parameter = sample.getParameter(type);
          parameter.setValid(true);
          // sample.setValid(true);
          boolean sampleFailed = false;

          // İlk alınan örnekten son alınan örneğe kadar iterate edelim
          for (int i = min; i <= max; i++) {
            boolean smaller = (minQValue != Float.NaN &&
parameter.getValue(i) < minQValue);
            boolean greater = (maxQValue != Float.NaN &&
parameter.getValue(i) > maxQValue);
            if (smaller || greater) {
                sampleFailed = true;
            }
          }

          // Bu parametre fail etti ise :
          if (sampleFailed) {
            parameter.setValid(false);
            sample.setValid(false);
            increaseFailCountForParameter(type, parameter);
            failedSampleCountForSpecificType++;
            totalFailedSampleCountForThisForm++;
          }
      }
      float actualFailureRate = 0;
      try {
          actualFailureRate = 100 * failedSampleCountForSpecificType /
actualSampleCount;
      } catch (Exception e) {
      }
      float maxFailureRate = 100 - minSuccessRate;
      if (actualFailureRate > maxFailureRate) {
          this.valid = false;
          // totalFailedSampleCountForThisForm++;
```

```
      } else {
          this.valid = true;
      }
    }

    private void increaseFailCountForParameter(ParameterType type,
Parameter parameter) {
      int val = results.get(type);
      results.put(type, val + 1);
    }

    public int getFailedSampleCount() {
      return totalFailedSampleCountForThisForm;
    }

    public void setFailedSampleCount(int failedSampleCount) {
      this.totalFailedSampleCountForThisForm = failedSampleCount;
    }

    public int getTotalFailedSampleCountForThisForm() {
      return totalFailedSampleCountForThisForm;
    }

    public boolean isValid() {
      return valid;
    }

    public void setPeriod(String period) {
      this.period = period;
    }

    public String getPeriod() {
      return period;
    }

    public String getCondition() {
      return condition;
    }

    public int getLeastSampleCount() {
      return leastSampleCount;
    }

    public String getQcSpecialist() {
      return qcSpecialist;
    }

    public void setQcSpecialist(String qcSpecialist) {
      this.qcSpecialist = qcSpecialist;
    }

    public boolean isSampleCountCheck() {
      return sampleCountCheck;
    }

    public void setSampleCountCheck(boolean sampleCountCheck) {
      this.sampleCountCheck = sampleCountCheck;
    }

    public boolean isStorageBeforeQCCheck() {
      return storageBeforeQCCheck;
    }

    public void setStorageBeforeQCCheck(boolean storageBeforeQCCheck) {
      this.storageBeforeQCCheck = storageBeforeQCCheck;
    }

    public boolean isStorageDuringQCCheck() {
```

```
    return storageDuringQCCheck;
  }

  public void setStorageDuringQCCheck(boolean storageDuringQCCheck) {
    this.storageDuringQCCheck = storageDuringQCCheck;
  }

  public boolean isStorageExterminationCheck() {
    return storageExterminationCheck;
  }

  public void setStorageExterminationCheck(boolean
storageExterminationCheck) {
    this.storageExterminationCheck = storageExterminationCheck;
  }

  public boolean isGroupingCheck() {
    return groupingCheck;
  }

  public void setGroupingCheck(boolean groupingCheck) {
    this.groupingCheck = groupingCheck;
  }

  public boolean isScreeningCheck() {
    return screeningCheck;
  }

  public void setScreeningCheck(boolean screeningCheck) {
    this.screeningCheck = screeningCheck;
  }

  public String getEsBloodCountEquipment() {
    return esBloodCountEquipment;
  }

  public void setEsBloodCountEquipment(String esBloodCountEquipment) {
    this.esBloodCountEquipment = esBloodCountEquipment;
  }

  public String getEsCentrifuge() {
    return esCentrifuge;
  }

  public void setEsCentrifuge(String esCentrifuge) {
    this.esCentrifuge = esCentrifuge;
  }

  public String getEsComponentBag() {
    return esComponentBag;
  }

  public void setEsComponentBag(String esComponentBag) {
    this.esComponentBag = esComponentBag;
  }

  public String getEsExtractorUsage() {
    return esExtractorUsage;
  }

  public void setEsExtractorUsage(String esExtractorUsage) {
    this.esExtractorUsage = esExtractorUsage;
  }

  public String getEsStorage() {
    return esStorage;
  }
```

```
    public void setEsStorage(String esStorage) {
      this.esStorage = esStorage;
    }

    public String getEsUnitsToBeInformed() {
      return esUnitsToBeInformed;
    }

    public void setEsUnitsToBeInformed(String esUnitsToBeInformed) {
      this.esUnitsToBeInformed = esUnitsToBeInformed;
    }

    public int getNoOfProducedComponents() {
      return noOfProducedComponents;
    }

    public void setNoOfProducedComponents(int noOfProducedComponents) {
      this.noOfProducedComponents = noOfProducedComponents;
    }

    public Button getSampleUpdateButton() {
      return null;//this.sampleUpdateButton;
    }

    /*public void __createUpdateButton(Group group) {
      if (this.sampleUpdateButton!=null) {
          this.sampleUpdateButton.dispose();
          System.out.println("Disposed");
      }
      this.sampleUpdateButton = new Button(group, SWT.NONE);
    }*/

}
```

Iform

```
package com.ptah.kankalite.domain;

import java.util.List;

public interface IForm {
      public String getName();
      public void checkState();
      public String getCondition();
      public int getLeastSampleCount();
      public List<String> getSampleParameterNames();
      public List<Float> getSampleParameterValues(int key);
}
```

Parameter

```
package com.ptah.kankalite.domain;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import com.ptah.kankalite.constants.ParameterType;
```

```
public class Parameter implements Serializable {
	private static final long serialVersionUID = -6311775687328445799L;
	private Map<Integer, Float> values;
	private boolean valid;

	public boolean isValid() {
		return valid;
	}

	public void setValid(boolean valid) {
		this.valid = valid;
	}

	public Parameter(int index, float value){
		this.values=new HashMap<Integer, Float>();
		this.values.put(index,value);
		this.valid=true;
	}

	public List<Float> getValues(ParameterType type) {
		Set<Integer> keys=values.keySet();
		List<Float> result=new ArrayList<Float>();
		for (Integer key:keys) {
			result.add(values.get(key));
		}
		return result;
	}

	public void addValue(int index, float value) {
		values.put(index, value);
	}

	public float getValue(int i) {
		float result=-1;
		try {
			result=values.get(i);
		} catch (Exception e) {
			result=-1;
		}
		return result;
	}

	public void setValue(int i, float value) {
		this.values.put(i, value);
	}
}
```

Sample

```
package com.ptah.kankalite.domain;

import java.io.Serializable;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import com.ptah.kankalite.constants.ParameterType;

public class Sample implements Serializable {
    private static final long serialVersionUID = 3779132908709268582L;
    /*
         * Id parametresi, her formdaki indeks numarasidir
```

```
         */
    private int id;
    private int bloodNo;
    private Date productionDate;
    private Date expiryDate;
    private int expiryPeriod;
    private boolean valid;
    private Map<ParameterType, Parameter> parameters = new
HashMap<ParameterType, Parameter>();

    /*
     * Default olarak productionDate set edilsin, expiryDate hesaplansin
     */
    public Sample(int id) {
      this.id = id;
      this.valid = true;
    }

    public Parameter getParameter(ParameterType type) {
      return parameters.get(type);
    }

    public void setParameter(ParameterType type, Parameter parameter) {
      parameters.put(type, parameter);
    }

    public Map<ParameterType, Parameter> getParameters() {
      return parameters;
    }

    public void setParameters(Map<ParameterType, Parameter> parameters)
{
      this.parameters = parameters;
    }

    public Set<ParameterType> getParameterTypes() {
      return parameters.keySet();
    }

    public int getId() {
      return id;
    }

    public void setId(int id) {
      this.id = id;
    }

    public Date getExpiryDate() {
      return expiryDate;
    }

    public void setExpiryDate(Date expiryDate) {
      this.expiryDate = expiryDate;
      Calendar calendar = Calendar.getInstance();
      calendar.setTime(expiryDate);
      calendar.add(Calendar.DAY_OF_MONTH, (-1) * this.expiryPeriod);
      this.productionDate = calendar.getTime();
    }

    public Date getProductionDate() {
      return productionDate;
    }

    public void setProductionDate(Date productionDate) {
      this.productionDate = productionDate;
      if (productionDate != null) {
          Calendar calendar = Calendar.getInstance();
          calendar.setTime(productionDate);
```

```
            calendar.add(Calendar.DAY_OF_MONTH, this.expiryPeriod);
            this.expiryDate = calendar.getTime();
        }
    }

    public int getExpiryPeriod() {
        return expiryPeriod;
    }

    public void setExpiryPeriod(int expiryPeriod) {
        this.expiryPeriod = expiryPeriod;
    }

    public int getBloodNo() {
        return bloodNo;
    }

    public void setBloodNo(int bloodNo) {
        this.bloodNo = bloodNo;
    }

    /*
         * type : Parametre tipi index : Parametrenin kaçıncı değeri (0,
1 - initial, expiring) value : Parametrenin
         * değeri
         */
    public void add(ParameterType type, int index, float value) {
        Parameter parameter = null;
        try {
            parameter = parameters.get(type);
        } catch (NullPointerException e) {
            System.out.println("!!! - No parameter found with type : " +
type);
        }
        parameter.setValue(index, value);
    }

    public boolean isValid() {
        return valid;
    }

    public void setValid(boolean valid) {
        this.valid = valid;
    }

}
```

Storage

```
package com.ptah.kankalite.domain;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class Storage {
    private static String fileName = "ptah.store";
    private static String directory = "/tmp";

    private void initialize() {
        File file;
        file = new File(directory);
        file.mkdir();
```

```
    }

    public Storage() {
      initialize();
    }

    public void store(Object object) {
      try {
          String target = directory + "/" + fileName;
          FileOutputStream fo = new FileOutputStream(target);
          ObjectOutputStream so = new ObjectOutputStream(fo);
          so.writeObject(object);
          so.flush();
      } catch (Exception e) {
          System.err.println("Exception while storing snapshot");
          e.printStackTrace();
      }
    }

    public Object load() throws Exception {
      Object object = null;
      String target = directory + "/" + fileName;
      FileInputStream fo = new FileInputStream(target);
      ObjectInputStream so = new ObjectInputStream(fo);
      object = (Object) so.readObject();
      return object;
    }

}
```

QCUtil

```
package com.ptah.kankalite.domain;

import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableItem;

public class QCUtil {

    public static void updateSamplesTable(Table table, Form form) {
      TableItem item = new TableItem(table, 0);
      String[] tableValues;
      tableValues = new String[form.getSampleParameterNames().size() +
2];
      // Once header'i olusturalim
      int i = 0;
      tableValues[i++] = "Unit #";
      for (String name : form.getSampleParameterNames()) {
          tableValues[i++] = name;
      }
      item.setText(tableValues);


      // Sonra data'lari set edelim
      for (int sampleNo = 0; sampleNo < form.getSamples().size();
sampleNo++) {
          item = new TableItem(table, 0);
          tableValues = new String[form.getSampleParameterNames().size()
+ 1];
          i = 0;
          tableValues[i++] = "" + sampleNo;
          for (Float value : form.getSampleParameterValues(sampleNo)) {
            tableValues[i++] = "" + value;
          }
          item.setText(tableValues);
```

```
      }
    }

}
```

Qtree

```
package com.ptah.kankalite.domain;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Tree;
import org.eclipse.swt.widgets.TreeItem;

public class QTree {

    private Tree tree;
    List<TreeItem> mainItems;
    Form selectedForm;

    public QTree(Composite parent, int style) {
      tree = new Tree(parent, style);
      tree.setBounds(5,40,205,250);
      tree.addSelectionListener(new SelectionAdapter() {
          public void widgetSelected(SelectionEvent e) {
            TreeItem tempItem=tree.getSelection()[0];
            selectedForm = (Form)tempItem.getData();
          }
      });
      mainItems = new ArrayList<TreeItem>();
    }

    public void setSelection(TreeItem item) {
      tree.setSelection(item);
    }

    public Tree getTree() {
      return tree;
    }

    public void setTree(Tree tree) {
      this.tree = tree;
    }

    public void addItem(String itemStr) {
      addItem(itemStr, null);
      tree.redraw();
    }

    public void addItem(String itemStr, Form form) {
      TreeItem item = null;
      if (form == null) {
          item = new TreeItem(tree, SWT.BORDER);
      } else {
          TreeItem tempItem = tree.getItem(form.getType().ordinal());
          item = new TreeItem(tempItem, SWT.BORDER);
      }
      item.setText(itemStr);
      item.setData(form);
    }
```

```
    public Form getSelectedForm() {
        return selectedForm;
    }

    public void setSelectedForm(Form form) {
      this.selectedForm=form;
    }

}
```

ESForm

```
package com.ptah.kankalite.domain;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Group;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Text;

import com.ptah.kankalite.constants.FormNames;
import com.ptah.kankalite.constants.FormState;
import com.ptah.kankalite.constants.FormType;
import com.ptah.kankalite.constants.ParameterType;

public class ESForm extends Form implements Serializable {

    private static final long serialVersionUID = 3719530664889402371L;
    private float minHbQValue;
    private float minHtcQValue;
    private float maxHtcQValue;
    private float maxHmlzQValue;

    transient Text hmlz42ndDay;
    transient Text hb42ndDay;
    transient Text htc42ndDay;
    transient Text hb1stDay;
    transient Text htc1stDay;

    public ESForm(String period) {
      super();
      minHbQValue = 45f;
      minHtcQValue = 50f;
      maxHtcQValue = 70f;
      maxHmlzQValue = 2.8f;
      expiryPeriod = 42;
      leastSampleCount = 4;
      condition = "%1 of number of products; with a minimum of 4 units";
      name = FormNames.ES;
      setPeriod(period);
      setType(FormType.ES);
      setState(FormState.SAMPLE_ABSENT);
      id = System.currentTimeMillis();

      addParameterType(ParameterType.Htc);
      addParameterType(ParameterType.Hb);
      addParameterType(ParameterType.Hmlz);
    }

    public List<String> getSampleParameterNames() {
```

```
  List<String> result = new ArrayList<String>();
  result.add("Htc 1st Day");
  result.add("Hb 1st Day");
  result.add("Htc 42nd Day");
  result.add("Hb 42nd Day");
  result.add("Hmlz 42nd Day");
  return result;
}

public void showParameterEntries(Group parameterEntry) {
  //Composite parent=parameterEntry.getParent();
  //parameterEntry=new Group(parent, SWT.NONE);
  parameterEntry.setBounds(170, 175, 390, 75);

  Label htc1stDayLabel = new Label(parameterEntry, SWT.NONE);
  htc1stDayLabel.setAlignment(SWT.RIGHT);
  htc1stDayLabel.setText("Htc 1st day (%) :");
  htc1stDayLabel.setBounds(23, 10, 85, 13);

  Label hb1stDayLabel = new Label(parameterEntry, SWT.NONE);
  hb1stDayLabel.setAlignment(SWT.RIGHT);
  // toolkit.adapt(hb1stDayLabel, true, true);
  hb1stDayLabel.setText("Hb 1st day (g/unit) :");
  hb1stDayLabel.setBounds(3, 30, 105, 13);

  Label htc42ndDayLabel = new Label(parameterEntry, SWT.NONE);
  htc42ndDayLabel.setAlignment(SWT.RIGHT);
  // toolkit.adapt(htc42ndDayLabel, true, true);
  htc42ndDayLabel.setText("Htc 42nd day (%) :");
  htc42ndDayLabel.setBounds(170, 10, 105, 13);

  htc1stDay = new Text(parameterEntry, SWT.BORDER);
  // toolkit.adapt(htc1stDay, true, true);
  htc1stDay.setBounds(115, 10, 50, 15);

  hb1stDay = new Text(parameterEntry, SWT.BORDER);
  hb1stDay.setBounds(115, 30, 50, 15);
  // toolkit.adapt(hb1stDay, true, true);

  htc42ndDay = new Text(parameterEntry, SWT.BORDER);
  htc42ndDay.setBounds(280, 10, 50, 15);
  // toolkit.adapt(htc42ndDay, true, true);

  Label htc42ndDayLabel_1 = new Label(parameterEntry, SWT.NONE);
  htc42ndDayLabel_1.setAlignment(SWT.RIGHT);
  htc42ndDayLabel_1.setBounds(170, 30, 105, 13);
  // toolkit.adapt(htc42ndDayLabel_1, true, true);
  htc42ndDayLabel_1.setText("Hb 42nd day (%) :");

  Label htc42ndDayLabel_2 = new Label(parameterEntry, SWT.NONE);
  htc42ndDayLabel_2.setAlignment(SWT.RIGHT);
  htc42ndDayLabel_2.setBounds(170, 50, 105, 13);
  // toolkit.adapt(htc42ndDayLabel_2, true, true);
  htc42ndDayLabel_2.setText("Hmlz 42nd day (%) :");

  hb42ndDay = new Text(parameterEntry, SWT.BORDER);
  hb42ndDay.setBounds(280, 30, 50, 15);

  hmlz42ndDay = new Text(parameterEntry, SWT.BORDER);
  hmlz42ndDay.setEnabled(false);
  hmlz42ndDay.setEditable(false);
  hmlz42ndDay.setBounds(280, 50, 50, 15);

  /*createUpdateButton(parameterEntry);
  sampleUpdateButton.setText("Update");
  sampleUpdateButton.setBounds(335, 10, 50, 60);*/
}
```

```
    @Override
    public void updateGUIFromParameters(Sample currentSample) {
          try {
            hb1stDay.setText("");
            Parameter hb = currentSample.getParameter(ParameterType.Hb);
            hb1stDay.setText("" + hb.getValue(0));
            hb42ndDay.setText("" + hb.getValue(1));
            Parameter htc =
currentSample.getParameter(ParameterType.Htc);
            htc1stDay.setText("" + htc.getValue(0));
            htc42ndDay.setText("" + htc.getValue(1));
            // Hemoliz parametresini hesaplatmak icin checkState()
methodu cagiriliyor.
            checkState();
            Parameter hmlz =
currentSample.getParameter(ParameterType.Hmlz);
            hmlz42ndDay.setText("" + hmlz.getValue(1));
          } catch (Exception e) {
          }
    }

    @Override
    public void extractParametersFromGUI(Sample currentSample) {
      try {
            float hb1st = Float.parseFloat(hb1stDay.getText());
            float hb42nd = Float.parseFloat(hb42ndDay.getText());
            float htc1st = Float.parseFloat(htc1stDay.getText());
            float htc42nd = Float.parseFloat(htc42ndDay.getText());

            Parameter hb = new Parameter(0, hb1st);
            hb.setValue(1, hb42nd);
            currentSample.setParameter(ParameterType.Hb, hb);

            Parameter htc = new Parameter(0, htc1st);
            htc.setValue(1, htc42nd);
            currentSample.setParameter(ParameterType.Htc, htc);
          } catch (Exception ex) {
            System.out.println("Some parameter values are invalid");
          }
    }

    public List<Float> getSampleParameterValues(int key) {
      List<Float> result = new ArrayList<Float>();
      result.add(getSamples().get(key).getParameters().get(ParameterType.
Htc).getValue(0));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
Hb).getValue(0));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
Htc).getValue(1));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
Hb).getValue(1));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
Hmlz).getValue(1));
      return result;
    }

    @Override
    public void checkState() {
      FormState state = FormState.SAMPLE_ABSENT;
      boolean expiringSamplesAreValid = true;
      boolean initialSamplesAreValid = true;
      Collection<Sample> samples = getSamples().values();

      for (Sample sample : samples) {
          boolean isInitialParameterValid = false;
          boolean isExpiryParameterValid = false;
          Parameter htc = sample.getParameter(ParameterType.Htc);
          Parameter hb = sample.getParameter(ParameterType.Hb);
```

```
            Parameter hmlz = sample.getParameter(ParameterType.Hmlz);
            calculate(sample);
            boolean bloodValid = (sample.getBloodNo() > 0);
            boolean productionDateValid = (sample.getProductionDate() !=
null);
            boolean expiryDateValid = (sample.getExpiryDate() != null);
            boolean htcInitialValid = ((htc.getValue(0) != -1f));
            boolean hbInitialValid = ((hb.getValue(0) != -1f));
            boolean htcExpiryValid = ((htc.getValue(1) != -1f));
            boolean hbExpiryValid = ((hb.getValue(1) != -1f));
            boolean hmlzValid = ((hmlz.getValue(1) != -1f));

            isInitialParameterValid = bloodValid && htcInitialValid &&
hbInitialValid && productionDateValid;
            isExpiryParameterValid = bloodValid && htcExpiryValid &&
hbExpiryValid && expiryDateValid && hmlzValid;

            expiringSamplesAreValid = (expiringSamplesAreValid &&
isExpiryParameterValid);
            initialSamplesAreValid = (initialSamplesAreValid &&
isInitialParameterValid);
        }

        boolean sampleCountValid = samples.size() >=
getNumberOfSamplesToBeTested();

        if (initialSamplesAreValid && sampleCountValid) {
            state = FormState.TESTS_WAITING;
        }

        boolean otherConditions = (isGroupingCheck() && sampleCountValid &&
isScreeningCheck()
                && isStorageBeforeQCCheck() && isStorageDuringQCCheck()
                && isStorageExterminationCheck() &&
!getQcSpecialist().equals(""));

        if (expiringSamplesAreValid && initialSamplesAreValid &&
otherConditions && sampleCountValid) {
            state = FormState.DONE;
        }
        super.setState(state);
    }

    // TODO : Hemoliz hesaplamasını öğren!!!
    private void calculate(Sample sample) {
        Parameter htc = sample.getParameter(ParameterType.Htc);
        Parameter hb = sample.getParameter(ParameterType.Hb);
        float value = 0f;
        try {
            value = htc.getValue(1) / hb.getValue(1);
        } catch (NullPointerException npe) {
            value = -1f;
        }
        sample.add(ParameterType.Hmlz, 1, value);
    }

    @Override
    public void checkResults() {
        for (Sample sample : getSamples().values()) {
            calculate(sample);
        }

        /*
           * Hb, Htc ve Hmlz parametreleri için kontroller yapılıyor
Sonuçlar Forms->results map'inin içinde boolean
           * olarak tutuluyor.
           */
```

```
      checkSpecificParameter(ParameterType.Hb, 0, 1, minHbQValue,
Float.NaN, 100);
      checkSpecificParameter(ParameterType.Htc, 0, 1, minHtcQValue,
maxHtcQValue, 100);
      checkSpecificParameter(ParameterType.Hmlz, 1, 1, Float.NaN,
maxHmlzQValue, 100);
    }


}
```

TSForm

```
package com.ptah.kankalite.domain;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Group;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Text;

import com.ptah.kankalite.constants.FormNames;
import com.ptah.kankalite.constants.FormState;
import com.ptah.kankalite.constants.FormType;
import com.ptah.kankalite.constants.ParameterType;

public class TSForm extends Form implements Serializable {

    private static final long serialVersionUID = 118686217882937243L;
    private float minVolumeQValue;
    private float maxResLeukQValue;
    private float minNoTQValue;
    private float maxPhQValue;
    private float minPhQValue;

    transient Text volumeText;
    transient Text resLeukText;
    transient Text pHText;
    transient Text noTText;


    public TSForm(String period) {
      super();
      minPhQValue = 6.4f;
      maxPhQValue = 7.4f;
      minNoTQValue = 60f;
      maxResLeukQValue = 0.2f;
      minVolumeQValue = 40.0f;
      minNoTQValue = 60.0f;
      expiryPeriod = 4;
      leastSampleCount = 10;
      condition = "%1 of number of products; with a minimum of 10 units";
      name = FormNames.TS;
      setType(FormType.TS);
      setPeriod(period);
      setState(FormState.SAMPLE_ABSENT);
      id = System.currentTimeMillis();
      addParameterType(ParameterType.Vol);
      addParameterType(ParameterType.ResLeuk);
      addParameterType(ParameterType.NoT);
      addParameterType(ParameterType.Ph);
```

```
    }

    @Override
    public void checkState() {
      FormState state = FormState.SAMPLE_ABSENT;
      boolean expiringSamplesAreValid = true;
      boolean initialSamplesAreValid = true;
      Collection<Sample> samples = getSamples().values();

      for (Sample sample : samples) {
          boolean isInitialParameterValid = false;
          boolean isExpiryParameterValid = false;
          Parameter vol = sample.getParameter(ParameterType.Vol);
          Parameter resLeuk = sample.getParameter(ParameterType.ResLeuk);
          Parameter not = sample.getParameter(ParameterType.NoT);
          Parameter ph = sample.getParameter(ParameterType.Ph);
          boolean bloodValid = (sample.getBloodNo() > 0);
          boolean productionDateValid = (sample.getProductionDate() !=
null);
          boolean expiryDateValid = (sample.getExpiryDate() != null);
          boolean volInitialValid = ((vol.getValue(0) != -1f));
          boolean resLeukInitialValid = ((resLeuk.getValue(0) != -1f));
          boolean notInitialValid = ((not.getValue(0) != -1f));
          boolean phExpiryValid = ((ph.getValue(1) != -1f));

          isInitialParameterValid = bloodValid && volInitialValid &&
resLeukInitialValid && notInitialValid
                && productionDateValid && expiryDateValid;
          isExpiryParameterValid = bloodValid && phExpiryValid;

          expiringSamplesAreValid = (expiringSamplesAreValid &&
isExpiryParameterValid);
          initialSamplesAreValid = (initialSamplesAreValid &&
isInitialParameterValid);
      }

      boolean sampleCountValid = samples.size() >=
getNumberOfSamplesToBeTested();

      if (sampleCountValid && initialSamplesAreValid) {
          state = FormState.TESTS_WAITING;
      }

      boolean otherConditions = (isGroupingCheck() && sampleCountValid &&
isScreeningCheck()
            && isStorageBeforeQCCheck() && isStorageDuringQCCheck()
            && isStorageExterminationCheck() &&
!getQcSpecialist().equals(""));

      if (expiringSamplesAreValid && state == FormState.TESTS_WAITING &&
otherConditions && sampleCountValid) {
          state = FormState.DONE;
      }
      super.setState(state);
    }

    @Override
    public void checkResults() {
      checkSpecificParameter(ParameterType.Vol, 0, 1, minVolumeQValue,
Float.NaN, 100);
      checkSpecificParameter(ParameterType.ResLeuk, 0, 1, Float.NaN,
maxResLeukQValue, 100);
      checkSpecificParameter(ParameterType.NoT, 0, 1, minNoTQValue,
Float.NaN, 75);
      checkSpecificParameter(ParameterType.Ph, 1, 1, minPhQValue,
maxPhQValue, 100);
    }
```

```
    public List<String> getSampleParameterNames() {
      List<String> result = new ArrayList<String>();
      result.add("Volume @1st Day");
      result.add("ResLeukocyte @1st Day");
      result.add("Number of platelets @1st Day");
      result.add("pH @4th Day");
      return result;
    }

    public List<Float> getSampleParameterValues(int key) {
      List<Float> result = new ArrayList<Float>();
      result.add(getSamples().get(key).getParameters().get(ParameterType.
Vol).getValue(0));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
ResLeuk).getValue(0));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
NoT).getValue(0));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
Ph).getValue(1));
      return result;
    }

    @Override
    public void showParameterEntries(Group parameterEntry) {
      parameterEntry.setBounds(170, 175, 390, 75);

      Label volumeLabel = new Label(parameterEntry, SWT.NONE);
      volumeLabel.setAlignment(SWT.RIGHT);
      volumeLabel.setText("Volume (mL) :");
      volumeLabel.setBounds(33, 10, 85, 13);

      Label resLeukLabel = new Label(parameterEntry, SWT.NONE);
      resLeukLabel.setAlignment(SWT.RIGHT);
      resLeukLabel.setText("Residual Leukocyte :");
      resLeukLabel.setBounds(3, 30, 115, 13);

      Label pHLabel = new Label(parameterEntry, SWT.NONE);
      pHLabel.setAlignment(SWT.RIGHT);
      pHLabel.setText("pH value :");
      pHLabel.setBounds(13, 50, 105, 13);

      Label noTLabel = new Label(parameterEntry, SWT.NONE);
      noTLabel.setAlignment(SWT.LEFT);
      noTLabel.setText("# of plateletes :");
      noTLabel.setBounds(190, 10, 80, 13);

      volumeText = new Text(parameterEntry, SWT.BORDER);
      volumeText.setBounds(125, 10, 50, 15);

      resLeukText = new Text(parameterEntry, SWT.BORDER);
      resLeukText.setBounds(125, 30, 50, 15);

      pHText = new Text(parameterEntry, SWT.BORDER);
      pHText.setBounds(125, 50, 50, 15);

      noTText = new Text(parameterEntry, SWT.BORDER);
      noTText.setBounds(275, 10, 50, 15);

      /*createUpdateButton(parameterEntry);
      sampleUpdateButton.setText("Update");
      sampleUpdateButton.setBounds(335, 10, 50, 60);*/
    }

    @Override
    public void extractParametersFromGUI(Sample currentSample) {
      try {
            float volumeF = Float.parseFloat(volumeText.getText());
            float resLeukF = Float.parseFloat(resLeukText.getText());
```

```
            float pHF = Float.parseFloat(pHText.getText());
            float noTF = Float.parseFloat(noTText.getText());

            Parameter volume = new Parameter(0, volumeF);
            currentSample.setParameter(ParameterType.Vol, volume);

            Parameter resLeuk = new Parameter(0, resLeukF);
            currentSample.setParameter(ParameterType.ResLeuk, resLeuk);

            Parameter pH = new Parameter(1, pHF);
            currentSample.setParameter(ParameterType.Ph, pH);

            Parameter noT= new Parameter(0, noTF);
            currentSample.setParameter(ParameterType.NoT, noT);
          } catch (Exception ex) {
            System.out.println("Some parameter values are invalid - ep");
          }
    }

    @Override
    public void updateGUIFromParameters(Sample currentSample) {
      try {
            Parameter volume =
currentSample.getParameter(ParameterType.Vol);
            volumeText.setText("" + volume.getValue(0));
            Parameter resLeuk =
currentSample.getParameter(ParameterType.ResLeuk);
            resLeukText.setText("" + resLeuk.getValue(0));
            Parameter pH = currentSample.getParameter(ParameterType.Ph);
            pHText.setText("" + pH.getValue(1));
            Parameter noT =
currentSample.getParameter(ParameterType.NoT);
            noTText.setText("" + noT.getValue(0));
            checkState();
          } catch (Exception e) {
          }
    }

}
```

TDPForm

```
package com.ptah.kankalite.domain;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Group;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Text;

import com.ptah.kankalite.constants.FormNames;
import com.ptah.kankalite.constants.FormState;
import com.ptah.kankalite.constants.FormType;
import com.ptah.kankalite.constants.ParameterType;

public class TDPForm extends Form implements Serializable {

    private static final long serialVersionUID = 118686217882937243L;
    private float maxLeuQValue;
    private float maxErythQValue;
```

```
    private float maxPlateletQValue;

    transient Text resErytText;
    transient Text resLeukText;
    transient Text resPlatText;

    public TDPForm(String period) {
      super();
      maxLeuQValue = 0.1f;
      maxErythQValue = 6.0f;
      maxPlateletQValue = 50.0f;
      expiryPeriod = 4;
      leastSampleCount = 4;
      condition = "%1 of number of products; with a minimum of 10 units";
      name = FormNames.TDP;
      setType(FormType.TDP);
      setPeriod(period);
      setState(FormState.SAMPLE_ABSENT);
      id = System.currentTimeMillis();
      addParameterType(ParameterType.ResLeuk);
      addParameterType(ParameterType.ResEryt);
      addParameterType(ParameterType.ResPlat);
    }

    @Override
    public void checkState() {
      FormState state = FormState.SAMPLE_ABSENT;
      boolean initialSamplesAreValid = true;
      Collection<Sample> samples = getSamples().values();

      for (Sample sample : samples) {
          boolean isInitialParameterValid = false;
          Parameter leukocyte =
sample.getParameter(ParameterType.ResLeuk);
          Parameter erythrocyte =
sample.getParameter(ParameterType.ResEryt);
          Parameter platelet =
sample.getParameter(ParameterType.ResPlat);

          boolean bloodValid = (sample.getBloodNo() > 0);
          boolean productionDateValid = (sample.getProductionDate() !=
null);
          boolean expiryDateValid = (sample.getExpiryDate() != null);

          boolean leukInitialValid = ((leukocyte.getValue(0) != -1f));
          boolean erythInitialValid = ((erythrocyte.getValue(0) != -1f));
          boolean platInitialValid = ((platelet.getValue(0) != -1f));

          isInitialParameterValid = bloodValid && platInitialValid &&
erythInitialValid && leukInitialValid
                && productionDateValid && expiryDateValid;

          initialSamplesAreValid = (initialSamplesAreValid &&
isInitialParameterValid);
      }

      boolean sampleCountValid=samples.size() >=
getNumberOfSamplesToBeTested();

      if (sampleCountValid) {
          state = FormState.TESTS_WAITING;
      }

      boolean otherConditions=(isGroupingCheck() && isSampleCountCheck()
&& isScreeningCheck()
            && isStorageBeforeQCCheck() && isStorageDuringQCCheck() &&
isStorageExterminationCheck());
```

```
      if (state == FormState.TESTS_WAITING && otherConditions &&
sampleCountValid ) {
          state = FormState.DONE;
      }
      super.setState(state);
    }

    @Override
    public void checkResults() {
      checkSpecificParameter(ParameterType.ResEryt, 0, 1, Float.NaN,
maxErythQValue, 100);
      checkSpecificParameter(ParameterType.ResLeuk, 0, 1, Float.NaN,
maxLeuQValue, 100);
      checkSpecificParameter(ParameterType.ResPlat, 0, 1, Float.NaN,
maxPlateletQValue, 100);
    }

    public List<String> getSampleParameterNames() {
      List<String> result = new ArrayList<String>();
      result.add("ResErythrocyte @1st Day");
      result.add("ResLeuk @1st Day");
      result.add("ResPlatelet @1st Day");
      return result;
    }

    public List<Float> getSampleParameterValues(int key) {
      List<Float> result = new ArrayList<Float>();
      result.add(getSamples().get(key).getParameters().get(ParameterType.
ResEryt).getValue(0));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
ResLeuk).getValue(0));
      result.add(getSamples().get(key).getParameters().get(ParameterType.
ResPlat).getValue(0));
      return result;
    }

    @Override
    public void extractParametersFromGUI(Sample currentSample) {
      try {
            float resErytF = Float.parseFloat(resErytText.getText());
            float resLeukF = Float.parseFloat(resLeukText.getText());
            float resPlatF = Float.parseFloat(resPlatText.getText());

            Parameter resEryt = new Parameter(0, resErytF);
            currentSample.setParameter(ParameterType.ResEryt, resEryt);

            Parameter resLeuk = new Parameter(0, resLeukF);
            currentSample.setParameter(ParameterType.ResLeuk, resLeuk);

            Parameter resPlat = new Parameter(0, resPlatF);
            currentSample.setParameter(ParameterType.ResPlat, resPlat);

          } catch (Exception ex) {
            System.out.println("Some parameter values are invalid - ep");
          }

    }

    @Override
    public void showParameterEntries(Group parameterEntry) {
      parameterEntry.setBounds(170, 175, 390, 75);

      Label resErytLabel = new Label(parameterEntry, SWT.NONE);
      resErytLabel.setAlignment(SWT.RIGHT);
      resErytLabel.setText("Residual Erythrocyte :");
      resErytLabel.setBounds(33, 10, 85, 13);

      Label resLeukLabel = new Label(parameterEntry, SWT.NONE);
```

```
      resLeukLabel.setAlignment(SWT.RIGHT);
      resLeukLabel.setText("Residual Leukocyte :");
      resLeukLabel.setBounds(3, 30, 115, 13);

      Label resPlatLabel = new Label(parameterEntry, SWT.NONE);
      resPlatLabel.setAlignment(SWT.RIGHT);
      resPlatLabel.setText("Residual Platelet :");
      resPlatLabel.setBounds(13, 50, 105, 13);

      resErytText = new Text(parameterEntry, SWT.BORDER);
      resLeukText.setBounds(125, 10, 50, 15);

      resLeukText = new Text(parameterEntry, SWT.BORDER);
      resErytText.setBounds(125, 30, 50, 15);

      resPlatText = new Text(parameterEntry, SWT.BORDER);
      resPlatText.setBounds(125, 50, 50, 15);
    }

    @Override
    public void updateGUIFromParameters(Sample currentSample) {
      try {
            Parameter resEryt =
currentSample.getParameter(ParameterType.ResEryt);
            resErytText.setText("" + resEryt.getValue(0));
            Parameter resLeuk =
currentSample.getParameter(ParameterType.ResLeuk);
            resLeukText.setText("" + resLeuk.getValue(0));
            Parameter resPlat =
currentSample.getParameter(ParameterType.ResPlat);
            resPlatText.setText("" + resPlat.getValue(0));
            checkState();
          } catch (Exception e) {
          }

    }
}
```

H.2 Source Code of Data Source

FormDataSource

```
package com.ptah.kankalite.constants;

import java.io.Serializable;

public enum FormType implements Serializable {
      ES,TS,TDP;

      public static FormType convertFromString(String formTypeString) {
          FormType result=null;
          if (formTypeString.equals(FormNames.ES)) {
            result=ES;
          }
          if (formTypeString.equals(FormNames.TS)) {
            result=TS;
          }
          if (formTypeString.equals(FormNames.TDP)) {
            result=TDP;
```

```
            }
            return result;
      }
}
```

H.3 Source Code of Constants

FormType

```
package com.ptah.kankalite.constants;

import java.io.Serializable;

public enum FormType implements Serializable {
      ES,TS,TDP;

      public static FormType convertFromString(String formTypeString) {
          FormType result=null;
          if (formTypeString.equals(FormNames.ES)) {
            result=ES;
          }
          if (formTypeString.equals(FormNames.TS)) {
            result=TS;
          }
          if (formTypeString.equals(FormNames.TDP)) {
            result=TDP;
          }
          return result;
      }
}
```

FormNames

```
package com.ptah.kankalite.constants;

public class FormNames {
      public static final String Long_ES="Erythrocyte Suspension";
      public static final String Long_TS="Platelet Suspension";
      public static final String Long_TDP="Fresh Frozen Plasma";
      public static final String ES="ES";
      public static final String TS="PS";
      public static final String TDP="FFP";
}
```

FormState

```
package com.ptah.kankalite.constants;

import java.io.Serializable;
/*
 * Durum degerleri :
```

```
 * 0  : Bütün örnekler uygun, örnek sayısı tamam, yapılması gereken bütün
testler yapılmış
 * -1 : Henuz bos
 * -2 : Örnek girilmiş, ancak örnek sayısı tamam değil
 * -3 : Örnek sayısı tamam, ancak bekleyen testler var
 */
public enum FormState implements Serializable {
      SAMPLE_ABSENT,TESTS_WAITING,DONE;
}
```

Parameter Type

```
package com.ptah.kankalite.constants;

import java.io.Serializable;

public enum ParameterType implements Serializable {
      Htc,Hb,Hmlz,Vol,ResLeuk,NoT,Ph,ResPlat,ResEryt;
}
```

H.4 Source Code of UI

QCMain

```
package com.ptah.kankalite.ui;

import java.text.DateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Collection;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.ResourceBundle;

import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperExportManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.data.JRMapCollectionDataSource;

import org.eclipse.swt.SWT;
import org.eclipse.swt.events.FocusAdapter;
import org.eclipse.swt.events.FocusEvent;
import org.eclipse.swt.events.PaintEvent;
import org.eclipse.swt.events.PaintListener;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Combo;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Display;
```

```
import org.eclipse.swt.widgets.Group;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.List;
import org.eclipse.swt.widgets.Menu;
import org.eclipse.swt.widgets.MenuItem;
import org.eclipse.swt.widgets.MessageBox;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.TabFolder;
import org.eclipse.swt.widgets.TabItem;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.swt.widgets.Text;
import org.eclipse.swt.widgets.ToolBar;
import org.eclipse.swt.widgets.ToolItem;

import com.ibm.icu.util.StringTokenizer;
import com.ptah.kankalite.constants.FormNames;
import com.ptah.kankalite.constants.FormState;
import com.ptah.kankalite.constants.FormType;
import com.ptah.kankalite.domain.Form;
import com.ptah.kankalite.domain.FormContainer;
import com.ptah.kankalite.domain.QCUtil;
import com.ptah.kankalite.domain.QTree;
import com.ptah.kankalite.domain.Sample;
import com.ptah.kankalite.domain.Storage;
import com.swtdesigner.SWTResourceManager;
import com.tiff.common.ui.datepicker.DatePickerCombo;

public class QCMain {

    private Group parameterEntry;
    private Label samplesList;
    private Label separator;
    private Label productTypeLabelName;
    private Label barcodeInputLabel;
    private Label manualInputLabel;
    private Label expiryDateLabelName;
    private Label samplingDateLabel;
    private Table samplesTable;
    private List sampleList;
    private DatePickerCombo expiryDateLabel;
    private Text productTypeLabel;
    private Label bceLabel_6;
    private Combo qcSpecialistCombo;
    private Label qcSpecialistLabel_1;
    private Text bceText_5;
    private Label bceLabel_5;
    private Text bceText_4;
    private Label bceLabel_4;
    private Text bceText_3;
    private Label bceLabel_3;
    private Text bceText_2;
    private Label bceLabel_2;
    private Text bceText_1;
    private Label bceLabel_1;
    private Text bceText;
    private Label bceLabel;
    private Combo exterminationOKCombo;
    private Combo conditionsOKDuringTestingCombo;
    private Combo conditionsOKBeforeTestingCombo;
    private Button nextButton;
    private Button previousButton;
    private Button saveButton1;
    private Button cancelButton1;
    private Button nextButton_4;
    private Button previousButton_4;
    private Button nextButton_3;
    private Button previousButton_3;
```

```
private Button nextButton_2;
private Button previousButton_2;
private Combo groupingOk;
private Combo screeningOk;
private Text qcFrequency;
private Button tab1_next;
private Text limitNoOfSampleCount;
private Text leastSampleCount;
private Text noOfProducedComponents;
private Text expiryDateBarcodeText;
private DatePickerCombo bloodCollectionDateText;
private Text productTypeBarcodeText;
private DatePickerCombo samplingDateText;
private Text unitNumberBarcodeText;
private Label bloodCollectionDateLabel;
private Text unitNumberText;
private Label unitNumberLabel;
private Group noSamplesGroup;
private List logList;
protected Shell shell, childShell;
Combo formTypeInChild;
Text periodTextInChild, bloodNoInChild;
Button sampleUpdateButton;
Group sampleInputGroup;
TabItem sampleInputTabItem;
Storage storage;
FormContainer formContainer;
FormListComposite formListComposite;
QTree formListTree;
Form currentForm;
Sample currentSample;
TabFolder tabFolder;
Button noOfSampleOkBox;
java.util.List<String> specialists = new ArrayList<String>();
Button childGoButton;

/**
 * Launch the application
 *
 * @param args
 */
public static void main(String[] args) {
  try {
      QCMain window = new QCMain();
      window.open();
  } catch (Exception e) {
      e.printStackTrace();
  }
}

/**
 * Open the window
 */
public void open() {
  final Display display = Display.getDefault();
  createContents();
  loadOrCreateForms();
  shell.open();
  shell.layout();
  while (!shell.isDisposed()) {
      if (!display.readAndDispatch())
        display.sleep();
  }
}

// TODO Onay veren kisi isimlerini setting'den girilebilir yap

protected void loadOrCreateForms() {
```

```
      try {
          formContainer = (FormContainer) storage.load();
      } catch (Exception e) {
          System.out.print(e.getMessage());
          System.out.println("\r\nCreating a new data store");
          formContainer = new FormContainer();
      }

      formContainer.createFormsForYear(FormType.ES, "2007");
      formContainer.createFormsForYear(FormType.TS, "2007");
      formContainer.createFormsForYear(FormType.TDP, "2007");

      if (formListTree != null) {
          Map<String, Form> formEntries;

          formEntries = formContainer.getForms(FormType.ES);
          if (formEntries != null) {
            for (String formName : formEntries.keySet()) {
                formListTree.addItem(formName,
formEntries.get(formName));
            }
          }

          formEntries = formContainer.getForms(FormType.TS);
          if (formEntries != null) {
            for (String formName : formEntries.keySet()) {
                formListTree.addItem(formName,
formEntries.get(formName));
            }
          }

          formEntries = formContainer.getForms(FormType.TDP);
          if (formEntries != null) {
            for (String formName : formEntries.keySet()) {
                formListTree.addItem(formName,
formEntries.get(formName));
            }
          }
      }
      updateQCList();
    }

    /**
     * Create contents of the window
     */
    protected void createContents() {
      storage = new Storage();
      shell = new Shell();
      shell.setSize(934, 662);
      shell.setText("BOUN BloodQCM v1.0");

      try {
          ResourceBundle bundle = ResourceBundle.getBundle("ptahbc");
          String specialistString = bundle.getString("specialists");
          StringTokenizer tokenizer = new
StringTokenizer(specialistString, ",");
          while (tokenizer.hasMoreTokens()) {
            String temp = tokenizer.nextToken();
            temp = temp.trim();
            specialists.add(temp);
          }
      } catch (Exception e) {
          System.out.println("Check ptahbc.properties file!!! " +
e.getMessage());
          System.exit(-1);
      }
```

```
      // TODO sample'in bütün field'larını doldur, ve sample'ın save
edildiğinden emin ol.
      // samples group'undaki her buton bunu çağırmalı
      formListComposite = new FormListComposite(shell, SWT.NONE);
      formListComposite.getAlarmList().setSize(210, 110);
      formListComposite.getAlarmList().setLocation(10, 345);
      formListComposite.setVisible(true);
      formListComposite.setBounds(10, 36, 230, 459);

      formListTree = formListComposite.getTree();
      if (formListTree == null) {
          formListTree = new QTree(formListComposite, SWT.BORDER);
      }

      /*
       * FormListTree selectionAdapter
       */
      formListTree.getTree().addSelectionListener(new SelectionAdapter()
{
          public void widgetSelected(SelectionEvent e) {
            page2Unbind();
            formSelection();
          }
      });

      formListTree.addItem("Eryhtrocyte Ssp");
      formListTree.addItem("Platelet Ssp");
      formListTree.addItem("Fresh Frozen Plasma Ssp");

      tabFolder = new TabFolder(shell, SWT.NONE);
      tabFolder.setEnabled(false);

      tabFolder.setBounds(250, 36, 661, 311);
      tabFolder.setLayout(null);

      final TabItem numberOfSamplesTabItem = new TabItem(tabFolder,
SWT.NONE);
      numberOfSamplesTabItem.setText("Number Of Samples");

      noSamplesGroup = new Group(tabFolder, SWT.NONE);

      noSamplesGroup.addPaintListener(new PaintListener() {
          public void paintControl(PaintEvent e) {
            if (currentForm != null) {
                // String leastSampleCountStr=leastSampleCount.getText();
                qcFrequency.setText(currentForm.getCondition());
                int t =
formContainer.getNeededSampleCount(currentForm.getType(),
currentForm.getPeriod());
                currentForm.setNumberOfSamplesToBeTested(t);
                leastSampleCount.setText("" +
currentForm.getNumberOfSamplesToBeTested());
            }
          }
      });

      numberOfSamplesTabItem.setControl(noSamplesGroup);

      final Label label_1 = new Label(noSamplesGroup, SWT.NONE);
      label_1.setAlignment(SWT.RIGHT);
      label_1.setBounds(66, 68, 181, 13);
      label_1.setText("The number of produced components");

      noOfProducedComponents = new Text(noSamplesGroup, SWT.BORDER);
      noOfProducedComponents.addFocusListener(new FocusAdapter() {
          public void focusLost(FocusEvent e) {
            try {
```

```
                int producedComponentCount =
Integer.parseInt(noOfProducedComponents.getText());

formContainer.setGeneratedProductCount(currentForm.getType(),
currentForm.getPeriod(),
                        producedComponentCount);
                //
leastSampleCount.setText(""+currentForm.getNumberOfSamplesToBeTested());
                int t =
formContainer.getNeededSampleCount(currentForm.getType(),
currentForm.getPeriod());
                int oldSampleCount =
currentForm.getNumberOfSamplesToBeTested();
                currentForm.setNumberOfSamplesToBeTested(t);
                currentForm.createSamples(oldSampleCount);
                leastSampleCount.setText("" +
currentForm.getNumberOfSamplesToBeTested());
                // currentForm.setNumberOfSamplesToBeTested();
                System.out.println("Current forms' sample count is : "
                        +
Integer.parseInt(noOfProducedComponents.getText()));
            } catch (Exception exc) {
                System.out.println("No form selected");
                // Do nothing
            }
          }
      });
      noOfProducedComponents.setBounds(253, 65, 36, 19);

      final Label qcFrequencyLabel = new Label(noSamplesGroup, SWT.NONE);
      qcFrequencyLabel.setAlignment(SWT.RIGHT);
      qcFrequencyLabel.setBounds(66, 97, 181, 25);
      qcFrequencyLabel.setText("QC Frequency");

      final Label theNumberOfLabel = new Label(noSamplesGroup, SWT.NONE);
      theNumberOfLabel.setAlignment(SWT.RIGHT);
      theNumberOfLabel.setBounds(72, 128, 174, 13);
      theNumberOfLabel.setText("The number of samples must be ");

      leastSampleCount = new Text(noSamplesGroup, SWT.BORDER);
      leastSampleCount.addFocusListener(new FocusAdapter() {
          public void focusGained(FocusEvent e) {
            try {
                int number =
Integer.parseInt(leastSampleCount.getText());

            } catch (Exception ex) {
                // Do nothing
            }
          }
      });
      leastSampleCount.setEditable(false);
      leastSampleCount.setBounds(252, 127, 36, 19);

      noOfSampleOkBox = new Button(noSamplesGroup, SWT.CHECK);
      noOfSampleOkBox.setBounds(296, 129, 35, 16);
      noOfSampleOkBox.setText("OK");

      final Label limitTheNumberLabel = new Label(noSamplesGroup,
SWT.NONE);
      limitTheNumberLabel.setVisible(false);
      limitTheNumberLabel.setAlignment(SWT.RIGHT);
      limitTheNumberLabel.setBounds(66, 162, 181, 25);
      limitTheNumberLabel.setText("Limit the number of samples");

      limitNoOfSampleCount = new Text(noSamplesGroup, SWT.BORDER);
      limitNoOfSampleCount.setVisible(false);
      limitNoOfSampleCount.setBounds(253, 159, 36, 19);
```

```
      final Button okButton_1 = new Button(noSamplesGroup, SWT.CHECK);
      okButton_1.setVisible(false);
      okButton_1.setBounds(295, 160, 36, 16);
      okButton_1.setText("OK");

      tab1_next = new Button(noSamplesGroup, SWT.NONE);
      // tab1_next.addSelectionListener(new StoreCurrentPageHandler());
      tab1_next.addSelectionListener(new Page1NextTabHandler());
      tab1_next.setBounds(385, 258, 50, 23);
      tab1_next.setText("Next");

      qcFrequency = new Text(noSamplesGroup, SWT.BORDER);
      qcFrequency.setText("%1 of number of products; with a minimum of 10
units");
      qcFrequency.setEditable(false);
      qcFrequency.setBounds(253, 94, 281, 22);

      /*
       * tab1_next.addSelectionListener(new SelectionAdapter() { public
void widgetSelected(SelectionEvent e) { try {
       * int t = Integer.parseInt(leastSampleCount.getText()); if (t !=
0) {
       * formContainer.setGeneratedProductCount(currentForm.getType(),
currentForm.getPeriod(), t);
       * System.out.println("Current form's count = " + t); int
sampleCount = currentForm.getSamples().size();
       * currentForm.createSamples(sampleCount); } } catch (Exception ex)
{ ex.printStackTrace(); }
       * populateSampleGUIParameters(); updateSampleGUI();
updateGUIFromParameters(currentForm.getType());
       * updateQCList(); } }); --------------------
       */

      final TabItem otherQcTestsTabItem = new TabItem(tabFolder,
SWT.NONE);
```

FomListComposite

```
package com.ptah.kankalite.ui;

import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.layout.FormAttachment;
import org.eclipse.swt.layout.FormData;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.List;

import com.ptah.kankalite.domain.Form;
import com.ptah.kankalite.domain.QTree;
import com.swtdesigner.SWTResourceManager;

public class FormListComposite extends Composite {

    //private StyledText styledText;
    private List alarmList;
    private QTree tree;
    //private TreeItem[] item=new TreeItem[3];
    Form selectedForm;
    /**
     * Create the composite
     * @param parent
     * @param style
```

```
     */
    public FormListComposite(Composite parent, int style) {
      super(parent, style);

      alarmList = new List(this, SWT.BORDER|SWT.V_SCROLL|SWT.H_SCROLL);
      alarmList.setBounds(5, 350, 200, 100);

      final Label qcAlarmListLabel = new Label(this, SWT.NONE);
      qcAlarmListLabel.setBounds(15, 320, 200, 25);
      qcAlarmListLabel.setFont(SWTResourceManager.getFont("", 14,
SWT.BOLD));
      qcAlarmListLabel.setAlignment(SWT.CENTER);
      qcAlarmListLabel.setText("QC Alarm List");

      final Label formListLabel = new Label(this, SWT.NONE);
      formListLabel.setBounds(15, 10, 200, 25);
      formListLabel.setFont(SWTResourceManager.getFont("", 14,
SWT.BOLD));
      formListLabel.setAlignment(SWT.CENTER);
      formListLabel.setText("QC Form List");


      tree = new QTree(this, SWT.BORDER);
      final FormData fd_tree = new FormData();
      fd_tree.left = new FormAttachment(0, 20);
      fd_tree.right = new FormAttachment(0, 210);
      fd_tree.bottom = new FormAttachment(0, 275);
      fd_tree.top = new FormAttachment(0, 25);
      tree.getTree().setLayoutData(fd_tree);
/*          tree.getTree().addSelectionListener(new SelectionAdapter() {
            public void widgetSelected(SelectionEvent e) {
                selectedForm = tree.getSelectedForm();
                if (selectedForm!=null) {
                }
            }
      });*/
    }

    @Override
    public void dispose() {
      super.dispose();
    }

    @Override
    protected void checkSubclass() {
      // Disable the check that prevents subclassing of SWT components
    }

    public QTree getTree() {
        return tree;
    }

    public void setTree(QTree tree) {
        this.tree = tree;
    }

    public Form getSelectedForm() {
        return selectedForm;
    }

    public void setSelectedForm(Form selectedForm) {
        this.selectedForm = selectedForm;
    }

    public List getAlarmList() {
        return alarmList;
    }
```

```
    public void setAlarmList(List alarmList) {
        alarmList = alarmList;
    }

}
```

H.5 Test Source Codes of Domain

FormContainerTester

```
package com.ptah.kankalite.domain;

import java.util.Calendar;
import java.util.Map;

import junit.framework.TestCase;

import com.ptah.kankalite.constants.FormState;
import com.ptah.kankalite.constants.FormType;
import com.ptah.kankalite.constants.ParameterType;

public class FormContainerTester extends TestCase {
    String year = "";
    Calendar calendar = Calendar.getInstance();

    public void setUp() {
      year = "" + calendar.get(Calendar.YEAR);
    }

    public void testCreateformContainer() {
      FormContainer formContainer = new FormContainer();
      assertNotNull(formContainer);
    }

    public void testFormGeneration() {
      FormContainer formContainer = new FormContainer();
      // İlgili dönemlerde üretilen ürün sayıları set ediliyor
      formContainer.setGeneratedProductCount(FormType.ES, "012007", 100);
      formContainer.setGeneratedProductCount(FormType.ES, "022007", 130);
      formContainer.setGeneratedProductCount(FormType.ES, "022007", 150);
      // En son yazilan deger gecerli
      formContainer.setGeneratedProductCount(FormType.ES, "032007", 60);
      int actual = formContainer.getGeneratedProductCount(FormType.ES,
"022007");
      assertEquals(150, actual);
      actual = formContainer.getGeneratedProductCount(FormType.ES,
"012007");
      assertEquals(100, actual);
      actual = formContainer.getGeneratedProductCount(FormType.ES,
"052009");
      assertEquals(-1, actual);
      actual = formContainer.getGeneratedProductCount(FormType.TS,
"012007");
      assertEquals(-1, actual);
    }

    public void testStorage() {
      Storage storage = new Storage();
      FormContainer formContainer = new FormContainer();
```

```
      formContainer.setGeneratedProductCount(FormType.ES, "01122007",
100);
      formContainer.setGeneratedProductCount(FormType.ES, "022007", 200);
      formContainer.setGeneratedProductCount(FormType.ES, "032007", 300);
      formContainer.setGeneratedProductCount(FormType.TS, "022007", 15);
      Form esForm=formContainer.getForm(FormType.ES, "022007");

      Map<Integer, Sample> samples= esForm.getSamples();
      Sample sample=samples.get(0);
      sample.add(ParameterType.Hb, 0, 1.1f);
      sample.add(ParameterType.Hb, 1, 1.3f);
      samples.put(1, sample);
      storage.store(formContainer);

      FormContainer newFormContainer = new FormContainer();
      try {
          newFormContainer = (FormContainer) storage.load();
      } catch (Exception e) {
          newFormContainer = new FormContainer();

      }
      assertEquals(200,
newFormContainer.getGeneratedProductCount(FormType.ES, "022007"));
      assertEquals(15,
newFormContainer.getGeneratedProductCount(FormType.TS, "022007"));

      esForm=formContainer.getForm(FormType.ES, "022007");
      samples= esForm.getSamples();
      assertEquals(1.1f,
samples.get(0).getParameter(ParameterType.Hb).getValue(0));
      samples.put(1, sample);
    }

    public void testSampleCount() {
      FormContainer formContainer = new FormContainer();
      formContainer.setGeneratedProductCount(FormType.ES, "012007", 120);
      formContainer.setGeneratedProductCount(FormType.ES, "022007", 130);
      formContainer.setGeneratedProductCount(FormType.ES, "032007", 80);
      formContainer.setGeneratedProductCount(FormType.TS, "032007", 100);
      formContainer.setGeneratedProductCount(FormType.TS, "042007", 680);
      formContainer.setGeneratedProductCount(FormType.TS, "052007", 630);
      formContainer.setGeneratedProductCount(FormType.TS, "062007", 600);
      formContainer.setGeneratedProductCount(FormType.TS, "072007", 601);
      formContainer.setGeneratedProductCount(FormType.TS, "082007", 400);
      formContainer.setGeneratedProductCount(FormType.TS, "092007", 401);
      int countForFeb = formContainer.getNeededSampleCount(FormType.ES,
"022007");
      assertEquals(4, countForFeb);
      int countFor100 = formContainer.getNeededSampleCount(FormType.TS,
"032007");
      assertEquals(10, countFor100);
      int countFor680 = formContainer.getNeededSampleCount(FormType.TS,
"042007");
      assertEquals(10, countFor680);
      int countFor630 = formContainer.getNeededSampleCount(FormType.TS,
"052007");
      assertEquals(10, countFor630);
      int countFor600 = formContainer.getNeededSampleCount(FormType.TS,
"062007");
      assertEquals(10, countFor600);
      int countFor601 = formContainer.getNeededSampleCount(FormType.TS,
"072007");
      assertEquals(10, countFor601);
      int countFor400 = formContainer.getNeededSampleCount(FormType.TS,
"082007");
      assertEquals(10, countFor400);
      int countFor401 = formContainer.getNeededSampleCount(FormType.TS,
"092007");
```

```
      assertEquals(10, countFor401);
    }

    public void testSampleCountWithUserForcedValue() {
      FormContainer formContainer = new FormContainer();
      int userValue = 3;
      formContainer.setUserSampleCount(userValue);
      formContainer.setGeneratedProductCount(FormType.ES, "012007", 120);
      formContainer.setGeneratedProductCount(FormType.ES, "022007", 130);
      formContainer.setGeneratedProductCount(FormType.ES, "032007", 80);
      formContainer.setGeneratedProductCount(FormType.TS, "032007", 100);
      formContainer.setGeneratedProductCount(FormType.TS, "042007", 680);
      formContainer.setGeneratedProductCount(FormType.TS, "052007", 630);
      formContainer.setGeneratedProductCount(FormType.TS, "062007", 600);
      formContainer.setGeneratedProductCount(FormType.TS, "072007", 601);
      formContainer.setGeneratedProductCount(FormType.TS, "082007", 400);
      formContainer.setGeneratedProductCount(FormType.TS, "092007", 401);
      int countForFeb = formContainer.getNeededSampleCount(FormType.ES,
"022007");
      assertEquals(userValue, countForFeb);
      int countFor100 = formContainer.getNeededSampleCount(FormType.TS,
"032007");
      assertEquals(userValue, countFor100);
      int countFor680 = formContainer.getNeededSampleCount(FormType.TS,
"042007");
      assertEquals(userValue, countFor680);
      int countFor630 = formContainer.getNeededSampleCount(FormType.TS,
"052007");
      assertEquals(userValue, countFor630);
      int countFor600 = formContainer.getNeededSampleCount(FormType.TS,
"062007");
      assertEquals(userValue, countFor600);
      int countFor601 = formContainer.getNeededSampleCount(FormType.TS,
"072007");
      assertEquals(userValue, countFor601);
      int countFor400 = formContainer.getNeededSampleCount(FormType.TS,
"082007");
      assertEquals(userValue, countFor400);
      int countFor401 = formContainer.getNeededSampleCount(FormType.TS,
"092007");
      assertEquals(userValue, countFor401);
    }

    public void testAreGeneratedFormsDifferent() {
      FormContainer formContainer = new FormContainer();
      String period1 = "012007";
      String period2 = "022007";
      String period3 = "042007";
      formContainer.setGeneratedProductCount(FormType.ES, period1, 120);
      formContainer.setGeneratedProductCount(FormType.ES, period2, 130);
      formContainer.setGeneratedProductCount(FormType.TS, period3, 280);
      Map<String, Form> forms = formContainer.getForms(FormType.ES);
      Form form1 = forms.get(period1);
      Form form2 = forms.get(period2);
      assertNotSame(form1.getId(), form2.getId());
    }

    public void testGeneratedFormsCount() {
      FormContainer formContainer = new FormContainer();
      formContainer.setGeneratedProductCount(FormType.ES, "01" + year,
120);
      formContainer.setGeneratedProductCount(FormType.ES, "02" + year,
130);
      formContainer.setGeneratedProductCount(FormType.ES, "03" + year,
80);
      formContainer.setGeneratedProductCount(FormType.TS, "03" + year,
180);
      Map<String, Form> forms = formContainer.getForms(FormType.ES);
```

```
      assertEquals(3, forms.size());
    }

    /*
         * Container'in formList objesini populate eder
         */
    public void testFormCreatorForThisYear() {
      FormContainer container = new FormContainer();
      container.createFormsForYear(FormType.ES, year);
      assertEquals(12, container.getForms(FormType.ES).size());
    }

    /*
         * İçinde bulunulan yıla ait form listesini döner
         */
    public void testFormsForCurrentYear() {
      FormContainer container = new FormContainer();
      container.createFormsForYear(FormType.ES, year);
      Map<String, Form> forms =
container.getFormsToBeCompletedForYear(FormType.ES);
      assertEquals(12, forms.size());
      container.createFormsForYear(FormType.ES, "2005");
      assertEquals(24,
container.getFormsToBeCompletedForYear(FormType.ES).size());
    }

    /*
         * İçinde bulunulan yıla ait daha önceden kısmen doldurulmuş
formList'i
         * doldurur ve eksik olanları döner
         */
    public void testFormsForCurrentYearWithPartlyPopulated() {
      FormContainer container = new FormContainer();
      container.createFormsForYear(FormType.ES, year);
      // createFormsForThisYear metodu güvenli mi? İki defa çalışınca
      // birşeyleri bozmasın
      container.createFormsForYear(FormType.ES, year);
      Map<String, Form> forms = container.getForms(FormType.ES);
      // Bir yıl için 12 tane form dönmesini bekliyoruz.
      assertEquals(12, forms.size());

      forms.remove("01" + year);
      assertEquals(11, forms.size());
      Map<String, Form> formsTBC =
container.getFormsToBeCompletedForYear(FormType.ES);
      assertEquals(11, formsTBC.size());
      assertEquals(11, forms.size());
    }

    public void testDifferentFormsExist() {
      FormContainer container = new FormContainer();
      container.createFormsForYear(FormType.ES, year);
      container.createFormsForYear(FormType.TS, year);
      Map<String, Form> tSForms = container.getForms(FormType.TS);
      Map<String, Form> eSForms = container.getForms(FormType.ES);
      assertEquals(12, tSForms.size());
      assertEquals(FormType.TS, ((Form)
(tSForms.values().toArray()[0])).getType());
      assertEquals(FormType.ES, ((Form)
(eSForms.values().toArray()[0])).getType());
    }

    public void testTSFormsStatesChangingWithSamples() {
      FormContainer container = new FormContainer();
      Form tsForm;
      container.createFormsForYear(FormType.TS, year);
      container.setGeneratedProductCount(FormType.TS, "02" + year, 150);
```

```
      Map<String, Form> forms =
container.getFormsToBeCompletedForYear(FormType.TS);
      tsForm = forms.get("02" + year);
      tsForm.setScreeningCheck(true);
      tsForm.setGroupingCheck(true);
      tsForm.setStorageBeforeQCCheck(true);
      tsForm.setStorageDuringQCCheck(true);
      tsForm.setStorageExterminationCheck(true);
      tsForm.setQcSpecialist("AAA");
      Map<Integer, Sample> samples = tsForm.getSamples();
      assertNotSame(samples.get(0), samples.get(1));
      int sampleSize = samples.size();
      assertEquals(10, samples.size());
      assertEquals(4, samples.get(0).getExpiryPeriod());
      assertEquals(FormState.SAMPLE_ABSENT, tsForm.getState());
      for (int i = 0; i < sampleSize; i++) {
          samples.get(i).setBloodNo(1001 + i);
          samples.get(i).setProductionDate(calendar.getTime());
          samples.get(i).add(ParameterType.Vol, 0, 100f + i);
          samples.get(i).add(ParameterType.Vol, 1, 41f + i);
          samples.get(i).add(ParameterType.ResLeuk, 0, 11f + i);
          assertEquals(FormState.SAMPLE_ABSENT, tsForm.getState());
          samples.get(i).add(ParameterType.NoT, 0, 22f + i);
      }
      assertEquals(FormState.TESTS_WAITING, tsForm.getState());

      for (int i = 0; i < sampleSize; i++) {
          assertEquals(FormState.TESTS_WAITING, tsForm.getState());
          samples.get(i).add(ParameterType.Ph, 1, 7f + i);
      }

      assertEquals(FormState.DONE, tsForm.getState());
      tsForm.checkResults();
      assertEquals(0, tsForm.getResult(ParameterType.Vol));
    }

    public void testTSFormsStatesChangingWithSamplesWithInvalidValues()
{
      FormContainer container = new FormContainer();
      Form tsForm;
      container.createFormsForYear(FormType.TS, year);
      container.setGeneratedProductCount(FormType.TS, "02" + year, 150);
      Map<String, Form> forms =
container.getFormsToBeCompletedForYear(FormType.TS);
      tsForm = forms.get("02" + year);
      Map<Integer, Sample> samples = tsForm.getSamples();
      assertNotSame(samples.get(0), samples.get(1));
      int sampleSize = samples.size();
      assertEquals(10, samples.size());
      assertEquals(4, samples.get(0).getExpiryPeriod());
      assertEquals(FormState.SAMPLE_ABSENT, tsForm.getState());
      for (int i = 0; i < sampleSize; i++) {
          samples.get(i).setBloodNo(1001 + i);
          samples.get(i).setProductionDate(calendar.getTime());
          samples.get(i).add(ParameterType.Vol, 0, 100f + i);
          samples.get(i).add(ParameterType.Vol, 1, 41f + i);
          samples.get(i).add(ParameterType.ResLeuk, 0, 11f + i);
          assertEquals(FormState.SAMPLE_ABSENT, tsForm.getState());
          samples.get(i).add(ParameterType.NoT, 0, 22f + i);
      }
      assertEquals(FormState.TESTS_WAITING, tsForm.getState());

      for (int i = 0; i < sampleSize; i++) {
          assertEquals(FormState.TESTS_WAITING, tsForm.getState());
          samples.get(i).add(ParameterType.Ph, 1, 10f + i);
      }

      assertEquals(FormState.TESTS_WAITING, tsForm.getState());
```

```
      tsForm.checkResults();
      assertEquals(0, tsForm.getResult(ParameterType.Vol));
    }

}
```

FormTester

```
package com.ptah.kankalite.domain;

import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Map;

import junit.framework.TestCase;

import com.ptah.kankalite.constants.FormState;
import com.ptah.kankalite.constants.FormType;
import com.ptah.kankalite.constants.ParameterType;

public class FormTester extends TestCase {

    String year = "";
    Calendar calendar = Calendar.getInstance();

    public void setUp() {
      year = "" + calendar.get(Calendar.YEAR);
    }

    public void testCreateESForm() {
      FormContainer container = new FormContainer();
      Form esForm = container.generateForm(FormType.ES, "022007");
      assertNotNull(esForm);
      Map<Integer, Sample> samples = esForm.getSamples();
      assertEquals(4, samples.size());
      Sample sample1 = samples.get(0);
      Sample sample2 = samples.get(1);
      Sample sample3 = samples.get(2);
      Sample sample4 = samples.get(3);
      assertEquals(0, sample1.getBloodNo());
      assertEquals(0, sample2.getBloodNo());
      assertEquals(3, sample3.getParameterTypes().size());
      assertEquals(42, sample4.getExpiryPeriod());
      assertEquals(FormState.SAMPLE_ABSENT, esForm.getState());

    }

    public void testGeneratedIds() {
      FormContainer container = new FormContainer();
      Form esForm = container.generateForm(FormType.ES, "022007");
      Form tsForm = container.generateForm(FormType.TS, "022007");
      long esId = esForm.getId();
      long tsId = tsForm.getId();
      assertNotSame(esId, tsId);
    }

    public void testSampleCountCalculation() {
      FormContainer container = new FormContainer();
      String date = "022007";
      container.setGeneratedProductCount(FormType.ES, date, 150);
      Form esForm = container.getForm(FormType.ES, date);
      assertEquals(4, esForm.getNumberOfSamplesToBeTested());
      esForm.setNumberOfSamplesToBeTested(100);
      assertEquals(100, esForm.getNumberOfSamplesToBeTested());
```

```
    }

    /*
         * FormContainer yaratılır Container'ın setGeneratedProductCount
method'u çağrılarak ilgili tipte, belirlenen
         * zamanda kaç adet ürün üretildiği set edilir. Container'ın
getForm methodu çağrılarak ilgili tipte, belirlenen
         * zaman için container tarafından üretilen form objesi alınır.
Bu form objesinin içersinde
         * setGeneratedProductCount method'unda set edilen ürün sayısına
göre belirlenen sayıda Sample objesi olması
         * beklenir.
         */
    public void testSampleCount() {
      FormContainer container = new FormContainer();
      String date = "022007";
      container.setGeneratedProductCount(FormType.ES, date, 150);
      Form esForm = container.getForm(FormType.ES, date);

      int i = 1000;
      for (Sample sample : esForm.getSamples().values()) {
          sample.setId(i++);
          sample.setProductionDate(new Date(System.currentTimeMillis()));
      }
      for (Sample sample : esForm.getSamples().values()) {
          assertNotSame(-1, sample.getId());
      }
    }

    public void testTSFormSampleCount() {
      FormContainer container = new FormContainer();
      container.createFormsForYear(FormType.TS, year);
      container.setGeneratedProductCount(FormType.ES, "02" + year, 150);
      assertEquals(10, container.getNeededSampleCount(FormType.TS, "02" +
year));
      container.setGeneratedProductCount(FormType.TS, "03" + year, 900);
      assertEquals(10, container.getNeededSampleCount(FormType.TS, "03" +
year));
      container.setGeneratedProductCount(FormType.TS, "04" + year, 1040);
      assertEquals(11, container.getNeededSampleCount(FormType.TS, "04" +
year));
      container.setGeneratedProductCount(FormType.TS, "05" + year, 1150);
      assertEquals(12, container.getNeededSampleCount(FormType.TS, "05" +
year));
    }

    public void testESFormsStatesChangingWithSamples() {
      FormContainer container = new FormContainer();
      Form esForm;
      container.createFormsForYear(FormType.ES, year);
      container.setGeneratedProductCount(FormType.ES, "02" + year, 150);
      assertEquals(4, container.getNeededSampleCount(FormType.ES, "02" +
year));
      Map<String, Form> forms =
container.getFormsToBeCompletedForYear(FormType.ES);
      esForm = forms.get("02" + year);
      Map<Integer, Sample> samples = esForm.getSamples();
      assertNotSame(samples.get(0), samples.get(1));
      assertEquals(4, samples.size());
      esForm.setScreeningCheck(true);
      esForm.setGroupingCheck(true);
      esForm.setStorageBeforeQCCheck(true);
      esForm.setStorageDuringQCCheck(true);
      esForm.setStorageExterminationCheck(true);
      esForm.setQcSpecialist("AAA");
      // Başlangıçta state, SAMPLE_ABSENT olmalı
      assertEquals(FormState.SAMPLE_ABSENT, esForm.getState());
```

```
      // Bütün parametreler -1 olacak ve toplam 12 parametre olacak (4
sample,
      // 3 parametre, 2 değer(initial - expired) )
      int counter = 0;
      for (Sample sample : samples.values()) {
          assertEquals(42, sample.getExpiryPeriod());
          for (ParameterType type : sample.getParameterTypes()) {
            Parameter parameter = sample.getParameter(type);
            List<Float> values = parameter.getValues(type);
            for (int loopVar = 0; loopVar < values.size(); loopVar++) {
                try {
                  float val = values.get(loopVar);
                  if (val == -1.0f) {
                      assertEquals(-1f, val);
                      counter++;
                  }
                } catch (Exception e) {
                  System.out.println(loopVar);
                }
            }
          }
      }
      assertEquals(12, counter);

      samples.get(0).setBloodNo(18047);
      samples.get(0).setProductionDate(calendar.getTime());
      samples.get(0).add(ParameterType.Htc, 0, 59.4f);
      samples.get(0).add(ParameterType.Hb, 0, 76.0f);
      assertEquals(FormState.SAMPLE_ABSENT, esForm.getState());

      // Alınmış olan initialSamples esForm'daki listeyle aynı yeri
işaret
      // ediyor mu? - esForm'u update etmeye gerek olmamalı!
      Sample actualSample = esForm.getSamples().get(0);
      assertEquals(18047, actualSample.getBloodNo());
      assertEquals(59.4f,
actualSample.getParameter(ParameterType.Htc).getValue(0));

      // Şimdi de bütün sample'ları update edelim
      samples.get(1).add(ParameterType.Htc, 0, 56.7f);
      samples.get(1).add(ParameterType.Hb, 0, 70.8f);
      samples.get(2).add(ParameterType.Htc, 0, 53.8f);
      samples.get(2).add(ParameterType.Hb, 0, 83.8f);
      samples.get(3).add(ParameterType.Htc, 0, 60.8f);
      samples.get(3).add(ParameterType.Hb, 0, 79f);

      Parameter parameter1 =
esForm.getSamples().get(0).getParameter(ParameterType.Htc);
      Parameter parameter2 =
esForm.getSamples().get(1).getParameter(ParameterType.Htc);

      assertNotSame(esForm.getSamples().get(0),
esForm.getSamples().get(1));
      assertNotSame(parameter1, parameter2);
      assertNotSame(samples.get(0), samples.get(1));
      assertNotSame(samples.get(0).getParameter(ParameterType.Hb).getValu
e(0), samples.get(1).getParameter(
            ParameterType.Hb).getValue(0));
      assertEquals(56.7f,
samples.get(1).getParameter(ParameterType.Htc).getValue(0));
      assertEquals(83.8f,
samples.get(2).getParameter(ParameterType.Hb).getValue(0));

      samples.get(1).setBloodNo(17750);
      Sample actualSample2 = esForm.getSamples().get(1);
      assertEquals(17750, actualSample2.getBloodNo());

      samples.get(1).setProductionDate(calendar.getTime());
```

```
      samples.get(2).setBloodNo(16665);
      samples.get(2).setProductionDate(calendar.getTime());
      assertEquals(FormState.SAMPLE_ABSENT, esForm.getState());
      samples.get(3).setBloodNo(17126);
      samples.get(3).setProductionDate(calendar.getTime());
      // Bütün sample'lar update edildiğine göre TESTS_WAITING state'e
gelmiş
      // olmamız lazım
      assertEquals(FormState.TESTS_WAITING, esForm.getState());

      samples.get(0).add(ParameterType.Htc, 1, 70.8f);
      samples.get(1).add(ParameterType.Htc, 1, 75.4f);
      samples.get(2).add(ParameterType.Htc, 1, 58.1f);
      assertEquals(FormState.TESTS_WAITING, esForm.getState());
      samples.get(0).add(ParameterType.Hb, 1, 63.7f);
      assertEquals(FormState.TESTS_WAITING, esForm.getState());
      samples.get(1).add(ParameterType.Hb, 1, 65.1f);
      samples.get(2).add(ParameterType.Hb, 1, 45.3f);
      assertEquals(FormState.TESTS_WAITING, esForm.getState());
      samples.get(3).add(ParameterType.Htc, 1, 61.8f);
      assertEquals(FormState.TESTS_WAITING, esForm.getState());
      samples.get(3).add(ParameterType.Hb, 1, 55.6f);
      assertEquals(FormState.DONE, esForm.getState());

      // Şimdi de parametre değerlerini test edelim - İki örnekte
hematokrit sınırların dışında bulunuyordu
      esForm.checkResults();
      assertEquals(0, esForm.getResult(ParameterType.Hb));
      assertEquals(2, esForm.getResult(ParameterType.Htc));
      assertEquals(0, esForm.getResult(ParameterType.Hmlz));
      assertEquals(2, esForm.getTotalFailedSampleCountForThisForm());
    }

}
```

SampleTester

```
package com.ptah.kankalite.domain;

import java.util.Calendar;
import java.util.Map;

import junit.framework.TestCase;

import com.ptah.kankalite.constants.FormType;
import com.ptah.kankalite.constants.ParameterType;

public class SampleTester extends TestCase {

      public void testSampleParameters() {
            FormContainer container=new FormContainer();
            String date="022007";
            container.setGeneratedProductCount(FormType.ES, date, 150);
            Form esForm=container.getForm(FormType.ES, date);
            Map<Integer, Sample> initialSamples=esForm.getSamples();
            /*
             * İlk örneği alıp parametrelerini kontrol ediyoruz
             */
            Sample sample=initialSamples.get(0);
            assertEquals(3,sample.getParameterTypes().size());

      assertTrue(sample.getParameterTypes().contains(ParameterType.Hb));

      assertTrue(sample.getParameterTypes().contains(ParameterType.Htc));
```

```
            assertEquals(-
1.0f,sample.getParameters().get(ParameterType.Htc).getValue(0));
      }

      public void testESFormSamples() {
            FormContainer container=new FormContainer();
            String date="022007";
            container.setGeneratedProductCount(FormType.ES, date, 150);
            Form esForm=container.getForm(FormType.ES, date);
            Map<Integer,Sample> samples=esForm.getSamples();
            assertEquals(4,samples.size());
            // ES Formunda dört örnek var, biz ilkini alıyoruz
            Sample sample=samples.get(0);
            /*
             * initialSample'da Hb bulunacak ancak Hmlz bulunmayacak
             */

      assertTrue(sample.getParameterTypes().contains(ParameterType.Hb));

      assertTrue(sample.getParameterTypes().contains(ParameterType.Htc));

      assertTrue(sample.getParameterTypes().contains(ParameterType.Hmlz))
;

      }

      public void testESFormInitialExpiringParameterTransfer() {
            FormContainer container=new FormContainer();
            String date="022007";
            container.setGeneratedProductCount(FormType.ES, date, 150);
            Form esForm=container.getForm(FormType.ES, date);


            Calendar calendar=Calendar.getInstance();
            calendar.set(2007,2,20);

            esForm.setSampleProductionDate(0,calendar.getTime());
            /* Herhangi bir sample'in production date'i set edilirse
             * buna karsilik dusen sample'in (initial - expiring)
             * expiry date'i kendiliginden set edilmeli.
             */

            calendar.add(Calendar.DAY_OF_MONTH, 42);

      assertEquals(calendar.getTime(),esForm.getSampleExpiryDate(0));
      }
}
```

DateTester

```
package com.ptah.kankalite.domain;

import java.text.DateFormat;
import java.text.ParseException;
import java.util.Date;

import junit.framework.TestCase;

public class DateTester extends TestCase {
    public void testCreateDate() throws ParseException {

      String dateStr="6/6/07";
      DateFormat formatter=DateFormat.getDateInstance(DateFormat.SHORT);
      Date actualDate=formatter.parse(dateStr);
```

```
        String actualDateStr=formatter.format(actualDate);

        /*Calendar calendar=Calendar.getInstance();
        calendar.set(2007,6,6);
        Date expectedDate=calendar.getTime();*/
        assertEquals(dateStr,actualDateStr);
    }
}
```

REFERENCES

1. Akdeniz Üniversitesi, "Kan Bankacılığı ve Transfüzyon Tıbbında Standartlar ve Kalite Kursu Notları", Antalya, 2002.
2. European Commission, Health & Consumer Protection Directorate General, "Opinion On Quality and Safety of Blood*", Decleration*, Brusseles, 2000.
3. European Council, "Guide to the preparation, Use and quality Assurance of Blood Components, 11th edition", *European Council of Publishing*, France, 2005.
4. Australian Society of Blood Transfusions, "Clinical Practice Guidelines on the Appropriate Use of Red Blood Cells", Sydney, 2000.
5. Akdeniz Üniversitesi, "Ulusal Kan Merkezleri ve Transfüzyon Tıbbı Kursu (VII) Notları", Antalya, 2002.
6. *WHO,* "AM Quality System for Blood Safety" Key Element and Requirements Notes, Switzerland, 2003, Available at*:* www.who.int/bct.
7. SANAS, South African National Accreditation System, "Accreditation of Blood Transfusion Services", Brochure, Zambia, Available at: www.sanac.co.za .
8. Kumari, S., "Quality Management in Blood Transfusion Service", World Health Organization, *BTS*, South Asia, 1998.
9. T.C. Sağlık Bakanlığı, "Organ Ve Doku Alınması, Saklanması ve Nakli" Yönetmeliği ve ilgili Kanunlar, Ankara, 2004.
10. Bayık, M., "Güvenli Kan", Kan Merkezleri ve Transfüzyonları Derneği, *Damla,* Vol. 59, pp.10-12, İstanbul, 2003.
11. Akdeniz Üniversitesi*,* "Ulusal Kan Merkezleri ve Transfüzyon Kursu V. Notları - Kan Merkezlerinde Kayıt.", Ankara, 2001.
12. Food and drug Administration, "Guidelines For Quality Assurance in Blood Establishments", Standart: 91N-450, USA, 1995.
13. European Parliament, "Quality and Safety Standards for Human Blood and Blood Components", *Decleration,* Brusseles, 2000.
14. Guyton, C., "Human Physiology", Student Edition, USA, 1991.
15. Stryer, A., "Stryer's Biochemistry", Hamburg, 1998.
16. AABB, American Association for Blood Banks, "Standarts for Blood Banks, and Transfusion Services, XX.Edition", *AABB Press*, USA, 2000.
17. Schneider, D.E., "Computer Programming Concepts and Visual Basic", Germany, 1999.
18. Kobryn, C., "Introduction to UML: Structural and Use Case Modeling", *Object Modeling with OMG UML Tutorial Series,* USA, 2006.
19. Wuyts, R., "Use Case Diagrams", *ULB – Analse et Methodologie Informatiques*, Vol.2005/2006, pp.506, Belgium, 2006.
20. Winnemiller, E., "Java Programming, Database How-To", *Objective Series,* N.Y., 1999.

21. Nehrer, P., "Java Development with Eclips Tool", 2005, Available at: www.developer.com/java/data/article.php/3528616 , pp.1-12.

22. SGS: Cortex Quality Software, "Hemotrack Blood Bank Management System for Hospitals", Data Sheet & Brochure, Belgium, Available at: www.sgscortex.com.

23. Türkiye 2inci Bilişim Şurası, e-Sağlık Çalışma Grubu, "e-Sağlık Taslak Raporu", Ankara 2004.

24. Akdeniz Üniversitesi, "Ulusal Kan Merkezleri ve Transfüzyon Tıbbı Kursu II. Notları - Kan Merkezleri arasında İletişim", Ankara, 2001.

25. Learoyd, P., "Good manifacturing practice for blood components, A Brief Guide", National Blood Service, *Council of Europe Publishing*, 1999.

26. Holzner, S., "Visual Basic 6*", Black Book Press*, N.Y., 1998.

27. Min, I.S., Choi, M.R., Kim, S.Y., Jung E.Y., Hwang, J.H., Lee, S., "Institution based Quality Assessment of Blood Transfusion in South Korea", *ISQua,* 2004.

28. Unitech, "Barkod Okuyucular için Programlama Manueli ve Teknik Referans Manueli", South Korea, 2006.

29. İgrapx, "Flowcharter Programming Manual", Corel Corporation, Germany, 2004.

30. Ergen, E., Dalkılıç, M., Akaoğlu, S., "Kan Komponentleri", Eskişehir Kızılay Kan Merkezi, *Kan,* Vol.1, pp.7-29, Eskişehir, 2002.

31. Acar, N., Koçak, N., "Transfüzyon Pratiği", Eskişehir Kızılay Kan Merkezi, *Kan,* Vol.1, pp.30-40, Eskişehir, 2002.

32. Uniform Code Council (UCC) & Health Industry Business Communications Council (HIBCC), "Eurocode-IBLS Uluslararası Kan ve Kan Ürünleri Barkod Etiketleme Standartları", *European Directive* 2002/98/EC, 2002.

33. Wright, C., "Visual C++ for Dummies - Quick Reference", *For Dummies Series*, USA, 2001.

34. Franklin, I.M., "Quality Improvement Program: Safe and Effective Transfusion in Scottish Hospitals – The Role of the Transfusion Nurse Specialist (SAET Study)", *NHS-Scotland*, Edinburgh, 2004.

35. European Commision, Health & Consumer Protection Diroctorate General, "Blood Regulatory Comitee – Summary Report" *Decleration,* Luxembourg, 2005.

36. Duman, C., Erden, B.F., "Birinci Basamak Sağlık Hizmetlerine Yönelik Biyokimyasal Laboratuvar verilerinin Kısa Yorumu", *Sted,* Vol 13, Issue.7, pp 256-262, Kocaeli, 2004.

37. Brunner, D., "PIC/S Gmp Guide For Blood Establishments", Pharmaceutical Inspection Convention, Geneva, 2004.

38. European Commission, Health & Consumer Protection Directorate General, "High Quality and Safety Standards for Human Blood and Blood Components", *Declaration*, Amsterdam, 2000.

39. SANAS, South African National Accreditation System, "Application Form and Criterias for Accreditation of Blood Transfusion Service Laboratories", Form, Zambia, Available at: www.sanac.co.za.

40. Erbaş, O., "Hastane Kan Merkezleri Çalışma Yöntemleri", İstanbul, 1999.

41. World Health Organization, "Quality Management Project For Blood Transfusion Services", Infosheet, Geneva, Switzerland, Available at: www.who.int/bct.

42. Clayberg, E., Rubel, D., "Java GUI Development for Swing, SWT, RCP and GWT: WindowBuilder Pro", *Instantiations,* USA, 2007.

43. Gray, E., "Gray's Anatomy", USA, 1930.

44. Schots, J., Cassiman, I., Tielemans, L., "Quality Assurance of Transfusion Practices in Belgium Hospitals", *Schots*, Brussels, 2002.

45. Dhringra, N., Lloyd, S.E., Fordham, J., Noryati, A.A., "Challenges in Global Blood Safety", *World Hospitals, and Health Services,* Vol.40, pp 45-49, *WHO.*

46. Kritchevsky, S., Simmons, B., "Continuous Quality Improvement: Concepts and Applications for Physician Care", *JAMA*, Vol. 266, pp.1817-1823, 1991.

47. Luciano, M., "Human Physiology and Anatomy", MacGrawHill, USA, 1991.

48. T.C. Sağlık Bakanlığı, Bilgi İşlem Daire Başkanlığı, "Türkiye Sağlık Bilgi Sistemi Eylem Planı", Rapor, Ankara, 2004.

49. World Health Organization, "Basic Operational Framework for Blood Transfusion Safety", Review, 2007, Available at: www.who.int/bct .