Nada Stefanie Breznik, BSc

Information Management to Support the Sustainable Implementation of Patient Blood Management

Requirement Analysis, Specification and Prototypical Implementation of a PBM Benchmarking Tool for Hospitals

MASTER THESIS

to achieve the university degree of

Diplom-Ingenieurin
Individual Master's degree programme:
IT-based Biomedical Engineering

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. Priv.-Doz. Günter Schreier, MSc
Institute of Neural Engineering

Graz, May 2016

In Cooperation with

AIT Austrian Institute of Technology GmbH

Digital Safety and Security Department

Reininghausstraße 13/1

8020 Graz

Austria

Supervisor

Dipl.-Ing. Peter Kastner, MBA

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master thesis dissertation.

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtliche und inhaltlich entnommene Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit/Diplomarbeit identisch.

Date/ Datum

Signature/ Unterschrift

Abstract

"*Transfusions are one of the most overused treatments in modern medicine, at a cost of billions of dollars* [1]". To place optimisation and preservation of the patient's own blood over the transfusion of donor blood are the basic principle for Patient Blood Management (PBM) [2]. To support sustainable implementation of PBM, benchmarking is an essential process, because its primary aim is to improve own performances and learn from others who have achieved high standards of excellence [3]. The PBM benchmarking process was designed under consultation of medical as well as PBM experts and based on existing benchmarking processes. The technical modules "data extraction", "data processing", "report generation" constitute the focus of this work. The PBM Analytics and Benchmarking Service (PBM-ABS) web application was implemented using AngularJS. An automatic data extraction method was established for the General Hospital of Vienna (Allgemeines Krankenhaus Wien – AKH Wien) as a showcase.

Kurzfassung

Transfusionen sind eine der am meisten übermäßig verwendeten Behandlungsarten in der modernen Medizin, mit verbundenen Kosten in Milliardenhöhe [1]. Das Grundprinzip von Patient Blood Management (PBM) ist es, die Optimierung und Erhaltung des Eigenblutes eines Patienten über die Transfusion von Spenderblut zu stellen [2]. Dabei ist Benchmarking ein essenzieller Prozess um die nachhaltige Umsetzung von PBM zu unterstützen. Das primäre Ziel dabei ist, seine eigenen Leistungen zu verbessern und von den besten seiner Konkurrenten zu lernen [3]. Der PBM-Benchmarking Prozess wurde unter Hinzuziehen medizinischer, wie auch PBM-Experten entwickelt und basiert auf bereits bestehenden Benchmarking Prozessen. Die technischen Module „Datenextraktion", „Datenverarbeitung" und „Report Generierung" bildeten den Fokus dieser Arbeit. Die PBM-Analyse und Benchmarking Service (PBM-ABS) Web-Applikation wurde mittels AngularJS implementiert. Als Showcase wurde eine automatische Datenextraktion aus bestehenden Datenquellen im AKH Wien durchgeführt.

Content

List of Tables

List of Figures

Abbreviations

ACID	Atomicity, Consistency, Isolation, Durability
AIT	Austrian Institute of Technology
AKIM	AKH Information Management
AKH Wien	General Hospital of Vienna
API	Application Programming Interface
CABG	Coronary Artery Bypass Surgery
CAR	Cardiac Surgery
Cardio-db	Database of the Cardiothoracic and Vascular Anaesthesiology Ward
CRM	Customer Relationship Management
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DOM	Document Object Model
EC	European Commission
EDC	Electronic Data Capture
EMR	Electronic Medical Record
Epo	Erythropoietin
EU	European Union
EU-PBM Project	Project for developing a European Guide on Good Practices for Patient Blood Management
FFP	Fresh frozen plasma
GCP	Good Clinical Practice
GUI/UI	(Graphical) User Interface
Hb	Haemoglobin
HIMSS	Health Care Information Management Systems
HIS	Hospital Information System
H-TEP	Hip-Total Endoprosthesis or THR (Total Hip Replacement)
HTTP/S	Hyper Text Transfer Protocol (Security)
ICCA	IntelliSpace Critical Care and Anaesthesia
ICU	Intensive Care Unit
IDE	Integrated Development Environment
JS	JavaScript
JSON	JavaScript Object Notation
KPI	Key Performance Indicator

MVC/W	Model-View-Controller or Model-View-Whatever
MVVM	Model-View-View-Model
Nadir	lowest point
NaN	Not a Number
ODBC	Open Database Connectivity
PAN	Pancreas Resection
PBM	Patient Blood Management
PBM-ABS	PBM-Analytics and Benchmarking Service
PDMS	Patient Data Management System
RBC	Red blood cells
SQL	Structured Query Language
SVG	Scalable Vector Graphic
TCP/IP	Transmission Control Protocol/ Internet Protocol
TDB	Transfusion Database
Web app	Web Application

1 Background

1.1 European Guide on Good Practices for Patient Blood Management (EU-PBM project)

"*Transfusions are one of the most overused treatments in modern medicine, at a cost of billions of dollars* [1]". One of the primary concerns to the European Commission (EC) is patient safety for which the safe and adequate use of substances derived from human blood plays an important role. Therefore, in 2013 the EC commissioned the AIT - Austrian Institute of Technology, which was supported by a panel of experts, to develop an „EU Guide for Member States on Good Practices for Patient Blood Management (PBM). This PBM Implementation Guide was used to start PBM pilot programmes at five participating teaching hospitals in different Member States of the European Union (EU). [4]

1.1.1 Concept of Patient Blood Management

To place optimisation and preservation of the patient's own blood over the transfusion of donor blood in order to improve patient safety and to reach optimal clinical outcomes constitutes the basic principle for PBM. For this purpose, the triad of anaemia, blood loss and transfusion (Figure 1), which are the three independent risk factors for an adverse outcome, should be addressed as early as possible in the course of treatment. [2]

Anaemia is characterised by a subnormal concentration of circulating red blood cells (RBCs) and is a prevalent trigger for transfusion, which was considered to be the optimal treatment for anaemia and/or blood loss for decades. Although RBC transfusions lead to corrected laboratory values but they do not treat the actual cause of anaemia, nor do they stop any bleeding. Additionally, several studies in various patient populations have shown a dose-response relationship between transfusion and nosocomial infections as well as an increased risk of rebleeding and respectively further blood loss. As a result, this triad constitutes a vicious circle which is illustrated in Figure 1. [2]

All clinical measures influencing these risk factors can be categorised into three classes called the three pillars of PBM (Figure 2). The first pillar represents mainly preoperative measures to optimise the patient's own red blood cell mass such as stimulating erythropoiesis or treating anaemia and iron deficiency. The second pillar includes any kind of

efforts to minimise diagnostic, interventional and surgical blood loss to preserve the patient's RBC mass. [2]

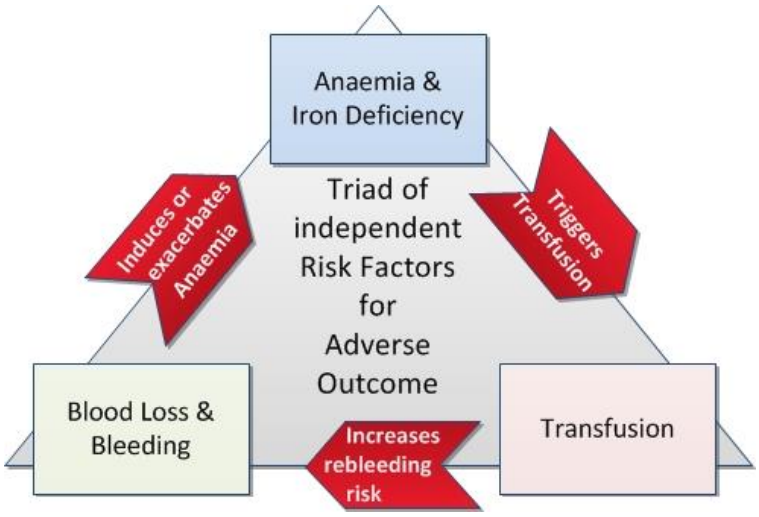


Figure 1: Triad of independent risk factors for adverse patient outcome [2]

Measures belonging to the third pillar are concerning the optimisation of the patient specific tolerance to anaemia by maximising the oxygen delivery while reducing the metabolic rate and adhere rigidly to physiological transfusion thresholds. Implementing the first two pillars are in most clinical scenarios sufficient to keep the haemoglobin values of the patient's majority above the predefined thresholds where transfusions are required. [2]

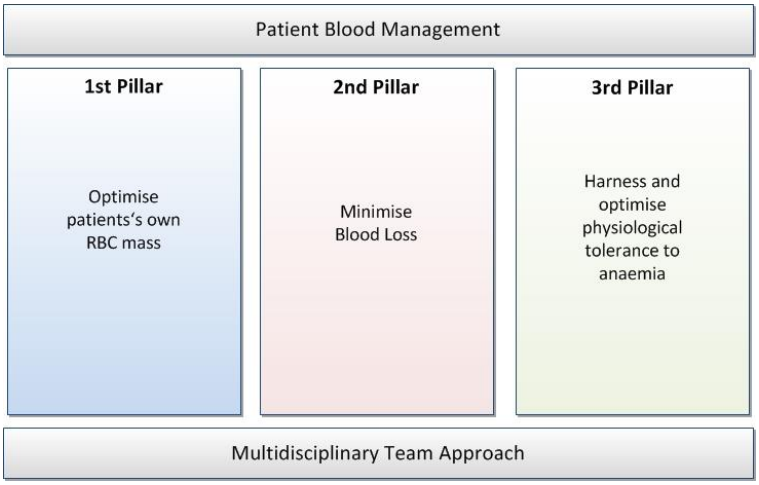


Figure 2: Three Pillars of Patient Blood Management [4]

As shown in Figure 2 the three pillars are built up on a multidisciplinary team approach which represents the basis of the PBM concept. This multidisciplinary team should involve for example physicians, nurses as well as quality managers and PBM experts. Team meetings

and audit programs for self-evaluation should improve the physician's skills and knowledge as well as undergoing critical self-assessment with active participation of medical experts. Resulting interventions of such assessment-cycles should enable the development of suitable implementation strategies and its implementation progress should be monitored over time. [2, 5, 3]
Accordingly, benchmarking turns out to be an indispensable process for these tasks.

1.1.2 Benchmarking

The term "Benchmarking" has its origin in the manufacturing industry in the late seventies when chief executive officer in Xerox Corporation, David T Kearns defined it as "*the process of measuring ourselves against the products, services, and practices of our toughest competitors*". The primary aim is to improve own performances and learn from others who have achieved high standards of excellence by comparing certain parameters. [3]
In the 1990s benchmarking was transferred to the health care sector with the distinct difference to the business world that patient outcomes and safety are replacing financial factors from being the key parameters of benchmarking concerns. [2, 3]

Basically, it has to be differentiated between internal and external benchmarking. Internal benchmarking is characterised by continuous measurement of one's own performances and compare and monitor the results reflecting the pre- and post-implementation situation of certain measures in regular intervals. The outcome comparison of institutional procedures and behavioural habits with best-performers in a typical group of organisations to learn about their latest methods and practices are established as external benchmarking. [6, 5]

In general an indication for benchmarking of specific health care services for comparable patients to achieve significant improvements is given when they [2]:

- have a high incidence and are affecting a large number of patients,
- have widespread variability in resource use and/ or outcome and
- are macro-economically relevant.

1.1.3 Benchmarking in Transfusion Medicine

In the field of transfusion medicine, all of the mentioned criteria indicating benchmarking to be valuable are fulfilled. With 85 million RBC units transfused worldwide to an estimated number of 25 million patients annually, the transfusion incidence rate can be considered as high. [2]
As can be seen in Figure 3 the variability in use of RBC units in European countries lies between 58,6 units/1000 population, in Greece 20,9 and in Romania [7]. Taking into account

that these results are concerning to countries with a similar high standard of care, also the variability can be stated as high.

In the Netherlands initial PBM measures were introduced in 2004, when the first national guideline “Blood Transfusion” including PBM was published. As a result, the number of RBC transfusions decreased around 26% until 2014. [8]

Consequently, the Netherlands turned out to be an appropriate benchmark for Figure 3. Assuming that with proper PBM efforts the other countries would be able to reach this benchmark and the reduction capacity for RBC transfusions amounts to over 4 million RBC units per year.

Due to the fact that the overall (direct and indirect) costs for a RBC unit lie between $522 and $1183 (mean: $761 ± $294, [9]) the macro-economic relevance cannot be ignored.

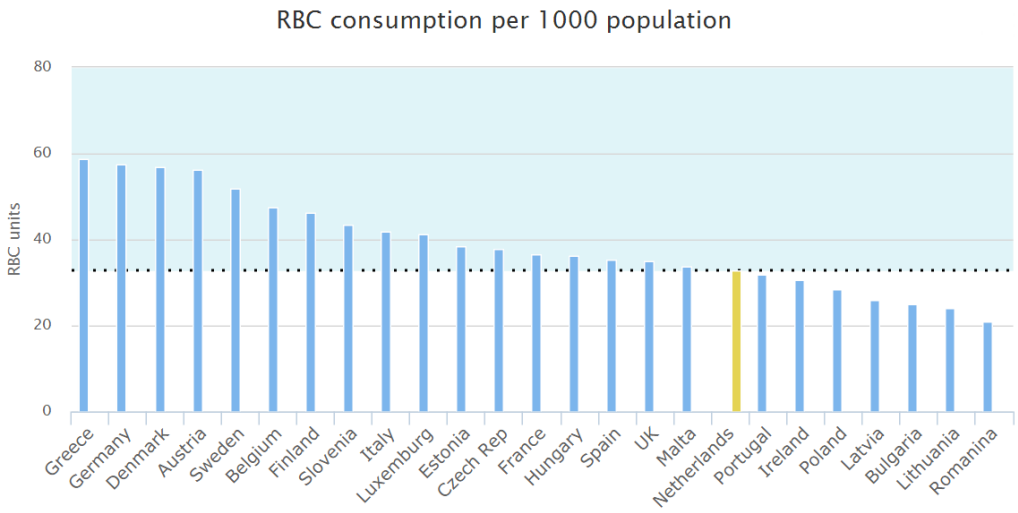


Figure 3: Number of red blood cell units transfused per 1000 inhabitants in European countries in 2010 [7]

Adapted for transfusion medicine the benchmarking process itself has to be additionally structured and collaborated with the aim of continuous quality improvement. Therefore, after implementation of new practices, re-evaluation of performances should be undertaken, preferably by continuous data collection. [3]

1.1.4 Existing Models for Benchmarking in Transfusion Medicine

Apelseth et al defined three possible models for benchmarking processes in transfusion medicine. The first model describes a national or regional benchmarking program using data from existing electronic sources linked by patient identification numbers, which is centrally coordinated. This model turns out to be the “gold standard” of benchmarking in transfusion medicine but implies many implementation challenges. The second model, called sentinel

site model, where data is collected from a limited number of institutions and a central coordinator provides the data analysis. In the third model an institution collects and analyses information and approach other institutions who they feel would be appropriate comparators. [3]

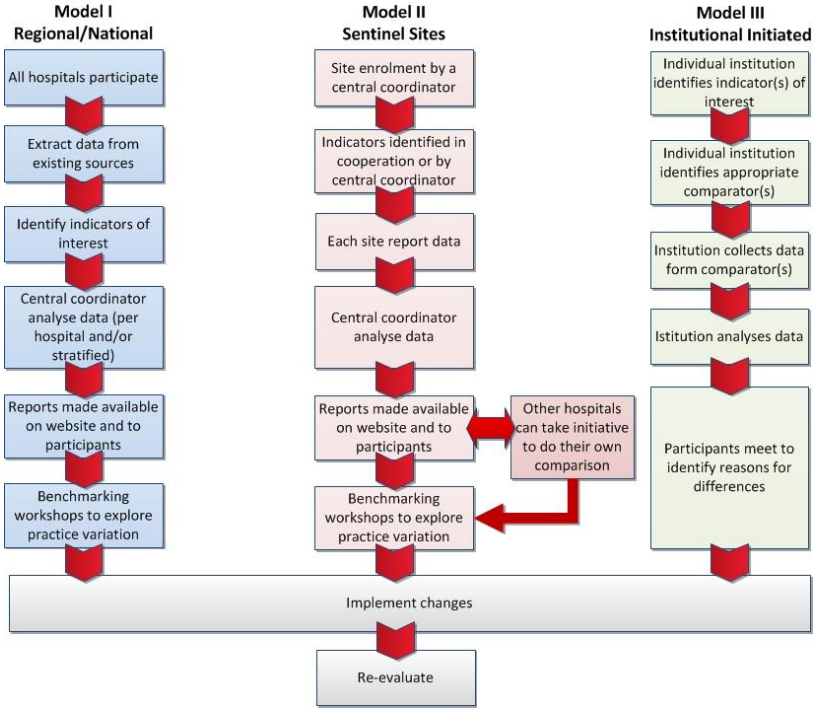


Figure 4: Three benchmarking models from Apelseth et al [3]

Another benchmarking program for transfusion medicine was introduced by Barty et al (Figure 5).

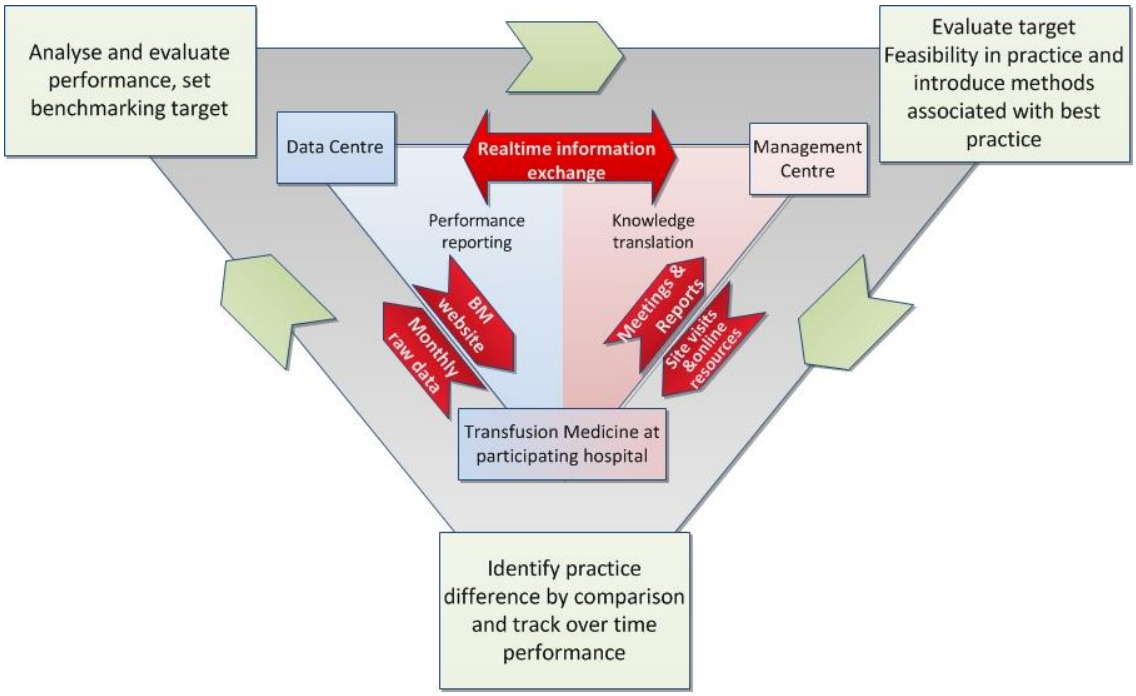


Figure 5: Conceptual benchmarking process from Barty et al [10]

It consists of two embedded active cycles to enhance continuous information flow and quality improvement. A crucial part of this concept is the data management system which provides a direct data flow form transfusion medicine programs at each participating hospital to the data centre for analysis, enabling high data accuracy. Furthermore a benchmarking website was introduced to hospitals giving feedback about their progress over time and comparing RBC outdating rates with those of other hospitals of similar size and geographic location. To raise awareness of certain issues and promote benchmarking targets annual meetings and site visits were accompanied by high-tech measures like webinars and other online resources. Certificates of recognition for achieved benchmarking targets were provided to increase motivation levels. [10]

The benchmarking process described by Kastner et al [5] already included specifically adjusted modules for PBM and was - with some modifications influenced by the previously described models - used for this work.

1.2 Used Benchmarking Process and Focus of this Thesis

1.2.1 Requirements for Information Management in Hospitals

"*Information management in hospitals is the sum of all management activities in a hospital that transpose the potential contribution of information processing to fulfil the strategic hospital goals into hospital's success.*" [11]

This definition of information management in hospitals of Winter et al is mainly targeted on a particular hospital information system (HIS). This leads to the term: "management of information systems" which implies the *planning* of a hospital information system constituting the basis, further *directing* the progress of its architecture and operation and *monitoring* its development with respect to the planned objectives. Beside technical information processing, also the aspect of humans as a part of information processing actions have to be considered. [11, 12]

This means, the goals of the PBM implementation and consequently the goals of the participating hospitals have to be taken into account within planning of the benchmarking tool with direct relation to the HIS.

The main task of a HIS is to provide the right information and knowledge concerning patients, at the right time and place, in the right format accessible for relevant personnel in making the correct decisions. Data collection, storage, processing, preparation and transmission are essential components and are playing an important role in a HIS and in the process of the EU-PBM project. [12]

Information management itself can be divided into three parts: strategic, tactical and operational information management. Strategic information management affects the hospital's information processing as a whole by describing the HIS's intended architecture or its systematic manipulation to fit into a strategic plan. It also has to deal with required resources like money, personnel, soft- and hardware, network architecture etc., depending on the specific requirements of every individual hospital. Tactical management relates to hospital functions like planning and documenting an operation while operational management is responsible for maintaining the HIS and its components. The range of operational management tools and methods comprises activities from intra-enterprise marketing of services to helpdesk and network management. All three are not just affecting machines and computers but also humans and their social behaviour, makes a HIS a sociotechnical subsystem of a hospital. [11]

Ashawi et al describes the required architectural process for customer relationship management (CRM) applications in health care organisations. In CRM patient data from internal (e.g. administrative, medical and pharmacy departments) and external (includes e.g. geographic, demographic and statistical data) data sources are required. [13]
The conditions for the described process show many parallels to those of the PBM benchmarking process, although using different data sources. Therefore, the four architectural process levels for CRM applications were also taken into account for designing the used process.
These four levels are [13]:

1. Source identification of patient data
2. Data quality matching and comparison phase
3. Data integration process
4. Final data quality checks (evaluation, monitoring, archival and distribution phase)

1.2.2 Description of the Used Process

Most of the modules constituting the used conceptual process (Figure 6) were adopted from Kastner et al [5], complemented with "data extraction" and "continuous improvement measures". Measures to improve own processes and performances are a crucial part of benchmarking and also integrated in the other described models of chapter 1.1.4. Therefore this module was also included in this process.

The "benchmarking management process" stands above all other modules. An internal team supported by an expert panel should undertake quality control and decide which setting of

internal and/or external benchmarking concerning procedures, time ranges for data capture and benchmarking indicators of interest are appropriate for their centre. Also providing required resources and communication are also their responsibility. [5]

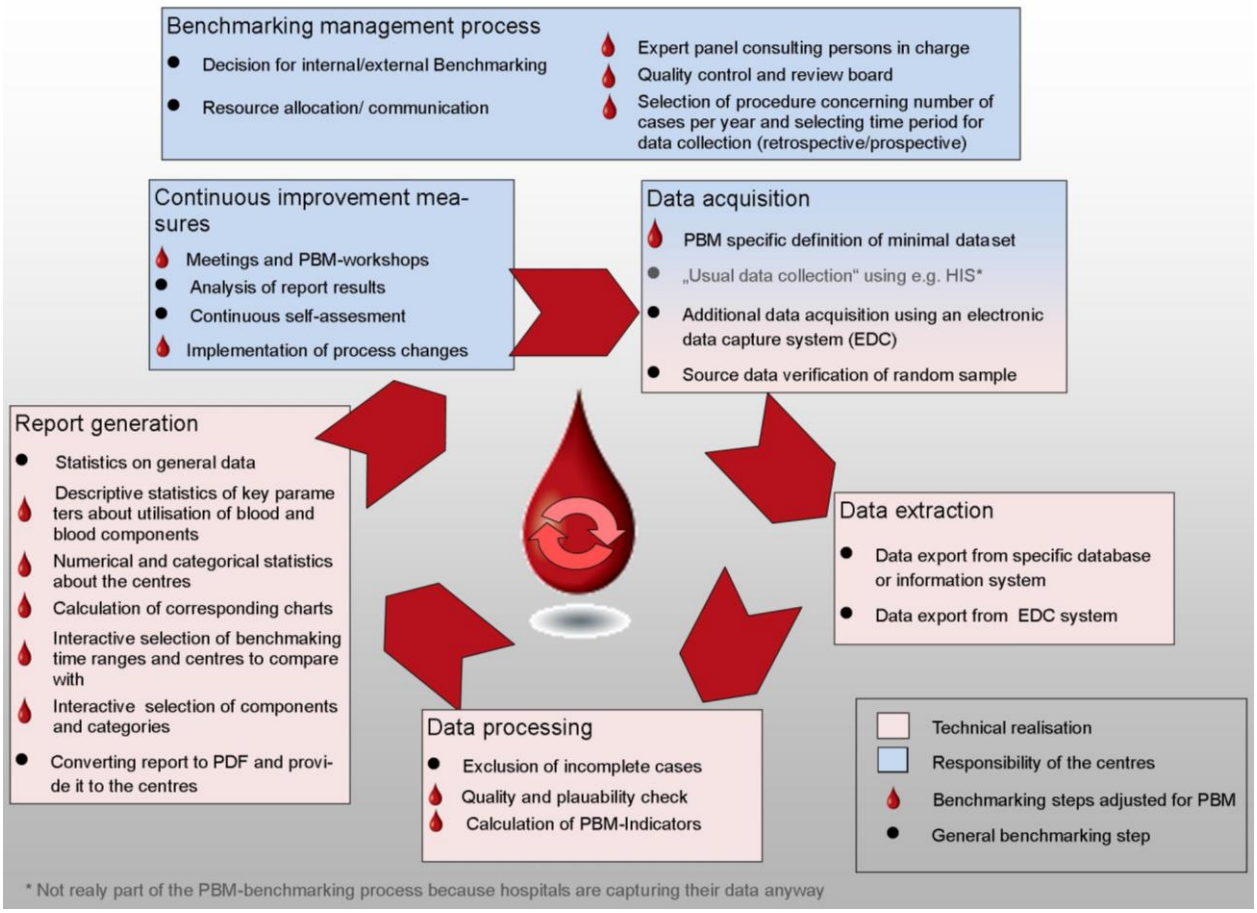


Figure 6: Used conceptual benchmarking process with PBM specific steps

The process was arranged as a cycle to illustrate the required continuity according to Apelseth et al. The whole process has been separated in two categories.
The first category represented by the blue modules contains all measures the centres supported by an expert panel are responsible for. The second category can be realised by technical means, which represent the focus of this work. For this reason the data extraction was outsourced from the “data acquisition module“ of [5] and will make up one of two big components in this work. The second part will be the implementation of a web application called “PBM-Analytics and Benchmarking Tool” (PBM-ABS tool) which encompasses the “data processing” and “report generation” modules.

The „data acquisition module" includes data collection from different sources. It belongs to both categories because of the definition of the minimal dataset. The minimal data set should

fulfil the main PBM requirements but has to be adjusted to the specific needs of the participating centres in coordination with the responsible team of PBM experts. In addition availability and accessibility of data items also influence the requirements for the technical realisation. For further information see section 2.2.1.

2 Methods

This chapter describes the use case scenarios of the PBM benchmarking process steps, which should be covered by the technical realisation of the software components. Derived from these scenarios the resulting requirements will be explained and the technological framework which has been chosen to fulfil them. Furthermore, the reasons for choosing these technologies will subsequently be given. The architectural design will also be described in detail, together with software tools, devices and technologies used for implementation. All following chapters will focus on the main benchmarking process steps (see red modules Figure 6), which are summed up under the term of PBM-Analytics and Benchmarking Service (PBM-ABS service).

2.1 Use Cases

By consultation with physicians form hospitals, which are participating at the EU-PBM project as well as international PBM experts a list of use cases were identified. The use cases are grouped in the following classes:

1. **Basis use cases**
   - 1.1. A PBM related core dataset should be defined which might be optionally extended by additional data items (supplementary dataset).
   - 1.2. Data from in-house electronic documentation systems or web based electronic data capture (EDC) systems should be used for benchmarking purposes to allow flexible utilisation. Data imports from EDC systems (comparable to clinical trials and online registries) should be supported in case that direct data export from HIS is not available. A dedicated EDC system was provided e.g. to the participating hospitals of the PBM Benchmarking trials in Austria [14].
   - 1.3. Only data which was undertaken automatic filtering regarding their plausibility should be stored.
   - 1.4. Data should be checked for completeness with respect to a predefined list of a core dataset.
   - 1.5. Benchmarking of data related to different predefined medical disciplines (e.g. coronary artery bypass graft, total hip/knee replacement) should be supported.
   - 1.6. No patient identification (name, address, date of birth) should be used for benchmarking to guarantee privacy of patients.
   - 1.7. A graphical user interface should guide the user through the different configuration steps.

2. Hospital specific use cases

2.1. Use case for analytics:

To analyse specific datasets, data should not be mandatorily stored outside the hospitals IT infrastructure, i.e. no upload to an external server should be required. For this case data should only be loaded into the client side application.

2.2. Use case for internal benchmarking:

For internal benchmarking data from previous uploads should be available to choose at least two different time periods for comparison. Therefore the user should be asked to commit the willingness to upload data from the own hospital to the benchmarking server. In this case the data should not be available for other hospitals.

2.3. Use cases for external benchmarking:

For external benchmarking, the user should be asked to commit the willingness to make the uploaded data from the own hospital available for other centres using the external benchmarking feature. Consequently the user should be able to select two other centres for comparison, which already committed to provide their data to other centres.

3. General use cases

3.1. It should be considered that the solution needs to be provided to hospitals in different European countries with heterogenic IT-infrastructures. The tool should require minimal PC infrastructure at the client side (PC, Internet Browser) without additional setup or installation.

3.2. For advanced utilisation of the PBM-ABS tool users should be able to upload and store centre specific data.

3.3. Access should be restricted to authorised users.

3.4. The tool should provide comparison of main key performance indicators (KPIs)

- Number of cases, period, age, medical indications
- Transfusion rate (TR) for RBC, FFP, factor concentrate
- Transfusion index (TI) for RBC
- Mean length of stay, complication rate
- Haemoglobin (Hb) before and after surgery

3.5. Results should be represented in tables and/or diagrams.

3.6. Printing or client side storing of a summary report should be enabled.

2.2 Requirements for the PBM-ABS Service

Based on the defined use cases the system requirements to support the benchmarking process are derived.

2.2.1 Definition of the Core Dataset

As previously mentioned, the used EDC system was already in use for the Austrian Benchmark Trial. The acquired parameters are shown in Table 1. From this dataset the most important parameters were selected according to [2] and some were added on recommendation of the PBM expert panel.

Table 1: Dataset collected by the AIT-EDC system

Basic data for each case:	
• ID • Name • Surgery date	• Phase (pre/post PBM) • Length of stay (days)
Surgery details	

• Intervetnion (3 letter code)
• Surgical technique (conventional/minimal invasive)
• Surgery duration (min)
• complications
	• estimated blood loss
	• reoperation (yes/no)
	• in case of cardiac surgery:
	  ○ Number of grafts
	  ○ Extracorporal circulation (ECC)
	  ○ Duration of ECC

Demographic data about each case	
• Gender (m/f) • Age (years) • Weight (kg)	• Size (m) • ASA-Score • Euroscore
Transfused blood components and blood products about each case	
• RBC ordered • Allogeneic RBC transfused (intra and post) • Autologous RBC transfused (intra and post)	• Fresh Frozen Plasma/ FFP (intra and post) • Platelet concentrates (intra and post) • Coagulation factor concentrates (intra and post)
Clinical data (pre / intra / postoperative) about each case	
• Platelet inhibitors (no/ ASS/ Plavix) • Thromboprophylaxis (no/ Iron/ Epo) • Type and Screen • Tranexamic aid	• Cell salvage • Unwashed shed blood • Type of anesthesia (general/regional/ other)
Laboratory Values	
• Preoperative haemoglobin value (g/dl) • Haemoglobin value of 3rd postoperative day (g/dl)	• Haemoglobin value of 5th postoperative day (g/dl) • Preoperative MCH • Preoperative MCV
Criteria for exclusion	
• Age < 18 years • Emergency surgery	• Coagulation dysfunction

Furthermore, it had to be considered that another approach of data acquisition beside the EDC system should be the retrospective extraction of relevant data from existing data sources of the AKH Wien. This dataset was divided into a core dataset, which is mandatory and contains only parameters available in existing data sources of the AKH Wien and a supplementary dataset. The core dataset in this case is the minimal dataset and the core and

supplementary datasets together are optimal. Analytics and Benchmarking features for parameters of the supplementary dataset should be provided but they have to decide whether they want to integrate those parameters to their benchmarking strategy or not.

Table 2 shows the whole dataset consisting of the core- and the supplementary dataset.

Table 2: Dataset for PBM-ABS tool

Core-Dataset	Supplementary Dataset

• ID
• Surgery date
• Length of stay (days)
• Intervention (3 letter code)
• Gender (m/f)
• Age (years)
• Weight (kg)
• Size (m)
• Allogeneic RBC transfused (number of units total)
• FFP (units total)
• Platelet concentrates (yes/no total)
• Coagulation factor concentrates (yes/no total)
• Preoperative haemoglobin value (g/dl)
• Haemoglobin value of 5th postoperative day (g/dl)
• Minimal haemoglobin value/ hb_nadir (g/dl)
	• Surgical technique (conventional/minimal invasive)
	• reoperation (yes/no)
	• Complications
	• Surgery duration (min)
	• in case of cardiac surgery:
	  ○ Extracorporal circulation (ECC)
	  ○ Duration of ECC
	• ASA-Score
	• RBC ordered (number of units)
	• Allogeneic RBC transfused (number of units intra)
	• FFP (units intra)
	• Platelet inhibitors (no/ ASS/ Plavix)
	• Tranexamic aid
	• Cell salvage
	• Type of anesthesia (general/regional/ other)
	• Haemoglobin value of 3rd postoperative day (g/dl)
	• Haemoglobin value at end of surgery (g/dl)
	• Haemoglobin value at discharge (g/dl)
	• Emergency surgery
	• Anaemia-treatment

For inter- and intra-institutional comparison the main key performance indicators are transfusion index and transfusion rate, which are defined by Gombotz et al as follows [2]:

- transfusion rate: *"proportion of transfused patients within a defined patient population*"
- transfusion index: *"average number of blood components administered per patient of a defined patient population*"

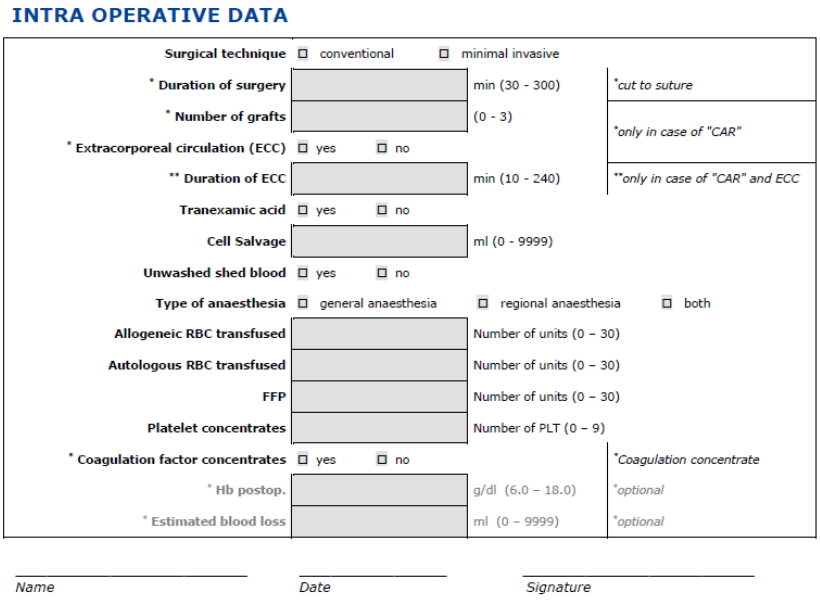
The defined core dataset provides all required parameters for calculating transfusion index and rate as well as perioperative blood loss according to the Mercuriali algorithm, which is another important parameter in PBM. [2]

2.2.2 Data Acquisition and Extraction (EDC System vs. Internal Sources)

2.2.2.1 EDC System

An EDC system (electronic data capture) is a convenient data collection tool because these systems are used for clinical trials, including quality management tools (audit trail, source data verification), as they are required by good clinical practise (GCP) guidelines. [5]

In the EU-PBM project four of the five centres are using such an EDC system for data collection. For data extraction this means that if a special EDC system is used, it depends on which data structures are provided for exports, if further steps of data processing are required. The EDC system used for the EU-PBM project was provided by the Austrian Institute of Technology, based on the EDC system which was already used for the Austrian Benchmark Trial in 2009 [15]. The parameters that are captured by using this system can be seen in Table 1, the form for the acquisition of intraoperative data is illustrated in Figure 7.



INTRA OPERATIVE DATA

Surgical technique	☐ conventional ☐ minimal invasive		
* **Duration of surgery**		min (30 - 300)	*cut to suture
* **Number of grafts**		(0 - 3)	*only in case of "CAR"
* **Extracorporeal circulation (ECC)**	☐ yes ☐ no		
** **Duration of ECC**		min (10 - 240)	**only in case of "CAR" and ECC
Tranexamic acid	☐ yes ☐ no		
Cell Salvage		ml (0 - 9999)	
Unwashed shed blood	☐ yes ☐ no		
Type of anaesthesia	☐ general anaesthesia ☐ regional anaesthesia ☐ both		
Allogeneic RBC transfused		Number of units (0 – 30)	
Autologous RBC transfused		Number of units (0 – 30)	
FFP		Number of units (0 – 30)	
Platelet concentrates		Number of PLT (0 – 9)	
* **Coagulation factor concentrates**	☐ yes ☐ no		*Coagulation concentrate
* **Hb postop.**		g/dl (6.0 – 18.0)	*optional
* **Estimated blood loss**		ml (0 – 9999)	*optional

Name *Date* *Signature*

Figure 7: Acquisition form for intraoperative PBM data

Extracting data in an automated way from existing data sources would decrease time exposure of clinicians and increase data accuracy and reliability.

2.2.2.2 Data Extraction for Centres Using the EDC System

The EDC system used by four of the participating centres is a web application developed by AIT. For each case, the data entered by the centres for the parameters listed in Table 1, is stored in a PostgreSQL® (PostgreSQL Global Development Group, [16]) database. Zope 2.13® (Zope Foundation, Richardson, USA, [17]), a Python™-based (Python Software Foundation, Beaverton, USA, [18]) framework for building secure web applications [19], was used for the application server. This EDC system already provides the required data extraction facilities for using the PBM-ABS tool.

2.2.2.3 Internal Data Sources

In general, if hospitals are equipped with a state-of-the-art and well-established HIS and/or other data sources the required data can be exported and - if more than one data source is used - joined by using patient identifiers, similar to the first model of Apelseth et al [5, 3]. To support the capability of deciding whether the own centre satisfies these requirements the Health Care Information and Management Systems company (HIMSS) provides a 7 stage model, which according to HIMSS "*identifies the levels of electronic medical record (EMR) capabilities ranging from limited ancillary department systems through a paperless EMR environment*" [20].

For the showcase in the AKH Wien the PBM-required data, according to a predefined core dataset (see next section) can be found in three different data sources:

- the hospital information system (HIS),
- the patient data management system (PDMS, IntelliSpace Critical Care & Anesthesia information system, ICCA Philips Healthcare, USA, [21]) of the intensive care unit (ICU)
- and the transfusion database (TDB).

The task is, to find an automated way with a high usability for physicians in this clinical setting to extract and merge data of these sources to get a file containing all parameters defined in the core dataset for specific interventions and time ranges.

The HIS of the AKH Wien is called AKIM (Cerner Corporation, [22]), which stands for AKH information management. It is a project of the city of Vienna and the medical University of Vienna, not from the AKH itself, which was one of the reasons for the restricted system access. It should support routine processes of the hospital as well as scientific research. AKIM is the sum of software- and EDV systems of former SIEMENS Health Services and since 2015 of the Cerner SOARIAN-family.
For routine processes, e.g. the documentation of out-patient and in-patient cases, the main feature is the so called "AKIM Viewer", in use since 2010, giving an overview over patient data, documents, diagnoses, services and laboratory results. The HIS contains all required features for patient administration and cost accounting. [23]
The AKIM research documentation and analysis platform (AKIM RDA-Platform) is an IT-tool of the medical University of Vienna and enables individual database construction and automated import of the routine patient data. It is integrated in the HIS and provides a central data pool for hospital routine and science. [24]
For extracting the EU-PBM core dataset only a small extract of the captured patient data was available in form of an excel export.

The PDMS system is an advanced tool used for clinical decision support and documentation in the ICU of the AKH. This encompasses patient data, admissions documents, vital signs, laboratory results and consultation notes. To enter a procedure or diagnosis, which can be displayed beside infusion orders or medications on patient's charts and nurses' worklist, it is possible to enter free text or select from standard or internal coding catalogues. Data analytics are not only limited to reports by patient, also reports by time, person, disease, process or a combination is possible. [25]
This flexibility can only be provided by a complex data structure, which had to be analysed in detail to query the required data.

2.2.2.4 Data Extraction from Internal Data Sources

From use case 1.1, 1.2 and 1.6 (section 2.1) results the requirement of an automatic data extraction when using in-house electronic documentation systems, because the only parameters of the predefined core or supplementary dataset should be contained in the export. A possible realisation of this automatic data extraction was shown in this thesis on the example of the AKH Wien.

The task was to collect the predefined core dataset for using the PBM-ABS app out of existing data sources of the AKH. It should give an example how such a data extraction can be done, but because of high differences in used hospital information systems and other required data sources has to be individually done for every hospital. First it was intended to merge data from the HIS (AKIM), the PDMS of the ICU (ICCA) and the TDB. Concerning these data sources, direct access to the PDMS was granted, for the other two data sources, only excel exports of corresponding systems were provided. Even if the excel export contained all required data concerning transfusions for the core dataset, no further detailed information about the TDB on the data capturing procedure or system was available.
Since the required data quality could not be provided by extracting the data form the mentioned sources a fourth data source, the cardio database (Cardio-db), was added for the extraction of the EU-PBM core dataset. The Cardio-db is a Microsoft® Access (Microsoft Corporation, Redmond, USA, [26]) database established by the IT-department of the Cardiothoracic and Vascular Anaesthesiology of the AKH. From this data source an excel export was provided.

Concerning the direct access to the PDMS it has to be mentioned that for quality management of the AKH there had already been the demand for individual data queries,

which was realised by Philips via accessing the data out of Microsoft® Access. This access was enabled by Open Database Connectivity (ODBC).
Because of this existing process- which will not require further user training or other unreasonable expense in its implementation- the decision was made to perform the EU-PBM data extraction as well by using Microsoft® Access.
For constructing queries to extract the required data from the PDMS and for combining the data of all four sources the SQL-view (structured query language) of Microsoft® Access was used because it enables a higher query complexity. This means, that all queries were constructed by using SQL, which is explained in more detail in chapter 2.5.

2.2.3 Data Processing and Report Generation

For the "data processing" and "report generation" steps a reporting tool should be developed, where the exported file containing at least parameters of the PBM-core dataset will be imported by the clinicians and all relevant analytics- and benchmarking-statistics will be calculated automatically and exported in the form of a PDF report. This tool is called PBM-ABS tool. Regarding data processing the tool should perform a set of quality checks.

First it has to be checked if the parameters meet the specification requirements concerning parameter name and type. All cases, not compliant with the criteria of the predefined core dataset, have to be excluded. Each parameter has to be checked for its physiological plausibility and cases containing outliers (i.e. data outside normal physiological, clinical or other reference values) have to be removed. Table 3 shows the parameters and the corresponding filter conditions concerning type and plausibility thresholds which are checked by the PBM-ABS tool.

Additionally, the tool should give a feedback, how many rows of the loaded dataset are valid and how many do not satisfy the given specification.
After processing key-performance indicators like transfusion index, transfusion rate or estimated blood loss have to be calculated. [5]

To cover use case 2.2 and 2.3 (section 2.1) the tool should be connected to a PBM database where the physicians have to upload data of their centre to use all features of the tool. If they are not willing to upload their data, only analytics of recent imported data should be available. Possibility should be given to upload the data for internal benchmarking without making it available to other centres. In this case, no comparison to other centres should be available but comparing oneself with results of former time ranges.

Table 3: Parameters and corresponding filter conditions concerning type and plausibility thresholds

Core-Dataset		**Supplementary Dataset**	
Parameter	**Filter conditions**	**Parameter**	**Filter conditions**
ID	-	Surgical technique (conventional/minimal invasive)	CONV, MNINV
Surgery date	>2011	reoperation (yes/no)	Yes, No
Length of stay (days)	1-99	Complications	Yes, No
Intervention (3 letter code)	REC, CAR, CABG , LIV , PAN,TEP, ECV	Surgery duration (min)	30-300
Gender (m/f)	m, f, male, female	**in case of cardiac surgery:**	
Age (years)	18-110	Extracorporal circulation (ECC)	Yes, No
Weight (kg)	51-180	Duration of ECC	30-300
Size (m)	100-220	ASA-Score	1-3
Allogeneic RBC transfused (number of units total)	0-30	RBC ordered (number of units)	0-9
FFP (units total)	0-30	Allogeneic RBC transfused (number of units intra)	0-30
Platelet concentrates (yes/no total)	Yes, No	FFP (units intra)	0-30
Coagulation factor concentrates (yes/no total)	Yes, no	Platelet inhibitors (no/ ASS/ Plavix)	0-30
Preoperative haemoglobin value (g/dl)	6-18	Tranexamic aid	Yes, No
Haemoglobin value of 5th postoperative day (g/dl)	6-18	Cell salvage	Yes, No
Minimal haemoglobin value/ hb_nadir (g/dl)	6-8	Type of anesthesia (general/regional/ other)	GA, RA, OTH
		Haemoglobin value of 3rd postoperative day (g/dl)	6-18
		Haemoglobin value at end of surgery (g/dl)	6-18
		Haemoglobin value at discharge (g/dl)	6-18
		Emergency surgery	Yes, No
		Anaemia-treatment	Yes, No

2.3 Architectural Design

Figure 8 illustrates the architectural design of the whole PBM-ABS service according to the requirements in 2.2. In the beginning stands the definition of the minimal and the optimal dataset as described in 2.2.1. In the data acquisition part the path is dividing in two branches. The fist branch includes data acquisition with existing data capturing and information management tools of the hospital. The following step in the data extraction module is the automatic data extraction from these sources. The second branch illustrates data acquisition using an EDC system, which requires an export depending on the technology of the EDC system as data extraction step. In the EU-PBM service this could be done with an SQL-

query. Having the resulting file of these branches, which should meet the predefined specifications and is shown as data export in Figure 8 the branches are merging into one branch again.

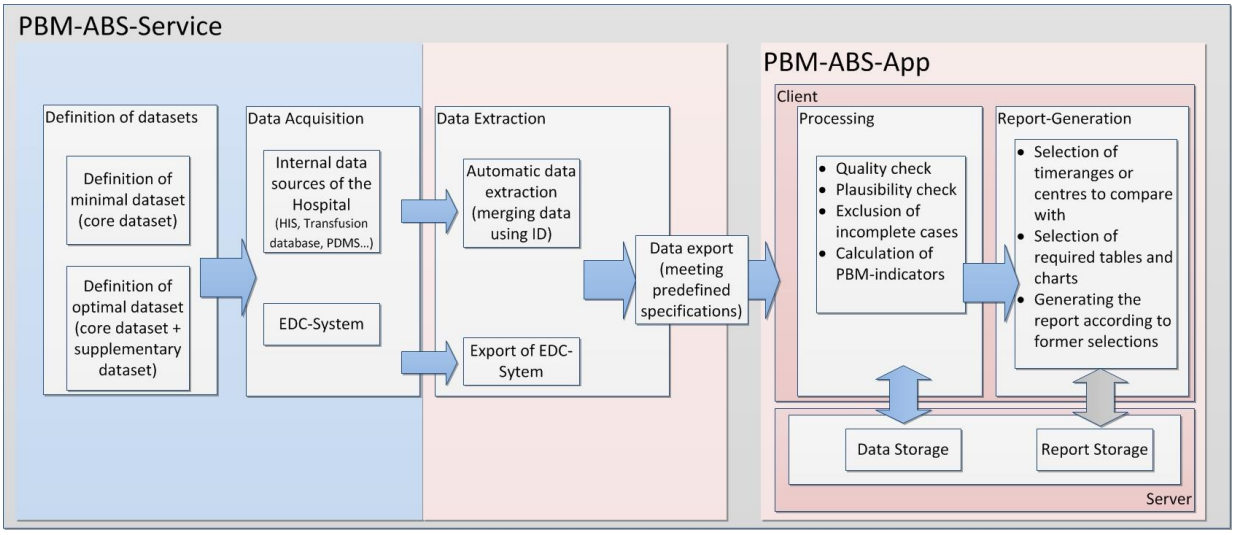


Figure 8: Architectural design of the PBM-ABS service

On this point the PBM-ABS app is entered. The data export is loaded into the client side app and is undertaken some processing steps, listed in the processing module of Figure 8. After data exchange between client and server of already stored and currently loaded data, depending on the users configurations, all calculations required for the report are made and the report is generated. The grey exchange arrow illustrates the storage of the generated report. This step will not be realised in this work but should be considered for the future. Storing the reports might have benefits for documentation and quality management purposes. For more details about the architectural designs of the single parts of the PBM-ABS service see section 3.2.

2.3.1 Realisation of the PBM-ABS Tool as Web Application

Based on the use cases and main requirements, the implementation of the PBM-ABS tool as a JavaScript® (JS, JavaScript.com, Orlando, USA, [27]) web application (web app), was decided. For this tool the term PBM-ABS app is used in this work. There are some disadvantages of web apps compared to desktop applications like:

- web apps rely on internet accessibility,
- web apps sometimes have reduced performance,
- execution of JS is dependent on the browser and
- the code is visible for the user.

However, web applications have far more advantages. It is easier to make a web app user friendly for multiple platforms and screen sizes, world wide access is provided and it requires no further installation than having a standard web browser available. Because everyone is

accessing the same web application provided by the developer, the application is always up to date and no user storage is required. A secure log-in can easily be provided which also reduces the problem that the code is visible for everyone, because only code of content displayed to the user can become visible. Consequently people without access to the application will have no access to the code. This can also be beneficial because it provides transparency, often required for quality management purposes.

Additionally, there are minifying programs which are making the code-size smaller less readable. [28] As integrated development environment (IDE) Visual Studio® 2015 Community edition (Microsoft, Redmond, USA, [29]) was used, which is "*an extensible IDE for creating modern applications for Windows, Android an IOS as well as web applications and cloud services"* [30] and is open source.
Most webpages are written in HTML®, with JS elements to handle the interaction between the interface and the back-end service and Cascading Style Sheet (CSS®) elements. [31]
JS is a programming language that can be interpreted by the browser, which means the browser has a built-in interpreter that can parse and execute the language. This provides direct access to browser events and the Document Object Model (DOM) objects and accordingly the possibility to add, modify and remove web page elements without reloading it. [31]

There are many libraries extending JS and one of the most common is JQuery (JQuery Foundation, Redwood, USA, [32]), which provides quick access to specific elements using selectors and built-in functions. Not just access to single DOM elements but also to groups such as paragraphs belonging to the same class or containing a certain substring in their ids. This access and the built-in functions enable easy manipulation of these DOM elements in many ways. [31]
Because of the high complexity- caused by many configuration options and dynamic content the PBM-ABS app should provide - using plain JS and JQuery would be possible but not very efficient.

Therefore AngularJS (Google, Mountain View, USA, [33]), a JS MVC framework developed by Google®, was used. Because of its architecture it is possible to make the implementation more structured and well-designed. AngularJS enables handling user input in the browser, manipulating data on the client side and controlling all elements visible to the user. [31]

There are many further advantages of AngularJS of which some contributed to the decision to use it for the PBM-ABS app [31, 34]:

- **Data Binding:**
  As mentioned in the beginning of this section JQuery for example can update parts of the DOM without reloading the whole page. In AngularJS this is much easier and more dynamical. Data bindings map parts of the UI to JS properties and are updated automatically. This works in both ways: changes in the UI will change the JS properties and vice versa.
- **Extensibility:**
  Beside many features provided by AngularJS almost every aspect, like filters or directives can also be realised by custom implementations.
- **Clearness:**
  The developer is forced by Angular to write clean and logical code.
- **Reusable Code:**
  The combination of extensibility and clearness of the code usually results in reusable custom services which can easily be incorporated in other software. How this can be facilitated can be seen in section 2.5.2.
- **Support:**
  Google is providing a lot of documentation and there is a large community using AngularJS, which is very helpful for any kind of problem during the implementation process.
- **Compatibility:**
  On one hand this means the compatibility with existing code: Because AngularJS is based on JS and is also related with JQuery, existing code can also be integrated. On the other hand high browser compatibility is given. The most popular browsers are Chrome® (Google, Moutain View, USA, [35]), Firefox® (Mozilla Foundation, USA, [36]and Internet Explorer® (Microsoft, Redmond, USA, [29]) and even in these three browsers some differences exist in compiling JS code. This is because the browser uses an engine to parse data from server, build objects etc. But each browser is using a different engine and accordingly interprets the scripts differently. When using AngularJS, it takes care of most of these issues.

2.3.2 Webserver

Compared to former websites with primarily static content, nowadays the websites have become much more dynamical enabled by the aforementioned technologies. Direct interaction with the user from a client side application as well as with back-end web services on the web server is facilitated by JS, JQuery and AngularJS. This communication between web server and browser consists basically of a request, sent by the browser and a response from the server which is further displayed to the user. Accordingly, the web server constitutes

a very critical component, checking the format and validity of requests to provide the required security. The PBM-ABS app is a combination of client side and server side scripting. The App itself is client side scripting, which means the code is sent to the client where it is processed. There is no processing on the server. This makes it easy to provide the application to a large number of people. The web services are processed on the server, so they are the server side component. Therefore choosing the right development platform for the webserver is essential.

For the PBM-ABS app Node.js (Node.js Foundation, USA, [37]) with express was chosen as a webserver. Node.js is a JS platform built on V8 engine of Google Chrome, which enables it to run JS applications outside a browser. It easily enables adding and managing external modules like the express module which is a framework that provides the actual web server. This provided webserver is easy to implement, has a robust feature set for e.g. parsing requests or error handling. Node.js itself is easy to install and set-up and it enables JS snippet testing without using a web browser. A plug-in for Visual Studio® is available which makes it simple to create Node.js projects. Because it's based on JS it is unnecessary to understand a back-end scripting language like Python or PHP. [31]
For data storage of the PBM-ABS app a MySQL(Oracle Corporation, Redwood, USA, [38]) database was used. For further information see section 2.3.3.

Another reason for choosing both technologies, MySQL and Node.js, was that there is an existing MySQL driver for Node.js, written in JS. This driver already provides the required features for creating the web service which handles the data exchange between application and database.

2.3.3 Data Storage

As mentioned above, MySQL® was chosen for storing the PBM data. But before choosing a relational database, it had to be decided if a relational or a so called "No-SQL"-database should be used. "No-SQL" does not mean that it is not SQL, it means that it is Not-Only-SQL. The main difference between those database types is, that "No-SQL" databases offer horizontal scaling, which means that data is not stored in tables or rows but in documents and one document can be embedded into another.
Furthermore, relational or SQL databases provide ACID transactional properties (Atomicity, Consistency, Isolation, and Durability), while "No-SQL" databases don't. [39]
Even if "No-SQL" databases offer many features to increase scalability and to achieve higher performances in certain circumstances (see [39]) the data storage of the PBM-ABS app just requires the conventional tabular relation data principle of having one entry per patient and

every entry has the same parameters. Therefore, no higher scalability is required, while ACID properties are quite important, so there is no need for choosing a “No-SQL” database. Generally the more features a software or database provides, the more complex its usage and maintenance becomes.

2.4 Verification of Software Components and Database Transactions

After implementation, the developed software components and database transactions have to be tested for their correctness.

Concerning the data extraction it would require too much effort to manually check every single entry of the results. Therefore, 30 random samples were checked for their correctness. This includes the inspection of correctly matched patient entries as well as checking the results of aggregation functions, like how many transfusions a patient received.

The only possibility for verifying the PDMS query was to check against the data of the Cardio-db. This was the case, because access was limited to database transactions, which means that there is no possibility to use features of the PDMS GUI itself. The only parameters that were appropriate for this comparison were gender, age and height.

Concerning the PBM-ABS app there are two different parts to check. For checking the data processing, a test-loading file was generated, containing failures of notation- and type-correctness, missing values as well as exceeding or falling below plausibility thresholds. All fields of these test-loading files were checked manually.

The main part was to verify the benchmarking indicators, analytics and statistics which are calculated by the PBM-ABS app and included in the generated report. Therefore the Microsoft® Access queries used for the required calculation of the EU-PBM-D6-Evaluation-Report were used to calculate the same indicators and statistics that would be provided by the generated report. The original dataset of the EU-PBM project was uploaded on the PBM database and the report was generated for every participating centre. Afterwards, the results of both technologies were compared. Verification via checking results against other results of the same kind but produced by different tools also meets the requirements of the AIT process definition for sustainability tests in the accredited field.

2.5 Used Software Tools, Devices and Technologies

2.5.1 Technologies for Automatic Data Extraction

2.5.1.1 ODBC

The Open Database Connectivity framework is an interface language, which provides Application Programming Interfaces (APIs) to enable application development without tying it to a specific database server or operating system. This means that beside developing an application without knowing which database server is going to be used, also replacing the

original database server without modifying the source code is possible. For this purpose all database interactions are handled by the API and the corresponding ODBC calls. The module translating queries from the application for the database server and the results the other way round is the ODBC driver. [40]
So the application, in this case Microsoft® Access, connects to the ODBC Driver Manager which uses the required driver e.g. Microsoft SQL ODBC driver to connect to the database server. [41]

2.5.1.2 Microsoft® Access

Access offers the WYSIWYG system (What you see is what you get) which facilitates dealing with very complex data queries over a graphical user interface. Tables in Access are created with data definition language (DDL) in a database management system (DBMS). DDL is usually part of the structured query language (SQL), which is - with some deviations from the original SQL standard - used in background to perform the actual database operations. The user has the possibility to create queries over the graphical user interface and the SQL-statement is created automatically in background or the user can directly use the SQL-view where he or she can construct the SQL query manually. Alternatively Visual Basic can also be used over another user interface. The tables and query results which are visible for the user are called the frontend database. All information about the frontend database including notations, links and the data itself are stored in the backend database. [42]

2.5.1.3 Structured Query Language (SQL)

SQL is a database manipulation language and basically there are two ways to issue an SQL command to a database:

- To send a manually typed single command to the database and receiving the result in form of a virtual table stored in main memory, or
- using embedded SQL, like in this case, in an application program with a form based or command line based interface for query construction.

The syntaxes of both kinds of SQL are very similar, but the embedded SQL does not require conforming to all standards. Nevertheless, all SQL commands are built up on the same elements: keywords, tables, columns and functions. Tables and columns mean the name of the tables and columns on which the command should operate. Keywords and tables are mandatory, columns and functions are functional. [43]

The main keywords for queries are SELECT and FROM. Between those keywords all attributes have to be inserted which should be selected from a specific table defined after the

FROM notation. As can be seen, SQL is not asking how to search for data but which data it has to search for. Using the keywords UNION, INTERSECT or EXCEPT, different tables or data sources can be combined and with GROUP-BY can be sorted alphabetically or numerically according to one or more defined columns. It is possible to add filter conditions to the query or group entries with the same attributes. On attributes, which are equal, aggregation functions can be applied returning for example average, standard deviation, minimum or maximum of a group. Also arithmetic operators like summation, subtraction, multiplication and division are applicable. [44]
These main functionalities facilitated the extraction of the relevant parameters from the PDMS databases.

2.5.2 Description of AngularJS for PBM-ABS App Development

As mentioned in 2.3.1, AngularJS - a JS MVC framework developed by Google®, which also includes a reduced JQuery library called JQuery lite - was used for developing the PBM-ABS app. [31]
An MVC structure, or Model View Controller structure, means that there is a clear separation of code dealing with the data representing the current state of the application (model), the application logic, which manages the relationship between view and model (controller) and the graphic illustration of the data presented to the user (view). [34]
Beside the definition of AngularJS as an MVC framework many publications are using the term of an MVW (model-view-whatever) framework which are quite similar. Figure 9 gives an illustration how AngularJS fits in two of these models.

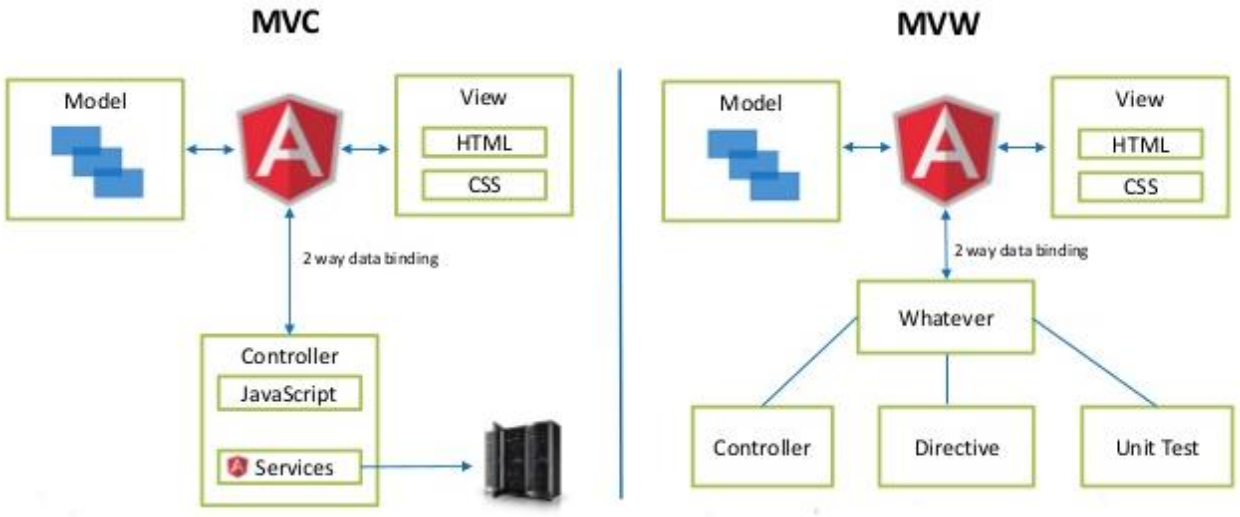


Figure 9: AngularJS interpreted as MVC or MVW framework. **[45]**

2.5.2.1 Modules

Modules are a kind of container for related JS code. Each module can contain its own controllers, services, factories and directives. With the declaration of a module AngularJS checks whether all defined parts of this particular module are available. Using the ng-app directive, the module can be defined, which should be loaded as main entry point for the application. [46]

2.5.2.2 Controllers

A controller is almost always linked to a corresponding view or HTML. As described in the MVC structure in 2.5.2 it is like a gateway between view and module and accordingly performs the main UI-oriented operations. The controller requests the required data for the current UI from the server and makes decisions which part of it is further shown to the user and its presentation. It also handles user interaction like data input or clicking events.
Each controller, as well as other elements – e.g. directives - has a scope, which constitutes its context and handles the access permissions of DOM elements in terms of functions, variables etc. [46]

2.5.2.3 Service, Factory and Provider

Services are functions or objects which can be accessed by every component of the application and are holding states or behaviours. One common AngularJS service is the $location service. (the $ sign is the prefix of all services) It enables interaction with the URL in the browser bar via manipulating it or just gaining information from it.
Another very important service is the $http service. [46]
HTTP stands for Hypertext Transfer Protocol, which defines the data exchange between browser and webs server. HTTPS adds an additional security layer in form of a certificate provided to the browser, which is required that the server will respond to any request. The most important requests are the GET-request, which retrieves information from the server and the POST-request that sends data to the server. [31]
Those requests are also provided by the $http service as well as setting the headers, caching and dealing with server responses and failures. [46]

Beside those built-in services it is possible to create own AngularJS services as well. One way to define such a services is the factory function, which is recommended for a functional programming style and if the developer prefers to use functions and objects. The service function is also a possibility for creating a service but instead of invoking the function and storing the return value, AngularJS will always call “new” on the function to create an instance of a class. Defining a service using the provider function is only recommended if configurations are required before loading the application. [46]

2.5.2.4 Directives

Directives are dealing with DOM manipulation and provide a way to create reusable UI widgets. DOM manipulation means, that they can change the behaviour of existing HTML snippets and reusable UI widgets mean that the directive creates a new HTML structure and has its own rendering logic. Every directive has its own scope and a link function, which is performed on the scope, when the directive gets called. When data is passed to a directive, this is always by reference. This can be realised by a two-way-binding, where changes performed by the directive are also performed in the scope of the corresponding controller or a one-way-binding where only changes of scope variables are passed to the directives but not the other way round. [46]

2.5.2.5 Filters

Filters are changing the format or the values of the user interface. They can be applied directly from the HTML or inside a controller or other services. An example of a predefined filter is the “number” filter which is rounding the digits after the decimal point or adding a thousands separator. [46]

2.5.2.6 Dependency Injection

Dependency injection constitutes the bases for the whole service concept in AngularJS. Every service can be “injected” into another service, controller or directive defining it as a dependency. AngularJS is then checking if all dependencies a service needs are available and fully instantiated before the service is executed. [46]

2.5.2.7 Routing

As mentioned before it is possible to have multiple controllers with their corresponding views, but the question is, how to switch from one to the other. This is enabled by the $routeProvider.when function. The first argument of this function is an URL for which the appropriate configuration will be triggered. This configuration tells the service, which controller is needed and which view should be shown by the ng-view directive. Usually the ng-view directive is integrated in an index-html - which is the “main HTML” and is always shown - and handles the content which is displayed in a certain step of navigating through the application. [46]

2.5.3 Additional technologies and JS Libraries used for the Realisation of the PBM-ABS App

2.5.3.1 MySQL® database

MySQL® is an open source project for a rationale database management system. MySQL® can be used through a text oriented mysql client, GUI clients or through the browser using a server side scripting language, like in this case. All three can access the database via a

TCP/IP network but the text oriented Client can also communicate directly with the server over the console. To access the MySQL server with root-rights, connection parameters like hostname, password etc. have to be given. The whole communication is based on SQL statements. Further information regarding SQL was already stated in section 2.5.1. [42]

2.5.3.2 Bootstrap

Bootstrap (Mark Otto et al, USA, [47]) is a CSS framework and its principal advantage is its responsive grid system. The framework separates the browser window in a 12-column grid and window sizes are classified in four classes: extra small for a mobile device, small for tablets or netbooks, medium for notebooks and large for large screen desktops. The developer can now define for every class on how many columns an element should be extended. But there are a lot of bootstrap components which are commonly used and some of them are shown in Figure 10. [48]

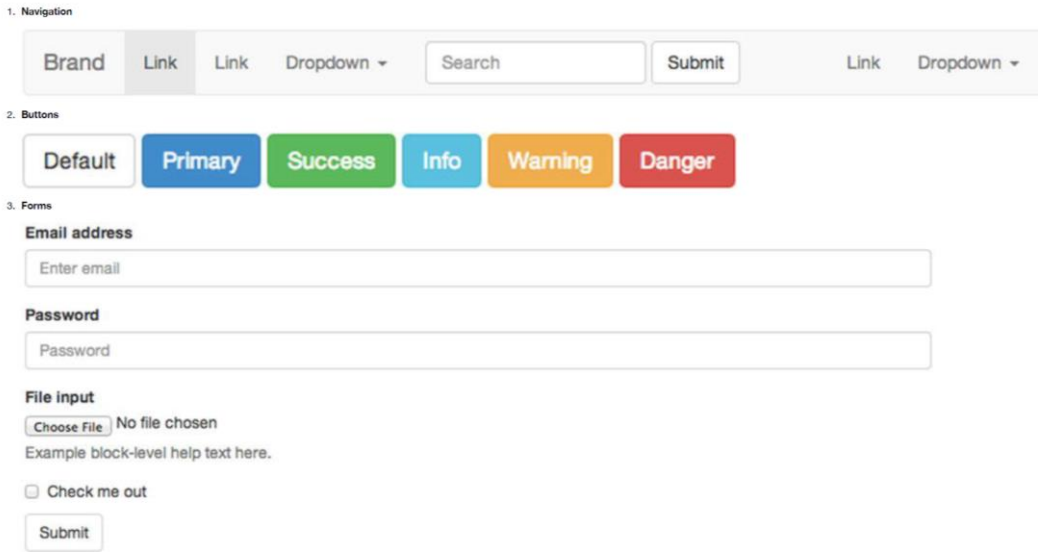


Figure 10: Commonly used bootstrap components (Navigation, Buttons, Forms). **[48]**

For the PBM-ABS app the Angular ui-bootstrap 13.3 library was used.

2.5.3.3 Other JS Libraries

The first step for using the PBM-ABS app is to upload a CSV file. Therefore the JS library Papa Parse was used to parse the CSV data to Java Script Object Notation (JSON) format for further processing. JSON has multiple hierarchies and is of key-value concept, which means the value can be obtained by using the corresponding key.

For all kinds of charts the highcharts (Highsoft, Vik i Sogn, Norway, [49]) library was used, which is a JS library for interactive chart creation. It supports more than 20 chart types and offers many features. HC adjusts itself to every screen size and is dynamic according to data updates and adding or removing axis or series. It is supported by almost all modern browsers and the charts are rendered using Scalable Vector Graphics (SVG). [50]

A lot of AITs web applications are already using highcharts, why its use was recommended and conform to corporate design. Essential for report generation is transforming the produced content to PDF format. For creating the PDF itself jsPDF (James Hall, Leeds, UK, [51]) library was used but first the content had to be transformed. This content consists mainly of tables and charts, and seemed the easiest solution in transforming each element onto a canvas. A canvas is an HTML element on which images, graphics or animations can be projected. For transforming HTML content, like text or tables onto a canvas html2canvas (Niklas von Hertzen, Helsinki, Finland, [52]) library was used. It was also possible to transform highcharts elements with this library, but because highcharts is using SVG, some parts of the charts were blurry. Consequently canvg(Gabe Lerner, Chicago, USA, [53]) library had to be used for highcharts elements, which is dependent on rgbcolor (Stoyan Stefanov, Los Angeles, USA) library and the exporting (Highsoft, Vik i Sogn, Norway, [49]) library of highcharts. Using these libraries was also recommended by the highcharts support.

3 Results

This chapter outlines a detailed description of all development, implementation, configuration and installation steps as well as verification results. The overall workflow of the PBM-ABS service, illustrated in Figure 11, is divided into the data extraction process in the AKH Wien and the PBM-ABS app. The data exports from the AIT-EDC system (see Table 1) fulfilled the specification and therefore no further adaptation was required.

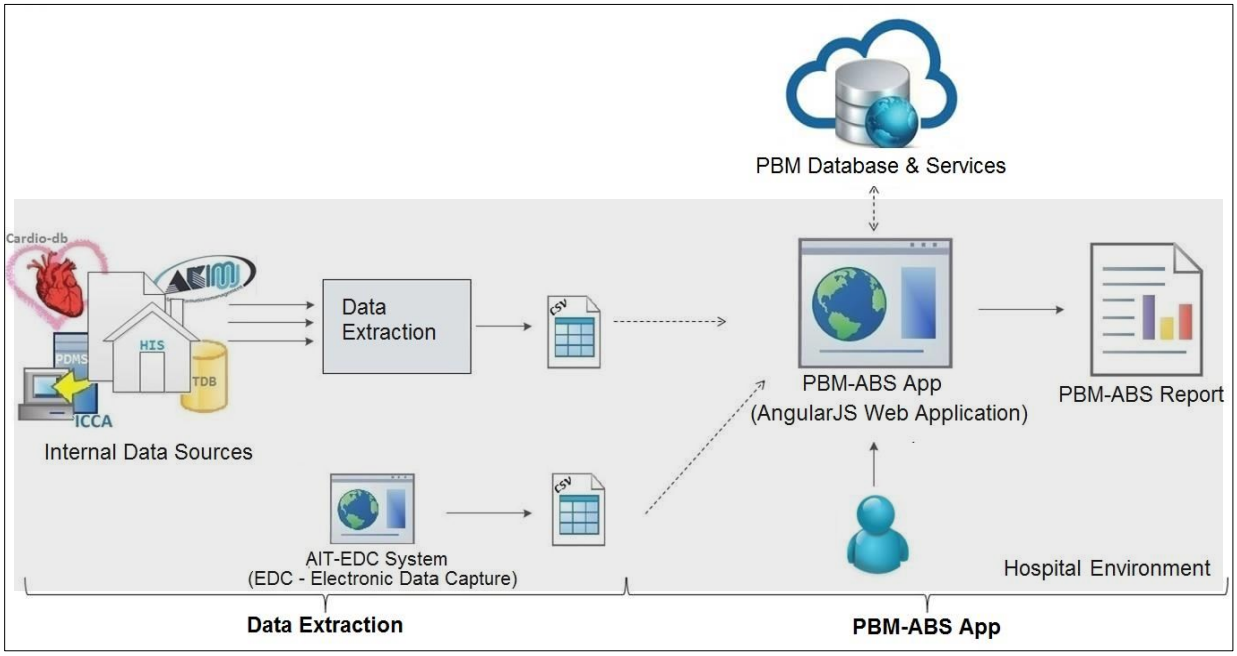


Figure 11: Overall workflow of the PBM-ABS service

An EDC system has the following advantages:

- As a stand-alone system, which means neither integration into the hospital's IT-infrastructure, nor cooperation or coordination with the hospital's IT-department is required.
- It is mature since it was successfully used and tested in many data acquisition processes in Austria
- It is tested for its usability, therefore easy to use.

Nevertheless, there are important reasons why using an EDC system is not the optimal method of choice:

- EDC systems are time consuming for clinicians because beside internal data capture for the hospital, all data have to be entered for a second time.
- Accordingly, additional personal resources have to be provided to ensure that the data entry is on time.

- Repeated data entry may cause individual faults (e.g. double data entry, source data verification), which makes time consuming quality checks essential in order to ensure data accuracy and reliability.
- Possible individual case selection may lead to non-consecutive data sets. This has to be checked, because consecutive data selection is an important requirement for reliable intra- and external for benchmarking

3.1 Showcase for Retrospective Data Extraction from Existing Data Sources in the AKH Wien

As previously mentioned, Figure 11 shows another kind of architectural design illustration of the overall workflow, in which the data extraction process in the AKH Wien makes up the first part, apart from extracting data from EDC systems. A more detailed illustration of this part can be seen in Figure 12.

In this diagram the four data sources are shown. As described in 2.2.2 for the TDB, the HIS and the Cardio-db an excel export was available, which is also indicated in Figure 12 as well as the direct access to the PDMS. Looking at the illustrated exports shown in Figure 12, all sources apart from the Cardio-db are containing an ID. This ID is internally called *BCMAC*, or *short MAC*, in the AKH Wien and is a case identifier. For PBM it is not important to have a unique patient identifier because every intervention is considered independently. The *BCMAC* was used for the core dataset. The identifier of the Cardio-db is the so called *MAC,* or *long MAC,* which is a patient identifier and is also contained in the HIS. For this purpose the Cardio-db had to be merged with the HIS data over the MAC first, to gain the case identifier for further steps.

For the showcase only parameters of the core dataset were extracted, even when the Cardio-db was containing some of the supplementary dataset parameters too. This was also because within the prototypical implementations of the PBM-ABS app, only analysis of the core dataset was included, but can be arbitrary extended. Table 4 shows the parameters of the core dataset and the sources in which they are contained.

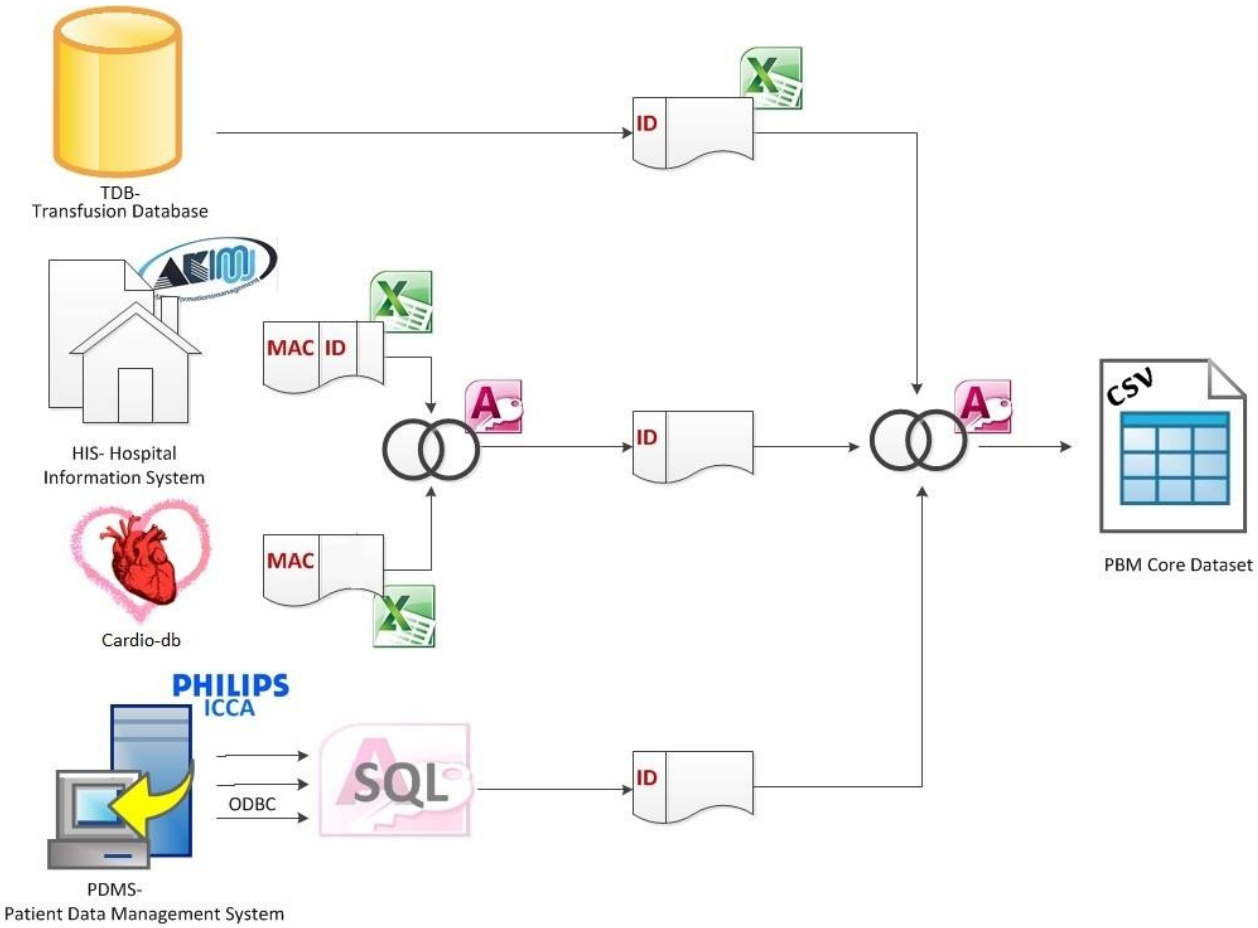


Figure 12: More detailed illustration of the data extraction part.

Table 4: PBM core dataset and corresponding data sources of the AKH Wien

Core-Dataset	
Parameter	**Source**
ID	All sources
Surgery date	PDMS or Cardio-db
Length of stay (days)	-
Intervention (3 letter code)	Cardio-db, (PDMS)
Gender (m/f)	Cardio-db, PDMS, HIS
Age (years)	Cardio-db, PDMS
Weight (kg)	Cardio-db, PDMS
Size (m)	Cardio-db, PDMS
Allogeneic RBC transfused (number of units total)	Cardio-db, TDB
FFP (units total)	Cardio-db, TDB
Platelet concentrates (yes/no total)	Cardio-db, PDMS
Coagulation factor concentrates (yes/no total)	Cardio-db, PDMS
Preoperative haemoglobin value (g/dl)	Cardio-db, PDMS, HIS
Haemoglobin value of 5th postoperative day (g/dl)	PDMS, HIS
Minimal haemoglobin value/ hb_nadir (g/dl)	PDMS, HIS

Looking at Table 4, there could already twelve of fifteen parameters of the core dataset be extracted from the PDMS under the assumption that all needed Hb values can be retrieved from the *HB and the HB_time* parameter and the intervention can be derived from the primary diagnosis. Actually this was not the case, because the PDMS contains just data from the ICU. If there are no adverse events the patient is just staying at the ICU after surgery and the rest of the time at the normal ward. Accordingly the PDMS contains almost no preoperative data. The data of the normal ward should be in the HIS. Even if the data about how many RBC units or other blood components were transfused should also be available in the PDMS, the transfusion database was used to retrieve this information because of higher data accuracy without dependency on the ward where the patient was located.

The following sections will describe each part of the process, which is needed to gain the PBM core dataset in form of a csv file. This csv file constituted the goal of the data extraction workflow as well as the beginning of the PBM-ABS app workflow.

3.1.1 Data Extraction from the Patient Data Management System (PDMS)

3.1.1.1 Basic Data Structure of the PDMS

As mentioned in section 2.5.1, before constructing an SQL query to extract the required data, a detailed analysis of the database structure had to be performed. Therefore existing SQL queries provided by Philips were analysed step by step to obtain the structure and relationships shown in Figure 13.

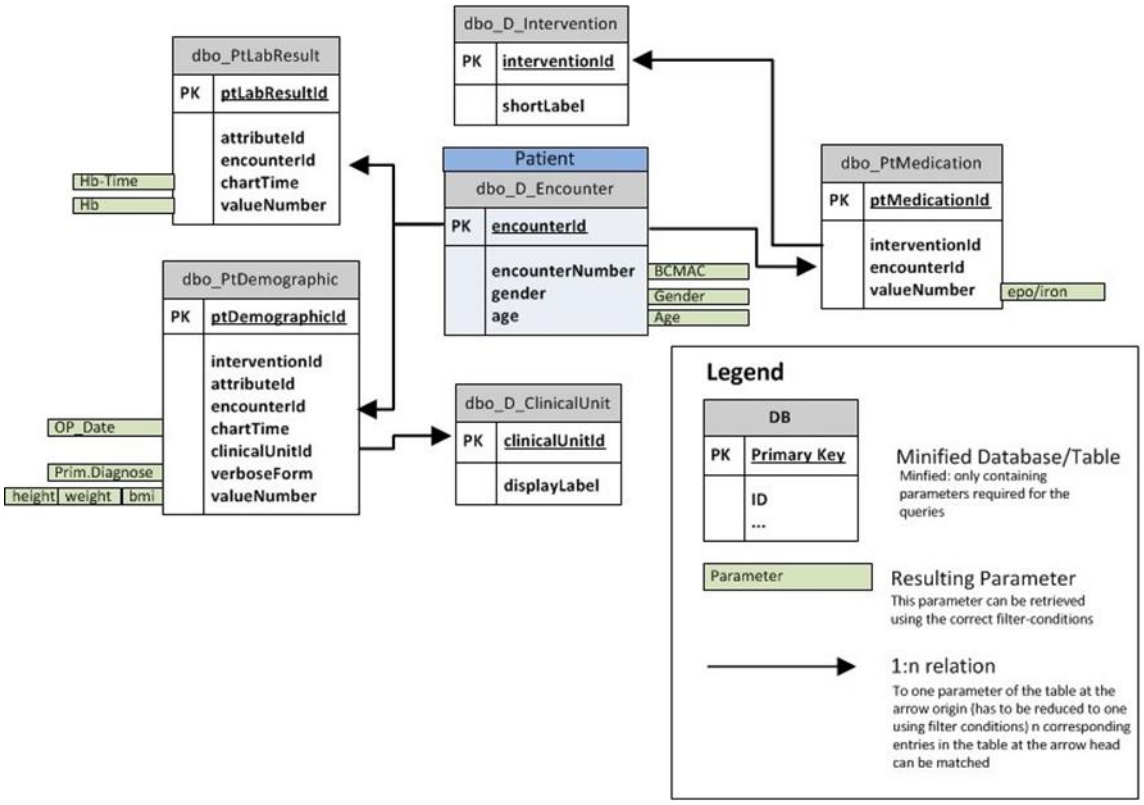


Figure 13: Database structure of the PDMS reconstructed by analysing existing queries

If the query should retrieve patient information the main table has to be *dbo_D_Encounter,* also marked as “Patient” in Figure 13. This means, that if no filter is applied all entries of this table will be shown in the query result. This table already includes the case identifier (BCMAC), age and gender of the patient. The case identifier is called *encounterNumber* within the PDMS database. Another ID used within the PDMS databse is the *encounterId*, which is also the primary key of the *dbo_D_Encounter* table. The primary key signifies the parameter which is used as identifier for each row and implies that every entry is unique inside this table. The arrow pointing to the *dbo_PtDemographic* table represents a 1:n relationship, which means there are n entries in this table connected to one encounterId of *dbo_D_Encounter.* These entries contain for example the surgery date, height, weight or BMI of the patient.

3.1.1.2 Data Extraction Measures

In the following sections the queries will be described in detail. Figure 14 shows a legend describing the required query elements used in the following database transaction diagrams.

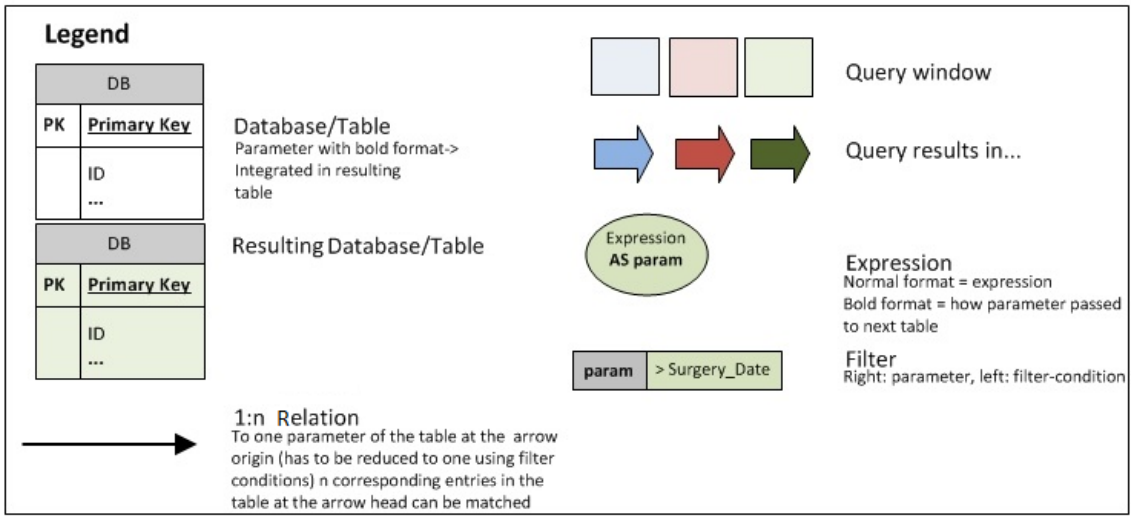


Figure 14: Legend for database transaction diagrams

The databases and tables as well as the 1:n relation are illustrated in the same way as in Figure 13. Inside of a query window expressions and filters are applied on the tables. Every query window has a resulting database/table which is indicated by the bold arrows.
Figure 15 shows the query that extracts the data from the PDMS, also including filters and expressions. For example to retrieve parameters of the patient from the *dbo_PtDemographic* table filters have to be applied. To get the height of the patient the table has to be filtered for entries with “*attributeId =11637”* then the parameter *valueNumber* will return the hight of the patients. The majority of the other subqueries are performed more or less analogously. It is also possible to perform this kind of query on more than two tables. The process steps

depicted with symbols of elliptical shape are indicating expressions which are applied. An expression can be an if-else expression, an aggregation function or just the renaming of a parameter. An expression is for example applied in the subquery, marked with the red dashed border in Figure 15.
In this expression an aggregation function is applied on the filtered tables. For example "GROUP BY encounterNumber; Count(valueNumber) AS ei" means that all entries with the same *encounterNumber* will be grouped and instead of the *valueNumber* the number of non-null entries of the parameter *valueNumber* will be returned. The new parameter will be called *ei*. Afterwards an if-else expression is performed, which returns *yes* if *ei* is greater than zero otherwise it will return *no*. The *AS* expression once again renames the parameter as *epo/iron* and returns if the patient has received erythropoietin(epo) or iron. Other expressions used in Figure 15 are performed analogously. For information about the SQL syntax see chapter 2.5.1.

3.1.1.3 Data Extraction Steps of the PDMS Query

In the PDMS query in Figure 15 following steps were performed:

- First the case-ID, gender and age are extracted from the *dbo_D_Encounter* table together with the surgery date from the *dbo_PtDemographic* table.
- Then the primary diagnose, age, weight, height and body mass index are extracted from *dbo_PtDemographic* table. (If height or weight is missing, it could be calculated from the body mass index)
- If epo or iron, tranexamic acid, factor concentrates or platelets have been transfused was extracted from the *dbo_PtMedication* table using the *dbo_D_Intervention* table for filtering.
- Hb values and corresponding Hb time (time when the value was entered into the sytem) were extracted from the *dbo_PtLabResult* table.
- Finally all extracted parameters were merged into one table named *ICCA.*

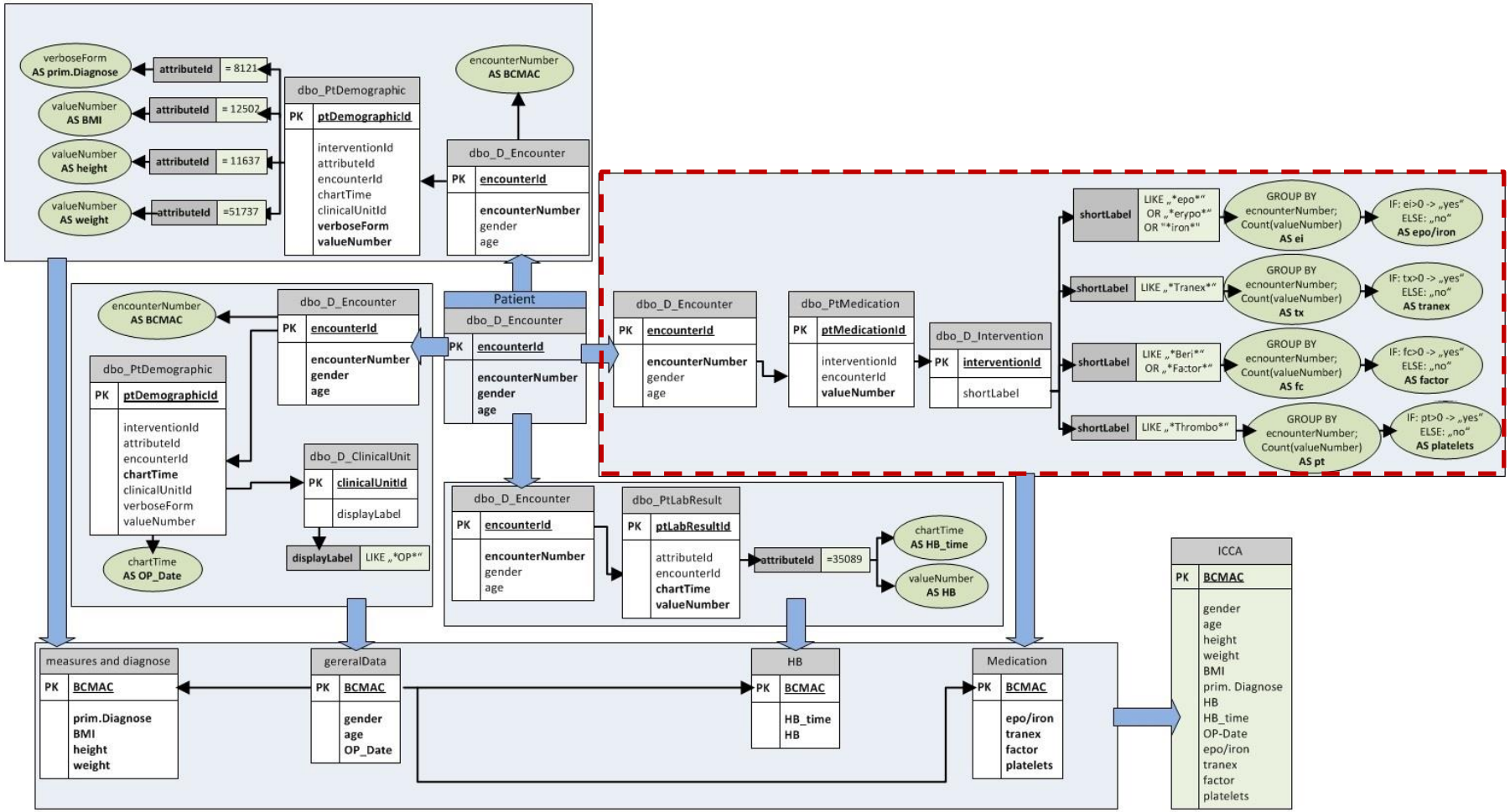


Figure 15: Query to retrieve the required parameters from the PDMS

3.1.2 Merging Data from Internal Data Sources

3.1.2.1 Data Availability

As stated in 2.2.2 the data quality of combining those three sources was not sufficient (see 3.1.3). The major problem was that the intervention could not be determined. There were some different kinds of diagnoses available in the HIS, but in most cases this information was not consequently documented in a structured way, but in free text. All diagnoses from the HIS were already imported into the PDMS under “primary diagnosis”, because there have already been some efforts of the AKH Wien to enable data exchange between the two systems. That was why the primary diagnosis of the PDMS was used for attempts of retrieving the intervention, but provided inexact results because of the free text representation.

Nevertheless, after all efforts it was only possible to identify a small fraction of total cases with the desired interventions. There have been some other data quality issues, like e.g. double entries for height – sometimes showing more than 10 cm differences - missing parameters and for some parameters only parts of the required information could be extracted. Therefore, the Cardio-db was used as a fourth source to extract additional data of cardiac surgery cases. The Cardio-db was introduced and administered by the department of Cardiothoracic and Vascular Anaesthesiology. Entering the required parameters into this database is well integrated in the daily hospital routine of the physicians, resulting in a high completeness of the database entries. For this reason it was decided to use the Cardio-db as primary source which then were extended by parameters retrieved from the other data sources.

3.1.2.2 Data Extraction

Figure 16 shows the first part of the query for merging the data of the four sources. This part contained following extraction steps:

- The left part of Figure 16 was to match the patient ids (MAC) in the Cardio-db to the case ids(BCMAC) by using the HIS export, which contains both of them.
- Because time of surgery - which is relevant to derive the correct data from transfusion database - was not available from the Cardio-db, the time of surgery, if available, was retrieved from the PDMS.

- The resulting table called *Step0_heart* was then containing all required parameters of the Cardio-db, the BCMACs which could be matched and reduce surgery time.

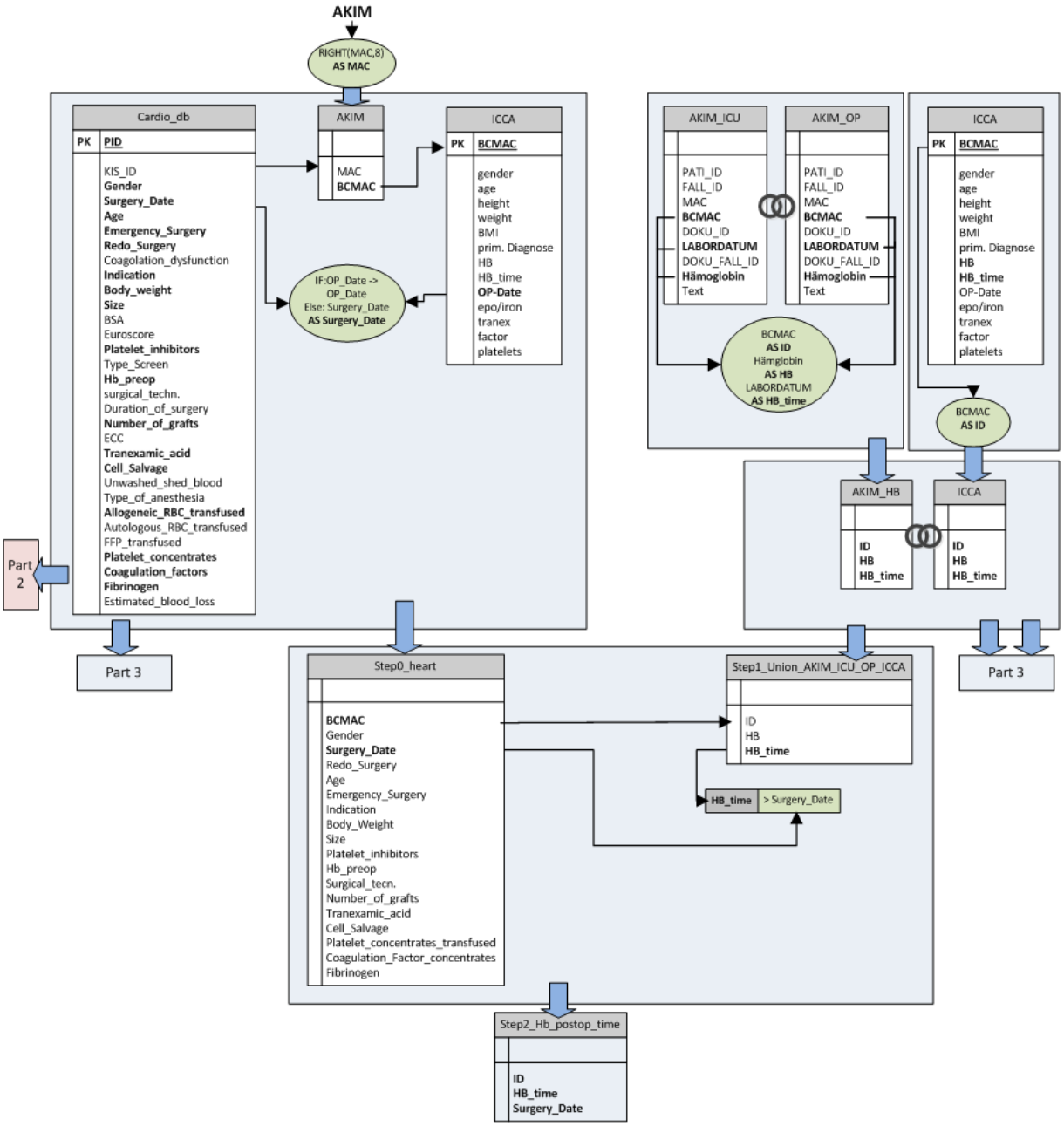


Figure 16: First part of the query for merging data of all four sources.

- In the right part of Figure 16 all Hb-values and the corresponding entering time of HIS and PDMS were joined into one table. From the HIS the following data exports were available:
    - Diagnoses- containing primary diagnoses (*AKIM*)
    - Intraoperative Hb values and associated laboratory parameters from operation room (*AKIM_OP*)

- Postoperative Hb values and associated laboratory parameters from intensive care unit (*AKIM_ICU*)

Lots of these Hb values existed in all three tables. Because of using the SELECT DISTINCT keyword for the query instead of just SELECT, the resultant table called *Step1_Union_AKIM_ICU_OP_ICCA* included all entries of all sources but only once. Because the preoperative Hb was already available in the Cardio-db, Hb values from blood samples before the surgery date were excluded. The resulting *Step2_Hb_postop_time* table contained just Hb values measured during or after the surgery.

The PDMS was also containing preoperative Hb values because they are automatically imported from the laboratory, but because the preoperative Hb was available in the Cardio-db, this value was used.

The second part shown in Figure 17 concerns the TDB:

- For comparing results of how many transfusions a patient received, from the Cardio-db and the TDB, a filter was included in the query where it was possible to just return the number of RBC or FFP units transfused from the Cardiothoracic and Vascular Anaesthesiology ward. This filter was only used for retrieving the results shown in Figure 21 (section 3.1.3), otherwise it was inactive.
- Blood components units with a delivery date lying out of a time range of three days before and seven days after surgery were excluded.
- Entries were later divided into RBC units and FFP units.
- The RBC units table was then separated into one table only containing units with status transfused and one with all units to get the number of ordered RBC units. They were then grouped and matched to case identifier and surgery date.
- The resulting table denoted as *Step8_join* was then containing BCMAC, surgery date, number of transfused RBC and FFP units as well as ordered RBC units.

Figure 18 illustrates the third part of this query. It is the continuation of part 1 for retrieving the required postoperative Hb values:

- Now the Hb values are separated into values between the 3rd and the 4^{th,} 4th and 5th and 5th and 6th postoperative day and are categorised as hb_3postop, hb_4postop and hb_5postop. The perioperative period is defined from the beginning of the surgery until the 5th postoperative day [2]. On the 5th postoperative day all influences from medication given during the surgery should be compensated which allows representative conclusions. If the Hb value of the 5th postoperative day is not available, the latest possible value should be taken as recommended in [15].

- Then an if-else expression is applied taking the hb_5postop if available, otherwise the hb_4postop and again if not available the hb_3postop as hb_postop. If no postoperative Hb value can be found for this patient hb_postop is set to null.
- The minimal postoperative Hb of the patient which can be determined in the given time range is categorised as hb_nadir (nadir = lowest point).
- In the last subquery all Hb values and all required parameters of the *Step0_heart* table constitute the *Step6_heat_and_hb* table.

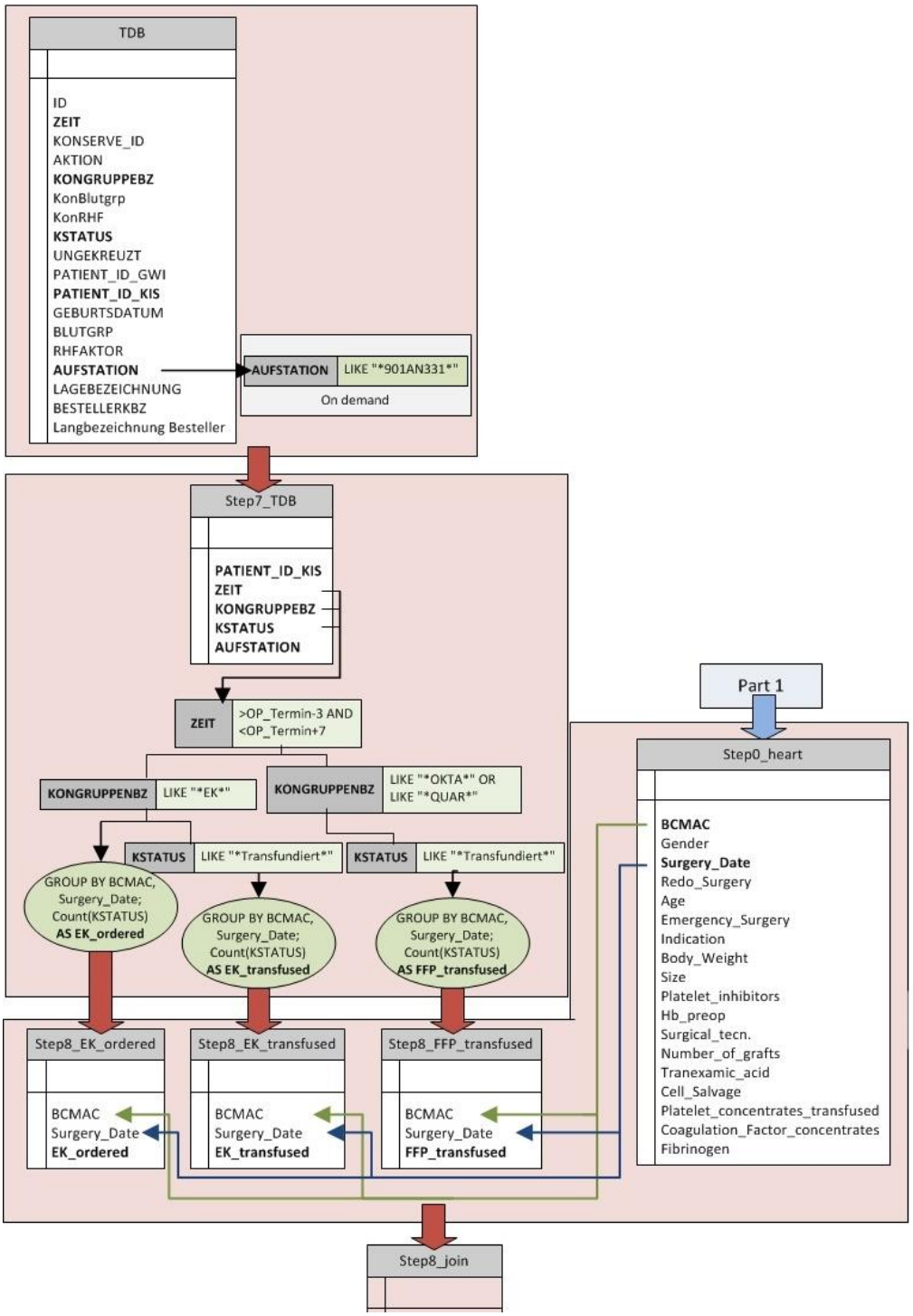


Figure 17: Second part of the query for merging data of all four sources.

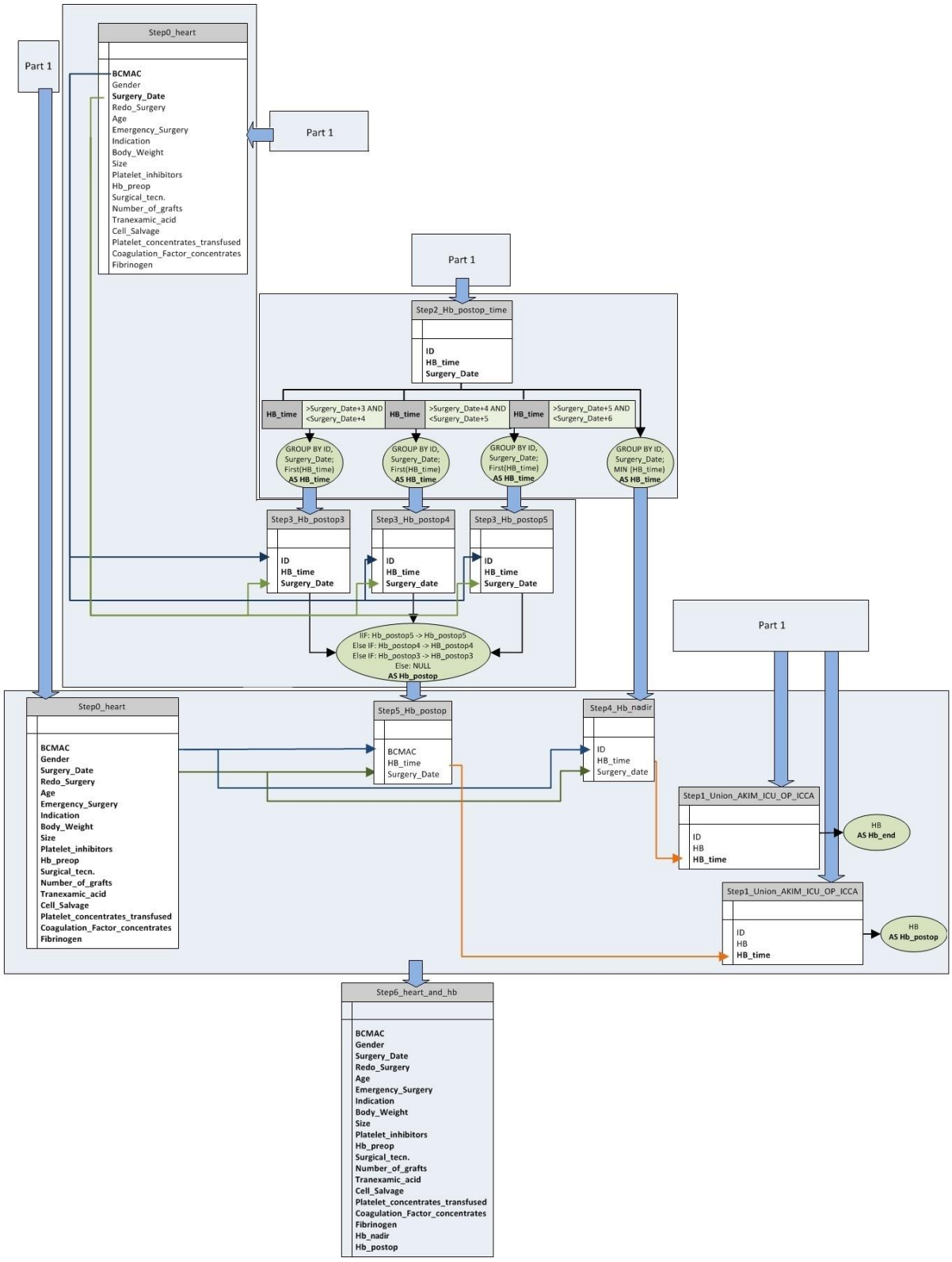


Figure 18: Third part of the query for merging data of all four sources.

Finally part 4, shown in Figure 19, combines the results of part 2 and part 3. How the four parts interact can be seen in Figure 20.

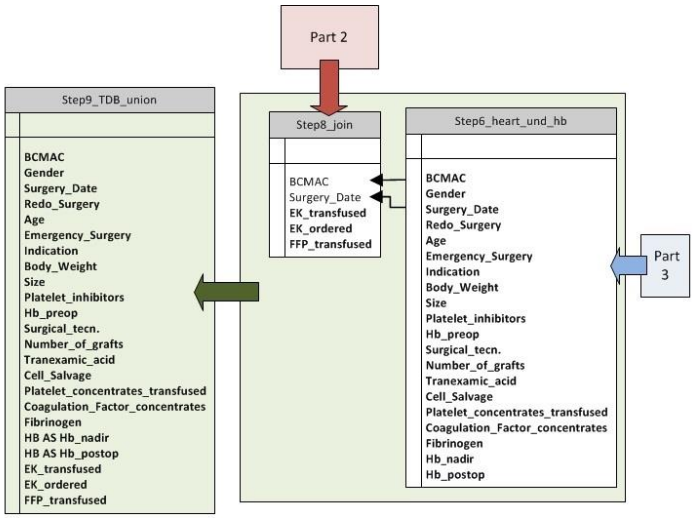


Figure 19: Fourth part of the query for merging data of all four sources.

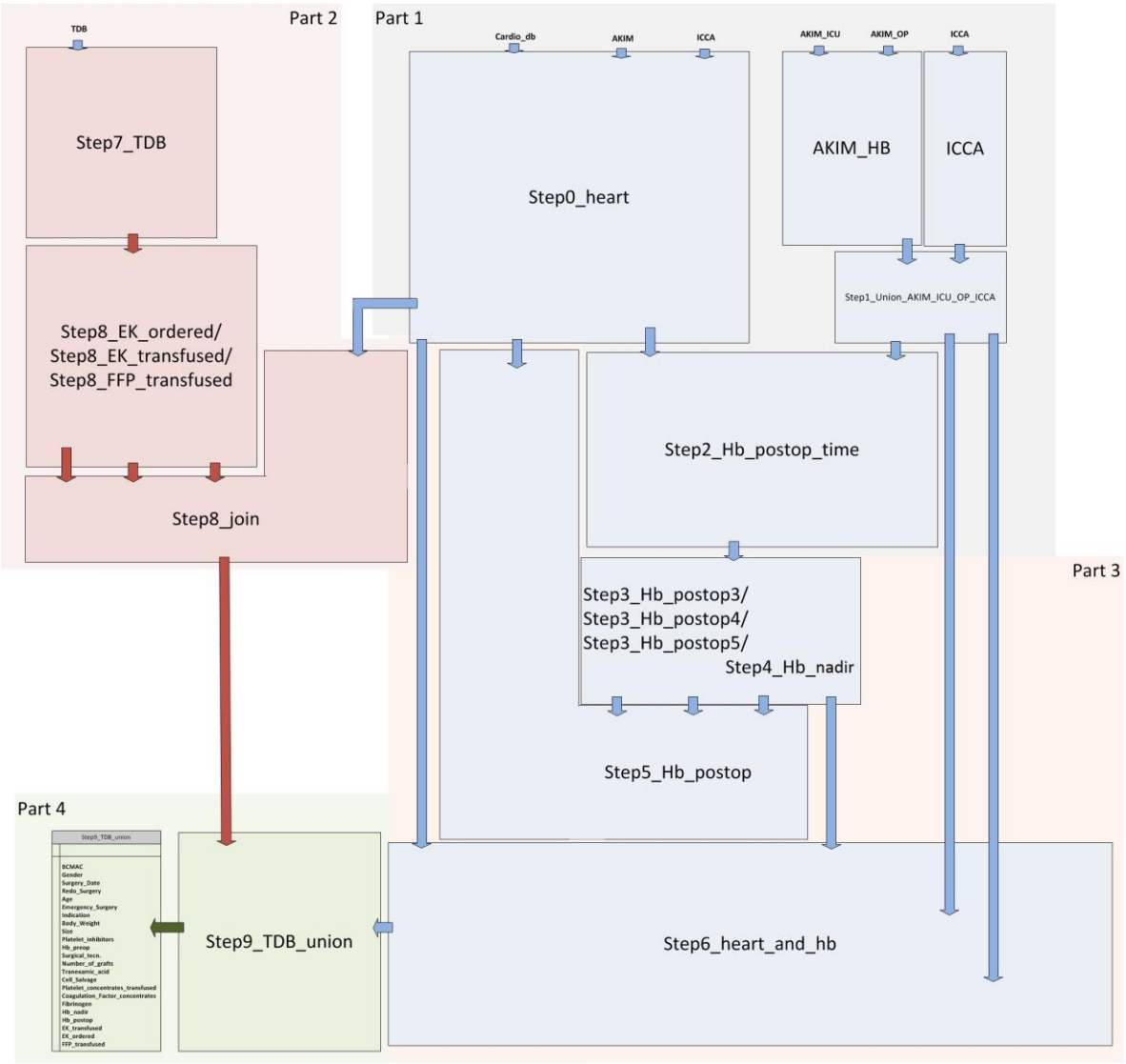


Figure 20: Overall query showing interaction of the four subquery parts.

3.1.3 Resulting Tables of the Data Extraction Queries

3.1.3.1 PDMS (ICCA) Query

The unfiltered query result had 588051 entries of 65620 patients for a time range between the beginning of December 2013 and the end of October 2015. To find cases with interventions of interest filters were applied on the primary diagnosis field. The interventions of interest in the AKH Wien were Coronary artery bypass surgery (CABG, or three letter code: CAR), with no valves and between one and four bypass grafts, Pancreas resection (PAN) and total hip endoprosthesis (H-TEP, or three letter code: THR). Table 5 shows the number of the extracted cases after applying filters for determining surgeries belonging to the three interventions of interest.

Table 5: Applied filters and number of cases from the extracted PDMS table

Intervention	Filter conditions	Number of cases
CABG (CAR)	LIKE "*CABG*"	214
H-TEP (THR)	LIKE "*TEP*" AND NOT LIKE "*Knie*"	94
PAN	LIKE "*Pan*"	194

These are fewer cases than expected. Regarding CABG cases, there were 214 extracted cases from the PDMS as shown in Table 5. But comparing this with the Cardio-db cases from 2014, which can be assumed as ground truth, these are far too less cases, because the Cardio-db contained 507 CABG cases just for the year 2014.

As mentioned in 2.4, for verification of data exports from PDMS, the only way to verify the resultant PDMS table was to compare some parameters of the entries which could be matched, to the corresponding entries of the Cardio-db. After filtering the CABG cases once again for having between one and four bypass grafts and for surgeries without valves, as well as for not having the apposition "diverse", there were 313 cases left. Table 6 shows the equality of the Cardio-db and the extracted PDMS table for the parameters gender, age and size.

Table 6: Equality of cases of the extracted PDMS table and the Cardio-db

Parameter	Overall # of cases	# of equal cases	[%]
gender	313	310	99,0
age	313	309	98,7
height	313	262	83,7

The greatest deviation of the two tables was for the height of the patients. Out of 51 unequal cases, 29 were because of missing values and 22 because of different values.

3.1.3.2 Query for Merging the Data Sources

As documented in 2.4, 30 random samples were checked for correctness of the matching and the aggregation functions. Regarding both categories 100% of the 30 random samples were correct. This means, that all 30 entries of the Cardio-db were matched to the correct entries of the PDMS, HIS and TDB as well as the correct number of RBC transfusions were determined using the “Count” aggregation function of Access.
The resulting table of the query contained 314 cases. 232 (73,0%) complete data rows and 82 (27%) with containing missing values. The Cardio-db export was also filtered for CABG-cases with one to four bypass grafts, without valves and apposition “diverse” before it was imported to Access for performing the query.

A comparison between the number of transfused RBC units according to the TDB and the Cardio-db was done. For this purpose the “on demand” filter shown in Figure 17 was applied to extract just the units which were deliverd to the Cardiothoracic and Vascular Anaesthesiology ward. The TDB was chosen as reference table. The results are shown in Figure 21 and Table 7.

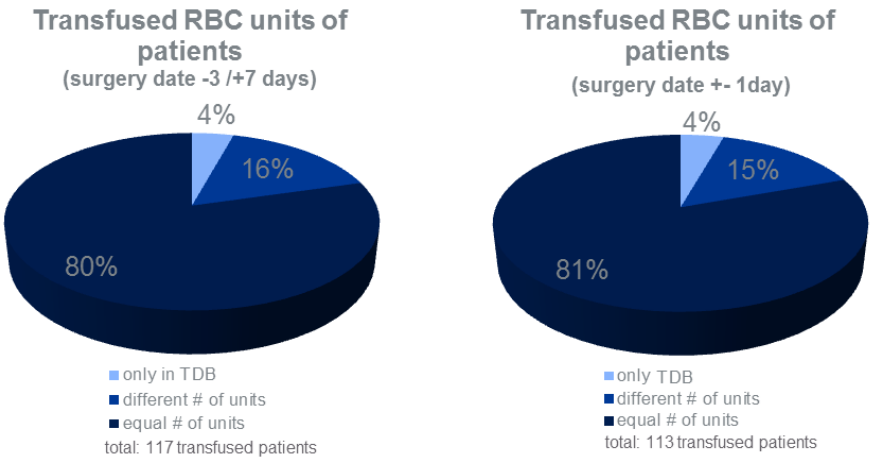


Figure 21: Pie diagram of concordance comparing transfused RBC units of TDB and Cardio-db

Table 7: Accordance of transfused RBC units per patient comparing TDB and Cardio-db

	±1 day	**-3 days and +7 days**
Number of transfused patients	113	117
Same number of transfused RBC units	91	93
Different number of transfused RBC units	17	19
Only transfused according to the TDB	5	5

To obtain results for the other interventions the available cases from the PDMS were used. To retrieve pancreatic resection cases the primary diagnosis field was filtered for “Panc” or “Pank” because the diagnosis field is a free text field and both terms, “Pancreas” and “Pankreas” were used. After consultation with responsible physicians a new strategy was found for extracting THR-cases. First, the primary diagnosis field was filtered for “Coxarthrose” or “Koxarthrose”, because almost all THR surgeries have a previous diagnosis of coxarthrosis. Also if a surgery date for the cases could be found it was further checked manually if this patient actually received a hip implant and an implant pass. Of 193 cases in the year 2014 69 remained after manual check. On the basis of these cases the rest of the query was performed analogously.

3.1.3.3 Additional Results Based on Internal Data Extraction

An essential benefit of extracting the data directly from internal sources is that the extraction can be arbitrarily extended. The internal data sources of a hospital contain very detailed information which can be analysed in multiple ways. This allows accessing the effectiveness of newly implemented changes in processes and logistics.

Just in Time Ordering and Single Unit Strategy

To improve the fast supply of transfusion products within the hospital a tube delivery systems was implemented in AKH Wien. Additionally, the ordering process was switched to “just in time” ordering, which should support the physicians to change their behaviour from double unit to a single unit transfusion strategy. This strategy implies *“gradual and controlled replacement of the deficient RBC volume until the recommended transfusion threshold is reached* [2]*.”*

Figure 22 and Figure 23 illustrate how these changes can be supported by internal data analytics. Figure 22 shows the increase of single unit orderings in 2015 for OR-Groups and ICU. Figure 23 shows the transfusion timeline of a patient. By analysing such timelines of random samples it can be checked the effectiveness of transfusions and if the transfusion behaviour complies with the guidelines of the single unit strategy. For this illustration, the Hb values of all internal sources were merged.

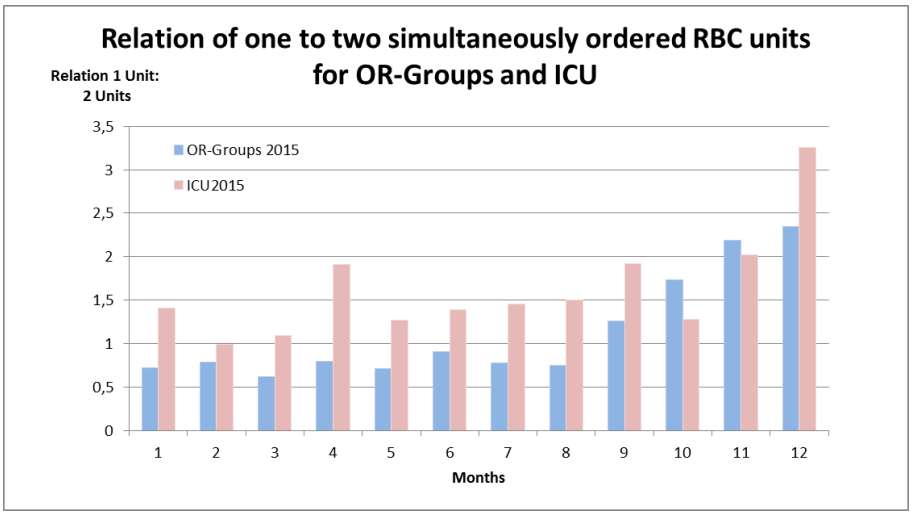


Figure 22: Relation of one to two simultaneously ordered RBC units within OR-Groups and ICU 2015

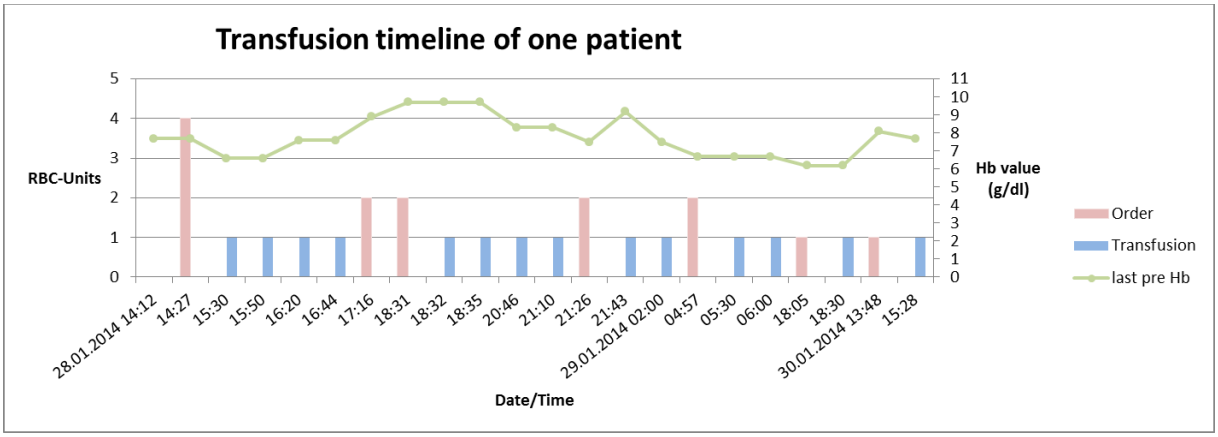


Figure 23: Transfusion timeline of a patient

3.2 PBM-ABS App

Figure 24 shows a flow diagram of the PBM-ABS app including the decision tree of the app. Additionally, a rough illustration of the implemented components is integrated to give an overview on the software architecture.

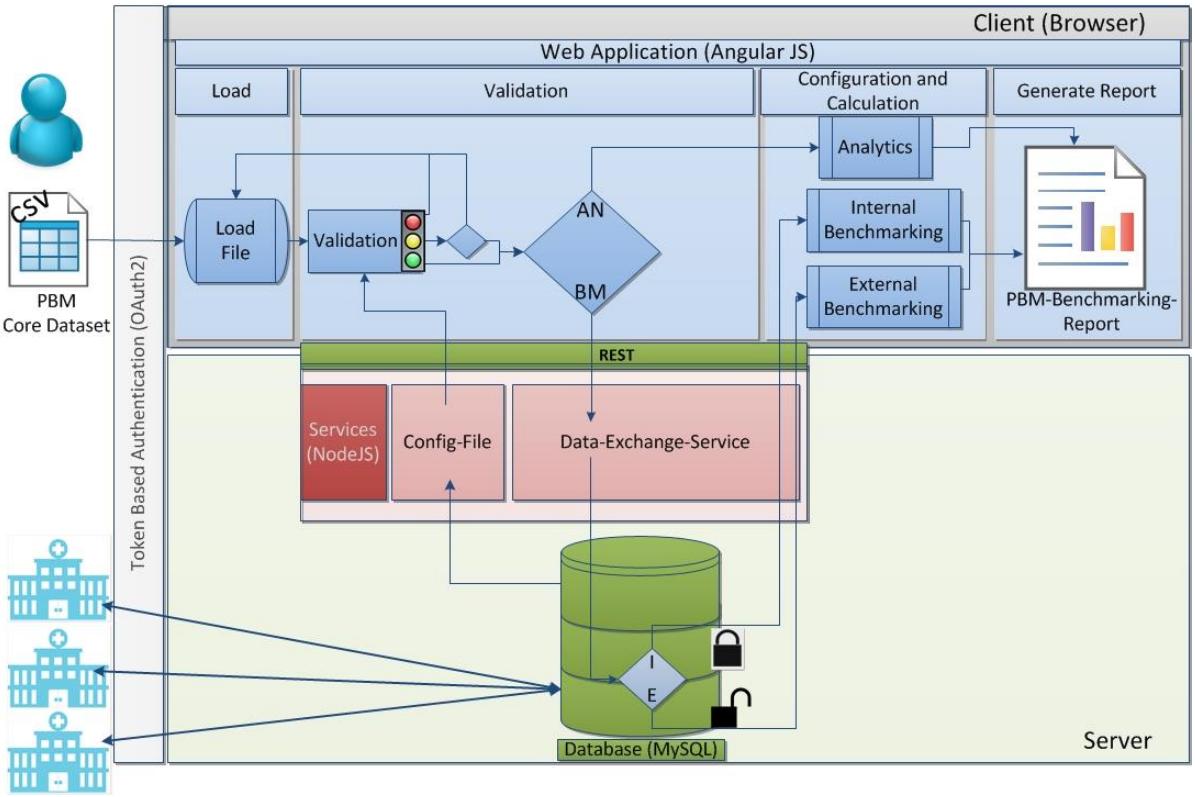


Figure 24: Software components and decision tree of the PBM-ABS app

The following section describes the 4 components of the client side app. The corresponding implementation steps are described in detail in section 3.2.2.

- **Load:** In this section, the user loads the csv file, containing the PBM core dataset into the application. The data is then converted to JSON format for further steps. The user also has the possibility to download an example csv file or an xls-file containing an example dataset and a description of all specifications, like plausibility thresholds or type definitions.
- **Validation:** In the validation part a preview of the loaded data is shown. The preview is displayed as a table, with different coloured fields. If there are valid rows, the user has the possibility to proceed, but invalid rows will be ignored. If there are no valid rows or the user is not satisfied with the feedback, he or she has to modify the file in a way that it meets the specifications and repeat the loading process. Before proceeding the user has to decide whether he or she just wants to use the analytics

features for the loaded data or the benchmarking features. When choosing benchmarking, the user has to agree with the terms and conditions because for this feature an upload to the PBM database will be required.

- **Configuration and calculation:** In this part the user has to do all required configurations for the desired report. If the benchmarking path was chosen the user has to decide if internal or external benchmarking is required.
  Further, the following decisions have to be made:
  - Report version: standard (predefined minimal version)
    extended (containing all possible charts and tables)
    user defined (the user can manually select from all possible charts and tables)
  - Time periods to be compared (only when internal benchmarking is included):
    Maximal three not intersecting timer ranges can be selected
  - If the analytics part is included the user has the possibility to interactively change some configurations of its tables and charts directly while seeing the results.

  When all configurations are done, the user can proceed to the report part.
- **Report:** For external benchmarking the centres for comparison can be selected. A bubble chart, showing the main PBM indicators (transfusion index and transfusion rate) as well as the number of patients for each centre and intervention is displayed to support the selection step. After selecting one or two other centres a preview of the report will be presented to the user. Subsequently the user has the possibility to create a PDF of this report. The PDF can be opened with any PDF viewer and accordingly printed or saved.

A more detailed presentation of the software components and how they are interacting is shown in Figure 25. It is an overview focusing on the location of the components and how authentication is realised.

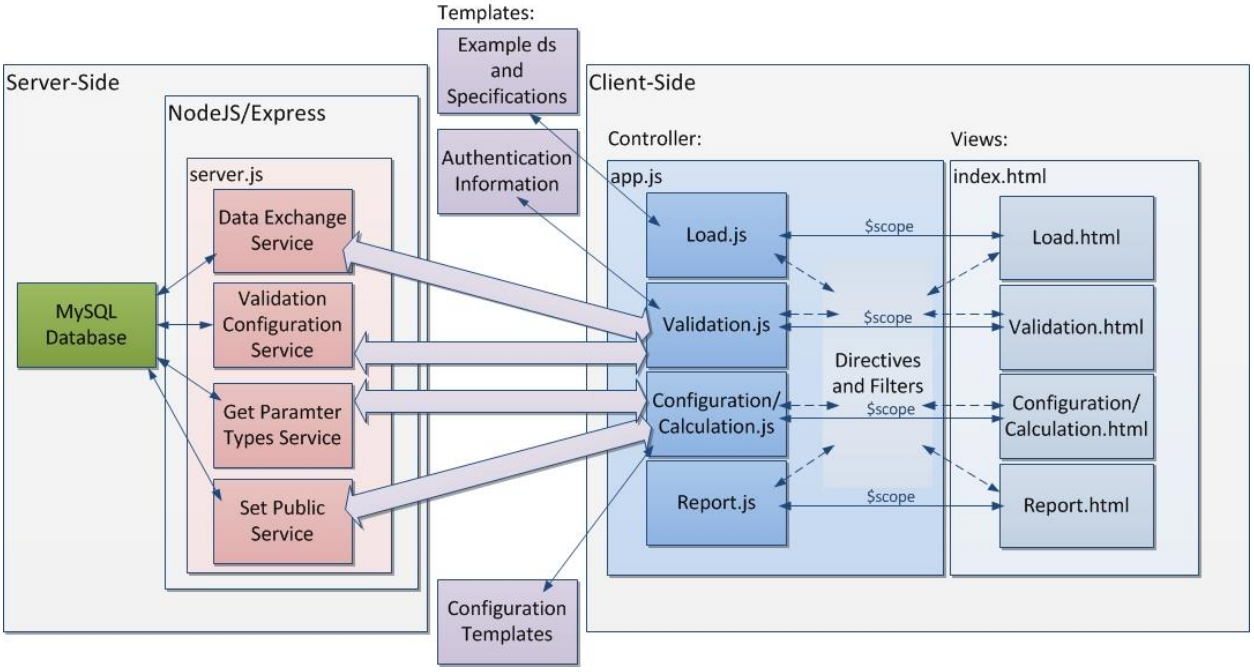


Figure 25: Detailed software architecture

In the ensuing sections implementation of the services will be described as well as the four components of the app itself: load, validation, configuration and calculation and report generation. Each component always contains a controller and the corresponding view. As can be seen in Figure 25 there is a strict binding between controllers and views. For more information see section 2.5.2. A description of the most important functions of the controllers, as well as directives, filters and templates interacting with the controllers and views will subsequently be given.

3.2.1 Services

As can be seen in Figure 25, the main file of the NodeJS/ Express server is server.js.
To enable all required functionalities, some node modules had first to be imported:

- The *morgan* module enables log requests to the console, which considerably eases the debugging process.
- The *express* module, see 2.3.2, is a framework which provides the actual webserver. This module is responsible for delivering all client side components of the app to the browser without any server side compilation or execution process.
- The *mysql* module provides the data exchange functionalities between the services and the database.

In the server.js file the port is set to any port which can be found in the environment and if no port can be found set to port 3000. It further contains the main interaction calls between server and app using service functions implemented in sub files.

3.2.1.1 Data Exchange Service

Initially, connection to the database has to be established. This was outsourced to an extra file, containing a *createSqlConnection*- and a *closeSqlConnection*- function, because it was required for all services. Therefore, the following parameters were required: host (localhost was used for developing) port, user, password and database name of the SQL database.

Once connection was established it was possible to perform queries using the connection.query function, which requires an SQL statement as a transfer parameter. The data exchange service contained two methods. The *getAllEntries* method queries all the data from the database and converts it into JSON format. The resulting JSON file will then be returned by the method. The *setNewEntries* method takes the data, also in form of a JSON file, sent from the application and creates the corresponding SQL statement to insert it into the database.

3.2.1.2 Validation Configuration Service

The validation configuration service just consists of the *getConfig* method, which is implemented analogously to *the getAllEntries* method of the data exchange service but instead the data table it is querying the validation table of the PBM database and converting it into JSON format.

3.2.1.3 Get Parameter Types Service

For the *getParamTypes* method, which is once again very similar to the *getAllEntries* method, instead of the SELECT keyword, the DESCRIBE keyword was used for the SQL statement.

3.2.1.4 Set Public Service

The *updatePrivacy* method of the set public service gets the ids of all entries of the current centre and the following SQL statement used for the query updates the corresponding entries: `"UPDATE data SET public = 1 WHERE id ='" + ids[id]+"'"`

3.2.2 Client Side App

As can be seen in Figure 25, the main components of the app itself were the app.js and the index.html file. The index.html file contained elements which are displayed in every step while using the app. These elements were header, footer and status bar, which are illustrated in Figure 26. The status bar only consisted of elements. The rest of the

representation was done by CSS and the ng-class directive. The step of the current location has the *active* CSS class ultimately it has the *complete* CSS class.

The main functionality contained in the app.js file is the routing. Therefore the *$routeProvider.when* function was used. (For more information see 2.5.2.7)
This shows an excerpt of the *$routeProvider* function:

```
$routeProvider.when("/load", {
    controller: "loadCtrl",
    templateUrl: "app/load/load.html",
    requireADLogin: false,//true,
}).when("/validation", {
    controller: "validationCtrl",
    templateUrl: "app/validation/validation.html",
    requireADLogin: false,//true,
```

It matches the keywords with the corresponding views and controllers.
Using the ng-view directive these html files can be inserted into the index.html, like the load.html file in Figure 26. The app.js file also contains additional implementation for elements like the loading spinner and error notifications.

3.2.2.1 Load

The load view is shown inside the blue frame in Figure 26.
It is just showing two example datasets of which the first contains the core dataset with mandatory parameters and the second one contains the supplementary dataset.

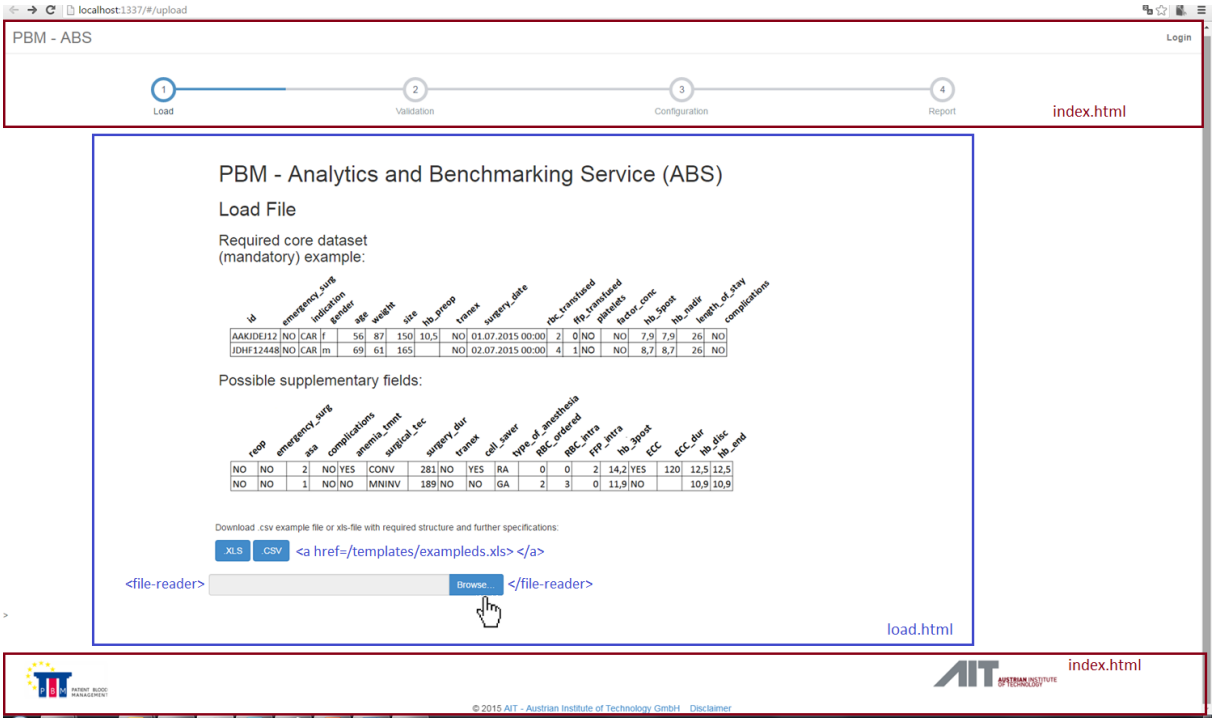


Figure 26: Screenshot of the start view with implementation comments and marked areas indicating the corresponding html files.

The *.XLS and* the *.CSV* buttons give the possibility to download an example dataset in that particular format. The excel file had an additional sheet giving a detailed description of all specifications. The *href* parameter, which can be seen in the html tag beside the buttons in Figure 26, linked the buttons to the files. This constituted the *Example ds and Specification Template* in Figure 25.

The <file-reader> tag in Figure 26 called the *fileReader* directive and is the usual way to call a directive from the html. This directive creates the file selection window. The transfer parameters of the directive were also passed using the tag, which means that through the html, $scope variables can be passed to the directive scope. This can be done as one- or two-way databinding (see 2.5.2). Those transfer parameters were required to restrict the file selection possibilities, e.g. if multiple files could be selected simultaneously, or which file endings and types were required.
Once the file was selected the directive imported the data and afterwards *papaparse* library was used to convert the imported data to JSON format to the *jsonData* variable. *jsonData.data* then contained the data itself and *jsonData.field* the parameter names.
If the directive execution ended with success status the $scope function *toValidation* was called, which changes the current location path from "/load" to "/validation". Otherwise an error message was displayed to the user.

3.2.2.2 Validation

Figure 27 shows screenshots cut-outs of two different validation screen scenarios. The fields of the preview table, showing the loaded data can be displayed in different colours:

- Green: the parameter is a mandatory field (field of the core dataset) and satisfies the specifications.
- Blue: the parameter is a supplementary field (field of the supplementray dataset) and satisfies the specifications.
- Red: the parameter is not satisfying the specifications.

Additional alerts give user feedback about the number of valid or invalid rows, missing or not required parameters:

- Information alert (blue): Gives feedback about the number of valid rows.
- Warning alert (yellow): Gives information about the number of incomplete rows, but the user could still proceed.
- Error alert (red): If the core dataset is not available, the user had to load a new file, containing the missing parameters.

The comments above and beyond the buttons are notifying the links, which has been selected or respectively the functions, which have been called when the button was clicked. The status bar changed to *active* for the validation step and to *complete* for load.

The preview table header contained the parameters of *jsonData.fields* and used the *fieldValidator* directive to check if the parameter names were meeting the specifications. The transfer parameters *wrongParams*, *missingParams* and *validParams* were $scope variables, passed to the directive with a two-way-databinding.
The ng-repeat directive was used to iterate over *jsonData.fields* (For information about ng-repeat see 2.5.2). This directive produced as many html elements of itself as values were contained in the array.

Then the *fieldValidator* directive checked every parameter for correctness of its name (not case sensitive). Depending on if the name was correct, a CSS class was added to the table field which changed the colour as mentioned above.

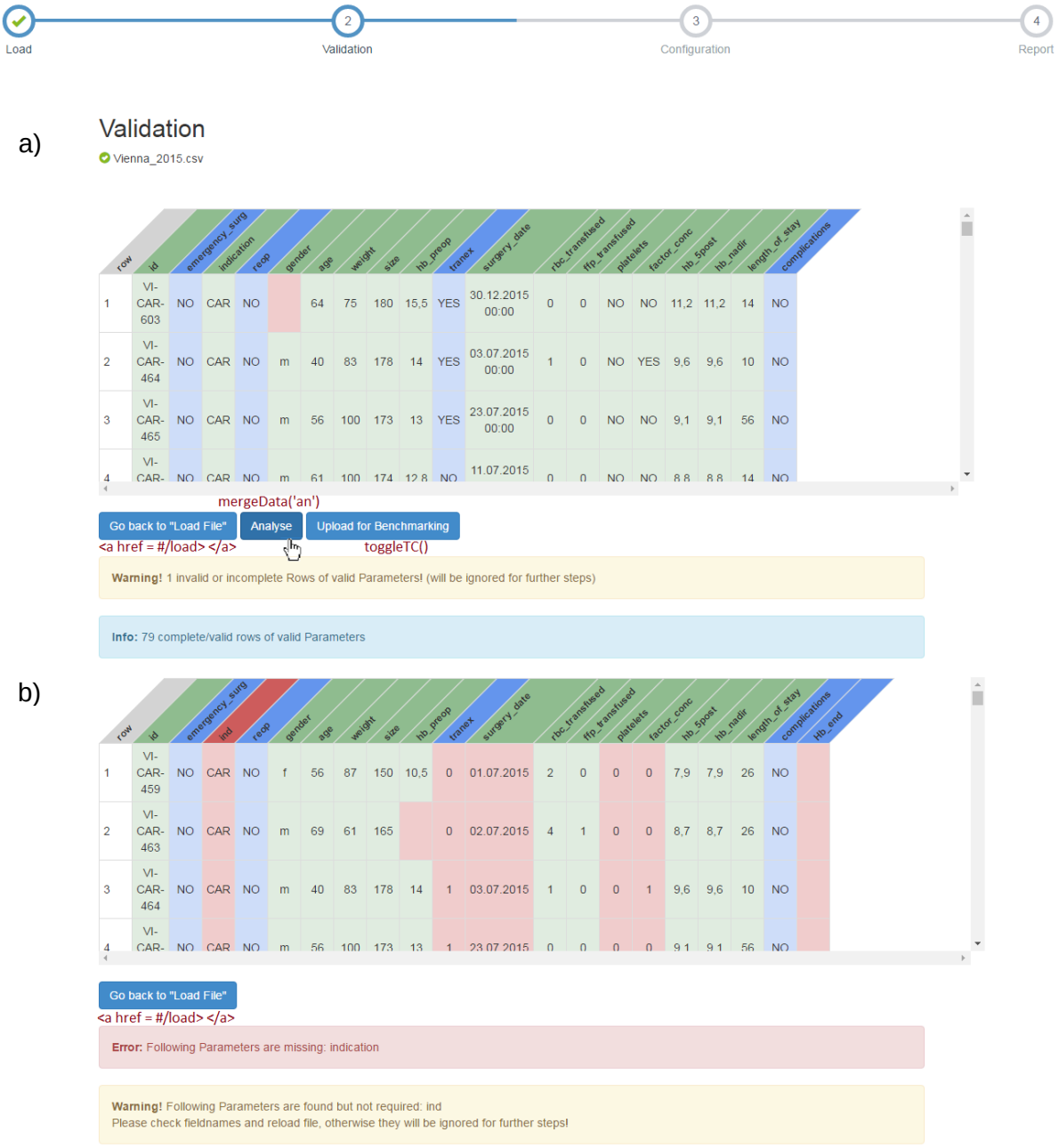


Figure 27: Two different scenarios of the validation screen with implementation comments. (a)First scenario shows the validation screen after upload of a complete dataset b) the second after upload of an incomplete dataset)

When the *missingParams* were passed to the directive, the array contained all parameters of the core dataset. If a parameter was correct, it was deleted from this array and added to the *validParams*, if incorrect it was added to *wrongParams*. These variables were then used for creating the alerts giving the user feedback. For feedback about the number of valid and invalid rows an additional $scope function was implemented. If *missingParams* was empty, the buttons *Analyse* and *Upload for Benchmarking* became visible.

The table body was evaluated by the *typeValidator* directive

The *typeValidator* directive allowed checking each parameter for correct data type and whether the plausibility criteria were fulfilled or not. The criteria were defined in the validation config file, which was a transfer parameter of this directive. Firstly loaded from the PBM database over a REST interface by a $http.get request (for more details see 2.5.2) and the validation configuration service. Analogous to the *fieldValidator* directive it added the corresponding CSS classes for changing the colour dependent on the correctness of the fields. If the corresponding field name was identified for being incorrect, the complete column could not be checked for correctness of type and for plausibility. Accordingly the column was coloured red, irrespective of its content.

As can be seen in Figure 27 a) if the *Analyse* button was clicked, the *mergeData()* has been called with the transfer parameter 'an', which stood for "analytics path". If the *Upload for Benchmarking* button was clicked the *toggleTC()* function has been called, which displayed the *terms and conditions* pop up. This popup was realised using the *termsConditions* directive. This directive put text and other elements, which were between its html tags into a popup window in front of a translucent dark grey background. It was also used in the configuration part and can be seen in Figure 28. Once the "terms and conditions" popup was shown the user either had to accept the terms and conditions by clicking the *Accept* button or skiped the popup. When clicking the *Accept* button, the *mergeData()* function was called with the transfer parameter 'bm' (bm = benchmarking) and the location path has been changed to "/configuration*"*, when skipping the popup the user returned to the validation screen.

As previously mentioned the *validation configuration file* was retrieved from the database via $http.get request and the *validation configuration service*. This happened right at the beginning when the validation controller was loaded as well as the $http.get requests fetching the data of the other centres from the database also using the *data exchange service.* A third $http.get request was located inside the *mergeData()* function and was retrieving mockup authentication information from the *authentication information template*. This authentication information was retrieved as a JSON file containing the shortcut and the full name of the current centre. Usually the authentication service should provide this information.

Once the information about the current centre was retrieved, the *mergeData()* function extracted only those parameters from the server data, which were also included in the loaded csv-file, by checking every parameter for being contained in the *validParams* array. This was only done under the condition that the *benchmarking* path was chosen. Further the valid rows from the loaded file were again added depending on the path. In the *analytics* path all

rows were added, in the *benchmarking* path only rows with new ids were added, which means they must not have already been included in the data from the server. Finally everything was stored in the *dataToReport* variable of the parent $scope. These rows which were not already contained in the server data were also added to the *dataToServer* $scope variable with the additional parameters: public=”No” and timestamp = new Date;
Depending on the path, the *toConfigurationBM()* or the *toConfigurationAnalyse()* function got called. If not already changed, they set the location path to “/configuration*”* and the *toConfigurationBM()* function additionally posted the content of *dataToServer* to the database by using a $http.post request and the *data exchange service*.

Two filters were also used inside the *mergeData()* function, one transformed a date string into date format and the other one transformed the date format into milliseconds representation. This was a common representation of dates and returned the milliseconds of a specific date counting form the 01.01.1970.

3.2.2.3 Configuration and Calculation

The configuration and calculation component included all preferences and settings which could be influenced by the user. If the user selects the benchmarking path, he or she can then choose between internal and external benchmarking. If external benchmarking was chosen the data of the current setting became viewable to other centres. Therefor the user had to agree that this would happen. At this point - for extending the prototypical implementation to a productive version - pseudonymisation of the data has to be ensured. For this agreement, the *termsConditions* directive was used again. In background Figure 28 shows the buttons for choosing internal or external benchmarking. In the foreground the pop up window for the privacy agreement can be seen.

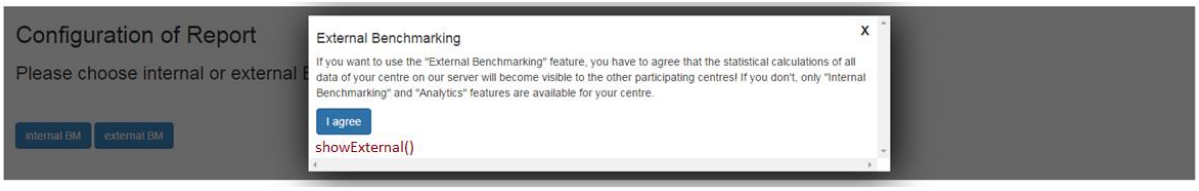


Figure 28: Popup for accepting the upload agreement

The $scope variable *toBeIncluded* contained all possible parts, which were set to *false* in the beginning. Figure 30 showed all configuration steps, which were required to define which tables and charts should be included in the report.
The three possible parts had a certain hierarchy:

- External benchmarking
- Internal benchmarking

- Analytics

This hierarchy of the features resulted from security reasons. For internal and external benchmarking the user was asked to agree with uploading the data to the PBM server. For external benchmarking the user was additionally asked to commit that the results of his or her own centre could become available for other participating centres. Features of a lower hierarchy could always be included. For example when the user chose the external benchmarking path, internal benchmarking could be included but not the other way round.

As can be seen in Figure 28 and Figure 30 if the benchmarking path was chosen either the *showInternal()* or the *showExternal()* function got called. These functions handled, which options were shown for parts that could be included. The parts which were already chosen were set to *true* inside the *toBeIncluded* variable. Additionally the *showExternal()* function set all entries of the current centre to public by using a $http.post request and the *set public service*.

Calculation of Additional Parameters

In the beginning, when the configuration and calculation controller was loaded, http.get requests were executed to get on the one hand all types of the parameters, additionally using the *get parameter types service* and on the other hand the configuration files for every part from the *configuration templates.* Inside the request for getting the parameter types, the retrieved types were used to reformat some parameters of the *dataToReport* variable, because this eased further calculations. At this point additional parameters were calculated using the corresponding filters. These filters were:

- The *hb_cat* filter: It added the *hb_cat* parameter, which contained the classification the filter has determined according to the preoperative Hb value of the entry and its position in relation to two predefined thresholds.
- The *transfused* filter: It filtered all entries of patients who received one or more RBC units and added the boolean *transfused* parameter, indicating if the patient received an RBC transfusion or not.
- The *bloodLoss* filter: It added the *blood_loss* variable, containing the blood loss during surgery according to the Mercuriali algorithm. A summarised formula for this calculation can be seen in Formula (3) adapted from [2].

$$Women: BloodVolume = 0{,}3561 \cdot height(m)^3 + 0{,}03308 * weight(kg) + 0{,}1833 \quad (1)$$

$$Men: BloodVolume = 0{,}3669 \cdot height(m)^3 + 0{,}03219 * weight(kg) + 0{,}6041 \quad (2)$$

Using Formula (1) and (2) it follows:

$$BloodLoss = BloodVolume \cdot \frac{(Hb_{preop} - Hb_{5postop}) \cdot 0{,}91}{MCHC} + RBC_{units} \cdot Volume_{RBC_{units}} \cdot Hct_{RBC\_units} \qquad (3)$$

The MCHC (mean corpuscular haemoglobin concentration) was not available, so according to [2] a value of 34 g/dl can be assumed. For Hct_{RBC_units} , which is the average haematocrit of the RBC units, the value of 43,5% was assumed. This value was chosen because according to [54] the normal haematocrit value of an adult lies between 42% and 45% of which 43,5% being the mean value.

Structure of the Configuration Template

Regarding the loaded configuration files, the data from these files was saved in the *reportParts* $scope variable. The configuration information was structured as can be seen on the example, showing the *transfusion index* table of the internal benchmarking part, in Figure 29. This information included: *title* (title of the table), *needed_params* (parameters required for constructing the table) and the corresponding classes, needed for calculation ("sep"= parameter required for separation, "av" = averaging required, "yn" = on basis of yes/no decisions, "cat" = category list required (e.g. for histograms), *details* (containing parameter labels for the tables), *extended* (extended parameters: not required for the table but can be included on demand), *selected* (set to true, if selected in the configuration for user defined report version), *std* (contained in the standard report verison), *separation* (separation parameters: parameters used for categorising the data).

```
"transfusion_index": {
  "title": "Transfusion-Index",
  "needed_params": {
    "indication": "sep",
    "rbc_transfused": "yn"
  },
  "details": {
    "rbc_transfused": {
      "yes": "patients transfused",
      "ysum": "rbc-units transfused",
      "yav": "TI"
    }
  },
  "extended": { },
  "selected": false,
  "std": true,
  "seperation": [ "indication" ]
},
```

Figure 29: Configuration information for transfusion index table.

The *prepareBM()* function, called when clicking the *Proceed* button in part 1 of Figure 30, deleted all parts which were not *true* according to the *toBeIncluded* variable. Another function was called inside *prepareBM(),* which checked if the parameters listed in *needed_params* of

every table were available and if not they have also been deleted from the *reportParts* variable.

Selection of Report-Content

Then the screen changed and the second part of Figure 30 became visible. When the user chose the analytics path at the end of the validation process, the other steps were skipped and he or she ended up at this step. Here the user had to decide how comprehensive the desired report should be. A standard version of the report would only include the most important tables and charts, which were selected by having the *std* parameter of their configuration set to *true* (Figure 29). The extended version would include all possible tables and charts and by choosing the user defined version the user got the possibility to select each table or chart manually (see part 3 Figure 30). For displaying part 3 of Figure 30, the *ng-repeat* directive was used, iterating over the *reportParts* variable for labelling the html input fields. As already mentioned, at this point the *reportParts* variable were just containing parts, already chosen by the user and elements of which all required parameters were available. Figure 31 shows an illustration of how the reportParts variable was changing dependent on the decisions of the user and which configuration steps had to be done.

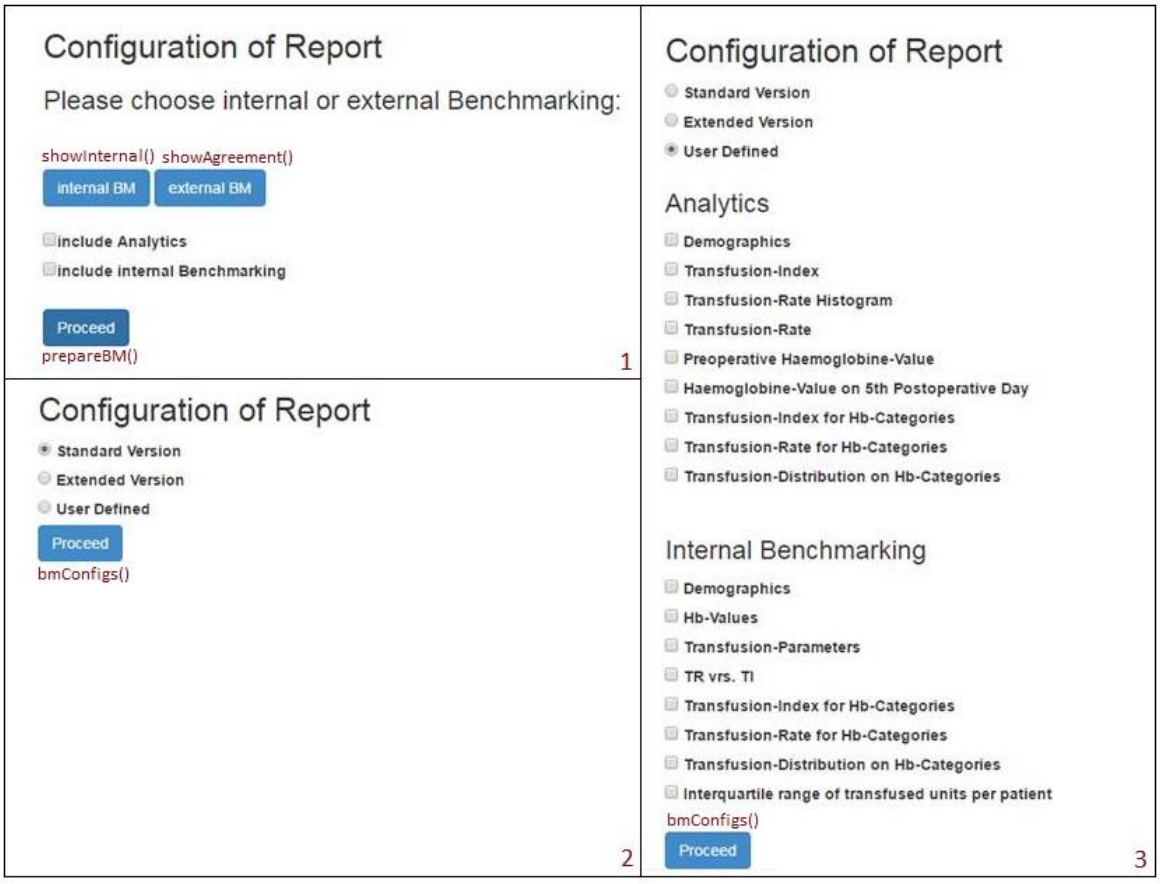


Figure 30: Three configuration steps to define, which tables and charts should be included in the report

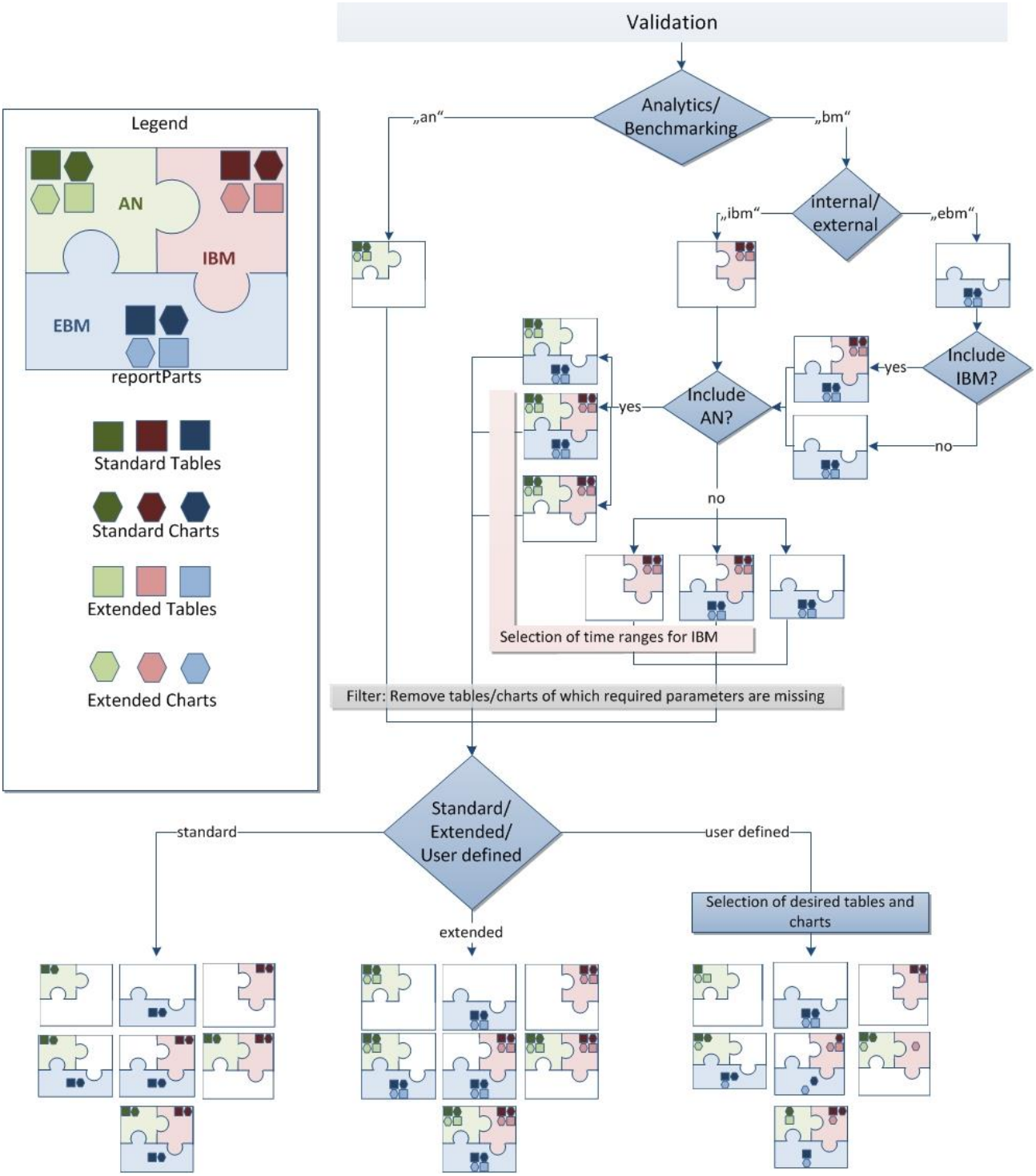


Figure 31: Illustration of how the *reportParts* variable was changing dependent on the decisions of the user

Selection of the Internal Benchmarking Time Ranges

If the current path was “ibm”, or “ebm” with included internal benchmarking (ibm = internal benchmarking, ebm = external benchmarking), the maximal and minimal surgery date of all entries of the current centre was calculated and internal benchmarking configurations had to be done. Further the path was set to ”ibmConfig”. If the path is neither “ibm” nor “ebm”, the *proceed()* function was called.

The internal benchmarking could be done by using a *datepicker*. A *datepicker* was a bootstrap element showing a calendar of which only dates lying between the minimum and maximum surgery date of the current centre, were enabled. The user could then select one date after another to select the time ranges. The whole process is illustrated in Figure 32. If two time ranges were intersecting, the selection would be invalid and a warning would be shown to the user. It was also possible to delete already selected time ranges, by clicking on the red X next to the time range (see Figure 32, part 3). The selected dates were added to the *ibmRange* $scope variable by the *setIbmRange(Date)* function and removed by the *removeDate()* function.

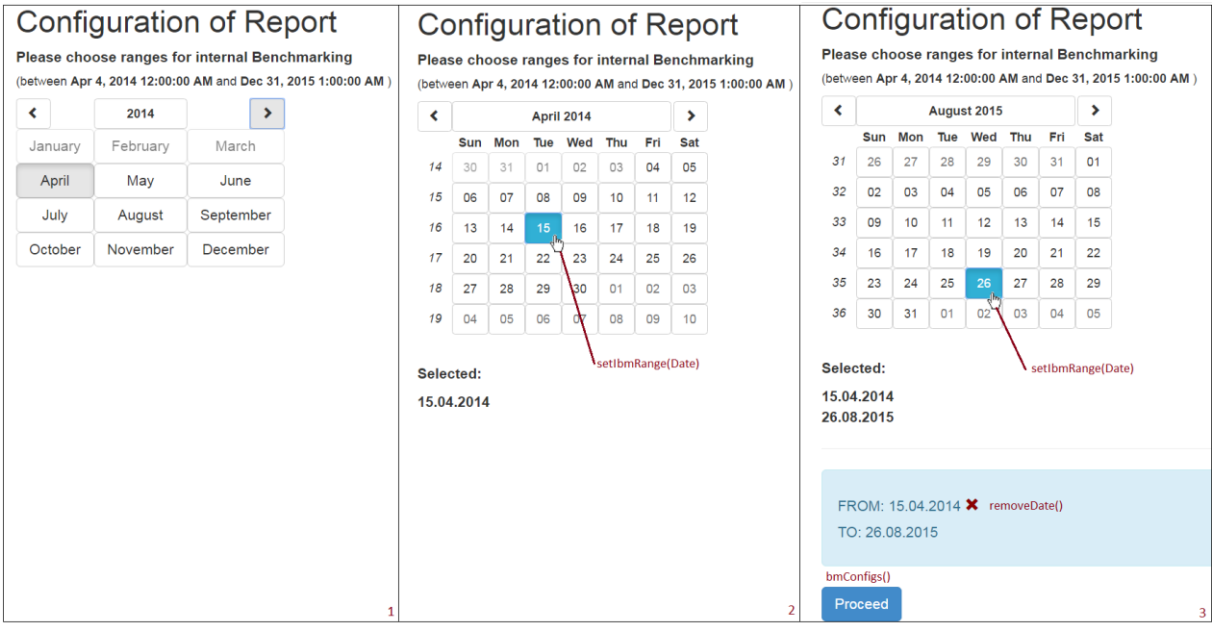


Figure 32: Configuration screen for selecting up to a maximum of three time ranges for internal benchmarking.

When the user confirmed his or her selection by clicking the *Proceed* button, the bmConfigs() function was called again, but now the current path was set to “ibmConfigs” and accordingly only the *proceed()* function got called.

Aggregation of Data Entries for *summary*

In the *proceed()* function all tables and charts that were not chosen in the configuration step, in the first part of Figure 30, were deleted from the *reportParts* variable.
Additionally two more filters were applied:

- The *separationRequired* filter was applied on the *reportParts* variable, returning all defined separation parameters of every category in the sub variable with the corresponding category name. In the sub variable separationTotal the separation parameters of all categories were contained. These results were saved in the $scope variable *requirements.*
- The *summarise* filter was applied on the *dataToReport* variable with 4 more filters and the requirements variable as transfer parameters.

Separation parameters, like gender or indication, were defining subcategories to which the different data entries could belong. For example the separation parameter of the *transfusion index* table was *indication*. This meant that the transfusion index in the table was calculated over every subcategory defined by the different indications. If there was a second separation parameter, the subcategories were all permutations of combining the two parameters. For example having gender and indication as separation parameters would result in two subcategories, male and female, for each indication (See Figure 33 and Figure 34).

Both filters differentiated between three classes: av for averaging, yn for yes-no decisions and cat for categorisation. In the configuration file, every parameter of the *needed_params* had an assigned class. Depending on this class, different aggregation calculations were made for this parameter. It was also possible that in the *needed_params* the same parameter was contained twice but with different classes (See Figure 29).
For the different aggregation calculations the *summarise* filter applied one of the following filters depending on the class, with the parameter name and the parameter value as transfer parameters:

- av: The *appendAverage* filter was applied: For every subcategory *sum* (sum of values, except values that were Not A Number (*NaN*) and *count* (number of values, without values that were *NaN*) was calculated.
- yn: The *appendYn* filter was applied: For every subcategory ysum (sum of all values >0, meaning that they are “yes”) and yes(count of all values >0) and count (count of all values) of the values was calculated. Additionally the ycount (count of values >0 for each of the separation categories) and ycountTotalCentre(count of values >0 for the whole centre) was calculated. This was needed for the calculation of percentages

of the subcategories from superior subsets. The gender/indication example for just one indication (ECV) would accordingly appear as shown in Figure 33.

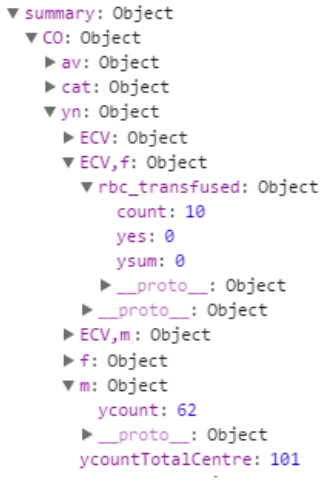


Figure 33: Example of the *yn* category of the *summary* variable, shown by a screenshot of the browser console.

This would enable the calculation of the percentage of women from all surgeries with indication ECV.

- cat: The *appendCatList* filter was applied: This filter created a list, with one entry for each value containing its number of occurrences as in Figure 34.

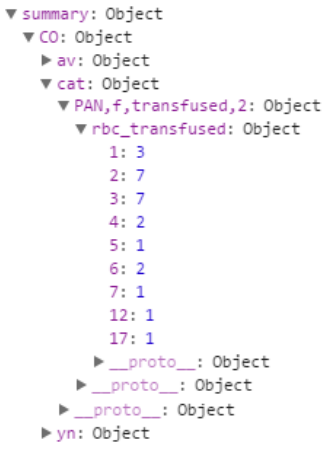


Figure 34: Example of the *cat* category of the *summary* variable, shown by a screenshot of the browser console.

If the classification according to different internal benchmarking phases (ibm_date) was required as a separation parameter, also the *appendIbmFilter* filter was applied before applying all the other filters. This filter added the *ibm_date* parameter containing the values: “phase 1”, “phase 2”, “phase 3” or “no phase” , depending on the time range - selected by the user as previously described -in which the surgery date of the entry was located. The resulting summary, saved in the summary $scope variable, was the basis for all tables and charts.

Generation of the Tables

If the analytics part was not included in the report, the *toReport()* function has been called at the end of the *proceed()* function. Otherwise the analytics configuration was displayed, where the user had the option to modify some configurations concerning the resulting tables and charts directly and interactively.

Figure 37 shows excerpts of the analytics part configuration screen, including interactive configuration possibilities, marked by the red frames. The most common table structure, which included the *demographics* and *preoperative haemoglobin value* table structures in Figure 37, was created by the *stdTable* directive. The transfer parameters were:

- the configuration information for the desired table from the *reportParts* variable (See Figure 29),
- the *summary* variable, the current report part (‘an’, ‘ibm’ or ‘ebm’),
- if the html tag should be replaced by the template table of the directive or if the directive was just used for calculating the required parameters (replace),
- the current centre (centre of the user),
- the separation parameters and
- the table, in which calculated parameters should have been saved.

A $scope.watch function at the beginning of the directive was executing the directive again if there were changes in the configuration file. This was necessary because of the interactive elements. If the directive was called from the “ebm” report part, the calculations were done for all centres contained in the summary file. Otherwise only calculations for the current centre were performed. Further all ycounts -if available- were extracted to a certain variable. As can be seen in the above *summary* examples, the subcategories consist of a joined string of separation parameters. These separation parameters were split and those parameters which were not required according to the configuration file were deleted. Ensuing the parameters were again joined and those subcategories which were equal after removing attributes, not required for this table, were summed up. Based on values already calculated for the different categories, the rest of the required parameters were determined at this point, again depending on the categories:

- av: already calculated: *sum, count*; currently calculated: *av* (average)
- yn: already calculated: *yes, ysum, yn*, *count*; currently calculated: *yav* (average of positive entries), *yperc* (percentage of subcategory from positive entries of superior category*), yperc_tot* (percentage of subcategory from all positive entries of the centre)

- cat: already calculated: list of values with corresponding *counts*; currently calculated: *catcount* (count of all categories), *percentage* (percentage of each category from all categories)

Afterwards all parameters were rounded to two decimal places and saved in the table which was passed as transfer parameter. According to the configuration file (See Figure 29), the table was created. If the standard structure for tables was needed for the desired table the replace parameter of the directive could be set to *true* and the table was inserted using the directives template. This template is shown in Figure 35.

```
<div ng-if="repl()==true">
   <div ng-if="path() != \'ebm\'">
      <h5><b>{{config.title}} </b></h5><table class="analytics-table">
         <thead>
            <tr>
               <th ng-repeat="(sep, sepVal) in config.seperation">
                  {{sepval}}
               </th>
               <th ng-repeat="(sep, sepVal) in config.needed_params" ng-if="sepVal != \'sep\'">{{sep}}</th>
               <th ng-repeat="(sep, sepVal) in config.needed_params" ng-if=" sepVal != \'sep\' && config.extended.stdd  == true">standard deviation {{sep}}</th>
            </tr>
         </thead>
         <tbody>
            <tr ng-repeat="(key,value) in table[ownCentre().centre_number]">
               <th ng-repeat="part in key.split(\',\')">
                  {{part}}
               </th>
               <td ng-repeat="(neededParam, neededParamVal) in config.needed_params" ng-if="neededParamVal != \'sep\'">
                  {{ value[neededParam][neededParamVal]}}
               </td>
               <td ng-repeat="(neededParam, neededParamVal) in config.needed_params" ng-if="config.extended.stdd == true && neededParamVal != \'sep\' ">
                  {{value[neededParam].stdd }}
               </td>
            </tr>
         </tbody>
      </table>
   </div>
</div>
```

Figure 35: Directive template used for creating standard tables.

The header was created by first iterating over the separation parameters and latterly over the *needed_params* of the configuration file. The body of the table repeated over the table extracting the parameters using again iterations over the *nedded_params* of the configuration file and using them as keywords (See Figure 29).
If standard deviation is required another directive was called for the calculation.

Generation of Charts

For every kind of chart a directive was created. These directives were using the data which was first calculated by the *stdTable* directive.

The data was prepared in a way that it satisfied the requirements for using highcharts, e.g. defining which parameters were displayed on the x-axis, or which entries were belonging to one series etc. An example of using a highcharts function was shown in Figure 36. The parameters *title, value* and *series* are variables prepared from the directive. An example of resulting charts can be seen in Figure 37.

```
$(function () {
$('#' + outId).highcharts({
    chart: {
        type: 'column'
    },
    title: {
        text: title
    },
    xAxis: {
        categories: categories//,
   // crosshair: true
    },
    yAxis: {
        min: 0,
        title: {
            text: value
        }
    },
    tooltip: {
        animation: true,
        enabled: true
    },
    plotOptions: {
        column: {
            pointPadding: 0.2,
            borderWidth: 0,

        },

        enableMouseTracking: false
    },
    series: series
});
});
```

Figure 36: Highcharts example

Interactive Configuration Elements

The interactive configuration parts can be seen in Figure 37 and are marked with a red frame. Beside the *needed_params* there were also extended parameters listed in the configuration file (See Figure 29). This means they were not required for constructing the table, but if they were available, they could be included in the analytics part configurations. The *stdTable* directive calculated the aggregation values for extended and needed parameters. Under the demographics table in part 1 of Figure 37 all available extended parameters could be chosen by clicking on the corresponding checkboxes. If activated, the parameter has been added to the *needed_params* of this table and accordingly added to the table. As already mentioned when clicking on the standard deviation checkbox, which could also be seen in part 1 of Figure 37, the *stdd* parameter within the extended parameters was set to *true.* Consequently the *stdDeviation* directive was called, which calculated the standard deviation and added the value to the corresponding table. The third interactive part was changing the Hb category thresholds. As can be seen in part 1 of Figure 37, the *hbHadChanged()* was executed, which called the summarise function again and consequently all calculations were repeated. The last interactive feature was the addition and removal of histogram classes. The corresponding *addHistClass()* and *removeHistClass()* functions were changing the configuration file of affected elements and because the *stdTable* directive had a $scope.watch function on changes to the configuration files, the directive was again executed. Accordingly also the calculations of the directive were repeated.

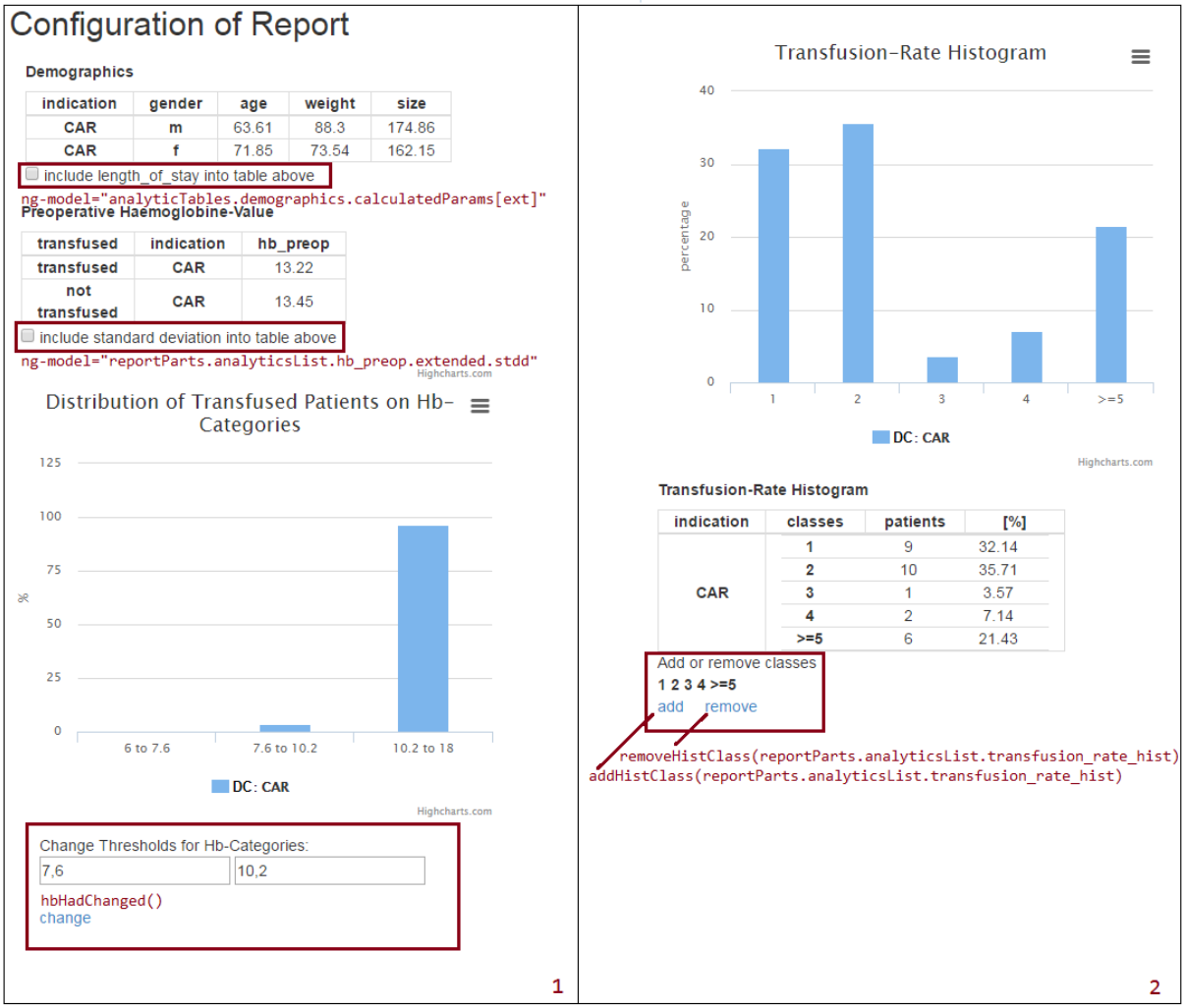


Figure 37: Excerpts of the analytics part configuration screen, including interactive configuration possibilities.

To finish the configuration steps, a Proceed button *at* the end of the analytics part configuration could be clicked and the *toReport* function changed the location path to "/report".

3.2.2.4 Report

If external benchmarking should be included in the report, the user was required to select maximal two centres with which he or she wanted to compare with their own centre. If no other centre was selected, calculations were only shown for that centre. Therefore a bubble chart of all centres with indication as separation parameter (*bubble* directive) was displayed (see Figure 38). The chart had the transfusion rate on the x-axis, the transfusion index on the y-axis and bubble sizes were correlating with the number of patients.

When the user pressed *confirm*, the *ebmSelectionDone()* function has been called. The function deleted the centres which were not required from the summary and created another $scope variable called *summaryForAll.* At the end of a report containing external

benchmarking, charts showing the transfusion rate and index from all centres were included. The *summaryForAll* variable was required for these charts. Then a preview of the report was shown.

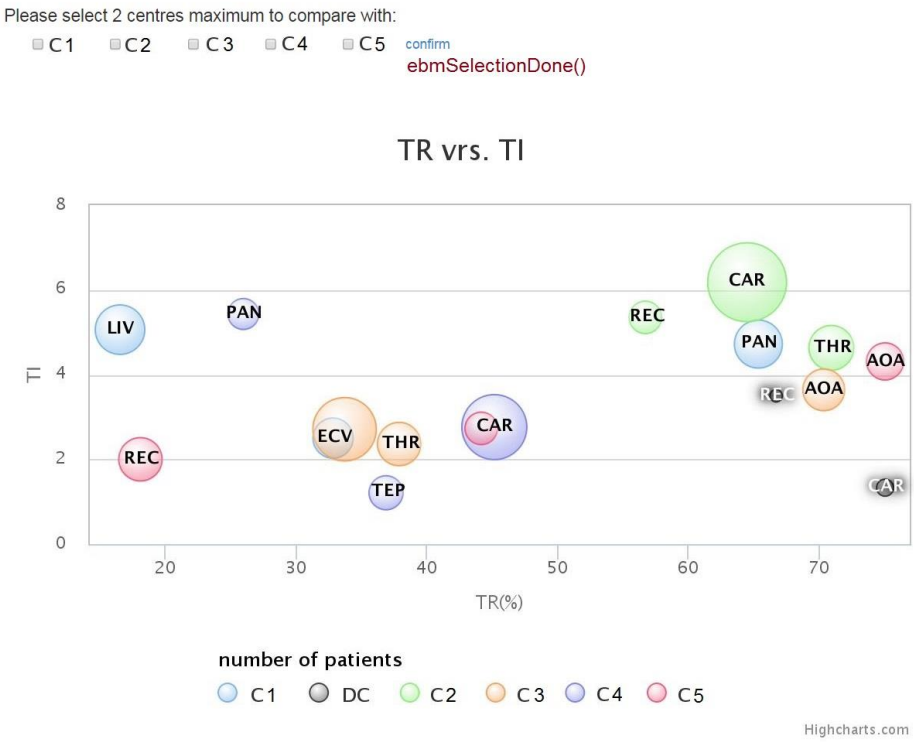


Figure 38: Selection screen for choosing two centres for comparison (C1-C5 stands for centre 1 to 5 and DC is the abbreviation of the user's centre, in this case "Demo-Centre")

The figures and charts were created analogously to the analytics part configuration, using the corresponding directives. One new element, which was not included in the analytics part, was a boxplot showing the interquartile ranges for transfused RBC units. An example for such a boxplot can be seen in Figure 39.

For creating a boxplot, the *interquartile* directive had to be utilised. This directive received the *summary* as data transfer parameter. After summing up the subcategories to the needed separation level given by the configuration file, the directive was calculating the interquartile range, minimum and maximum by using the list for a parameter of category *cat* (See Figure 34).

Zero was excluded for all calculations inside the *interquartile* directive. The minimum and maximum were determined by finding the smallest or greatest category of the list. Afterwards the index of median, lower quartile and upper quartile were determined using Formula (4), (5) and (6):

$$medianIndex = \frac{Sum_{all\ entries}+1}{2} \quad (4)$$

$$lQuartileIndex = \frac{Sum_{all\ entries}+1}{4} \quad (5)$$

$$uQuartileIndex = 3 \cdot \frac{Sum_{all\ entries}+1}{4} \quad (6)$$

Then it was iterated over the categories (See Figure 34) and the occurrences were summed up by using a counter. If the counter became greater than one of the indices, the category at this iteration step was the result for the parameter corresponding to the index. If the counter was equal to the down rounded index, the result was calculated with Formula (7).

$$X = category_i + [(Index_x - floor(Index_x)) \cdot (category_{i+1} - category_i)] \quad (7)$$

The *X* in Formula (7) stands for median, lower quartile or upper quartile.
The result of a resulting boxplot can be seen in Figure 39.

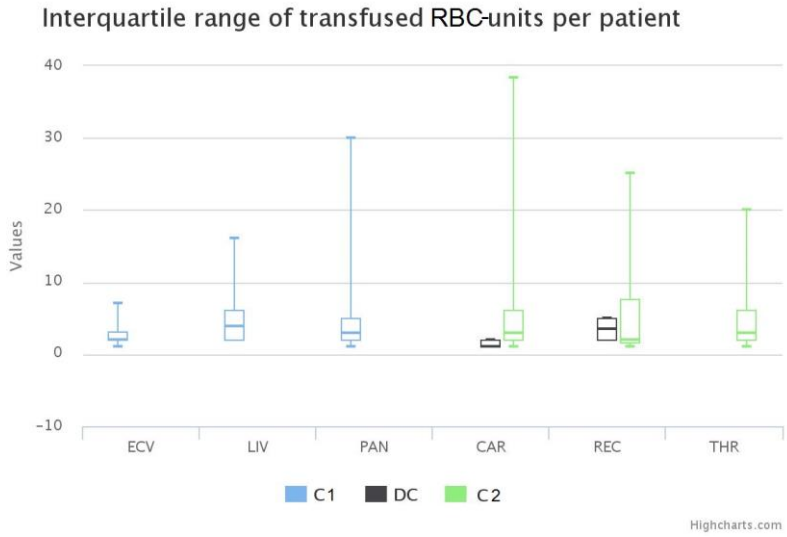


Figure 39: Box plot showing interquartile ranges of transfused RBC units for every indication of the three centres (DC stands for Demo-centre, C1 and C2 are the centres for comparison)

Finally the user had the possibility to create a PDF from the created PBM report. Therefore, the *create PDF* button had to be clicked, which called the *createPDF()* function.
Because this PDF creation needed between approximately 5 and 10 seconds, a progress bar was displayed to the user.
The *createPDF()* function first replaced all SVG (scalable vector graphic) elements with a canvas. This could be done using the *canvg* library and the *exports* library of highcharts. Furthermore, every html element, shown in the preview was also converted into a canvas and one after another was added to the PDF. When the PDF creation was done, the SVG elements were reconverted. An example of a resulting PDF report can be in the appendix.

3.2.3 Verification of the PBM-ABS App/ Feasibility Test

The verification of the PBM-ABS app was performed in three independent steps.

1. Verification of the filter conditions

A test data set containing possible data failures, which can be seen in Figure 27 was used to check, whether the filter settings work correctly. Following failures were included:

1. wrong field name
2. 1/0 instead of yes/no
3. Wrong date format
4. Weight, height and Hb values out of plausibility range
5. Missing values
6. Point used for separation instead of comma
7. Partly capital letters

The results showed that 100% of the failures, which should be identified for being incorrect were detected (1-5) and 100% of the failures, which should be ignored were classified for being correct (6,7).

2. Verification of calculation and report generation using a fictional data set

A fictional dataset including 7 entries for a “Test-Centre” was analysed by the PBM-ABS app. One entry was directly uploaded to the database and the other 6 were used as input file of the “Test-Centre”. To test the impact of missing values, one mandatory value of the core data set (preoperative Hb) was removed. The results for comparison were calculated with Microsoft Excel®. The comparison resulted in 100% equivalence. All results are shown in the Appendix.

3. Verification of calculation and report generation using real data

For testing the feasibility of the app, test reports for one centres of the EU-PBM project were created using the PBM-ABS app. Therefore, the original data of 4 centres was directly loaded into the PBM database and data of one selected “Demo-Centre” was uploaded using a csv file. Overall 1485 cases were available, however, not all of these single cases fulfilled the criteria for the core data set and consequently rows with missing values had been ignored. Overall 1102 values (e.g. blood loss, transfusion rate, transfusion index, etc.) were calculated which were compared to the original results of the Microsoft® Access queries used for EU-PBM project. The comparison showed equal results for 97,37% of the resulting values. The remaining 2,63% showed deviations below 1% which were caused because interim sums of the implemented java script were rounded for the nearest two decimals.

4 Discussion

4.1 Showcase for Retrospective Data Extraction from Existing Data Sources in the AKH Wien

4.1.1 Data Extraction from the PDMS

Extracting data from existing sources is part of the “gold standard” model for benchmarking in transfusion medicine of Apelseth et al [3]. According to Redman an operational repository usually contains 1 to 5% incorrect values [55]. Appropriate information concerning the data quality of hospital information systems or medical health records could not be found in relevant literature. However, this explains minor deviations when comparing the query results of the different sources (e.g database of the Department of Cardiothoracic and Vascular Anaesthesiology and the PDMS data). It should be considered that in an ICU environment, the optimal circumstances for comprehensive documentation are not always given, where the patients are usually in a critical condition and physicians and nurses are sometimes under high pressure. For example measuring the exact height or weight of the patient might be challenging.

During the design of the query, including extraction of specific interventions a patient has been undertaken (HTEP, PAN, CAGB, etc.), the problem became obvious that this information was not consequently documented within the available data sources. Although the primary diagnosis is well recorded and some diagnoses are indicating specific interventions (e.g. diagnosis “coxarthrosis” usually leads to total hip replacement), however this is not always the case. Other possible sources of information would be the discharge report and the accounting documentation of the hospital. Nevertheless, these documents were not part of the available databases.

4.1.2 Merging Data from Internal Data Sources

Referring to the results in 3.1.3 concerning correct merging of the sources Microsoft® Access provided a very powerful and flexible tool. For the definition and implementation of complex queries it is recommended to use the SQL-view, because the graphical user interface does not support transparent handling of logical expressions. To avoid unexpected errors or incorrect results, a hierarchical priorisation of parameters, which might be available from more than one source ore multiple times inside the same source, should be defined.

For example, the total number of transfused RBC units concerning cardiac surgeries was documented in the transfusion database (TBD) and the Cardio-db. Reasons for differences are incomplete and erroneous documentation on both sides. Even if the TDB documentation basically provides a very high level of completeness regarding the number of units delivered, the transfusion status might not be recorded permanently. On the other hand at the cardiac surgery ward documentation errors might happen because of input errors or because RBC units might be disposed and not transfused.

4.2 PBM-ABS App

During the conceptual phase of the PBM-ABS app, intensive discussions with physicians and PBM experts were necessary for the definition of the use cases and basic requirements. Typically for software prototypes the use cases and requirements changed during the implementation process, which led to a software structure that might have to be redesigned after receiving feedback from first users.

However, AngularJS provided a powerful technology for the agile development of the PBM-ABS tool that only requires standard internet browser at the hospitals. No installation at the client computer is necessary and therefore a Zero-footprint application could be provided. The App supports flexible utilisation because configuration of the tables and charts can be adapted to specific needs (e.g. adding subcategories, title, included parameters) by modifying the configuration file (Figure 29). The App itself provides high usability because the user is guided through a clear step-by-step workflow and is able to generate comprehensive benchmark reports within minimal efforts.

The modular design allows flexible extension concerning additional tables and charts with parameters which are already in the core dataset but also for many other features (See 4.3). Additional analytics for supplementary fields might be integrated based on specific user requirements.

To increase motivation for users at the hospitals, the analytics part of the PBM-ABS app provides immediate results without the necessity to upload data to an external server. The generated report supports self-assessment e.g. to monitor the effectiveness of implemented interventions. However, for comparison with other centres data upload and official commitment for participating at external benchmarking is required.

4.3 Future Work

In the near future it is planned to use the PBM-ABS app as demonstrational prototype for hospitals which are interested in internal and/or external PBM benchmarking.

Before first real world utilisation the following issues should be analysed and resolved:

- The PBM-ABS app should be integrated to a web portal with user management and authentication to replace the *authentication information template* of the prototype. This service should provide enrolment information for users and centres.
- Terms of use and disclaimer have to be specified.
- Data anonymisation or at least pseudonymisation should be implemented to comply with data protection law.
- For quality management reasons traceable documentation of reports and audit trials should be implemented.
- A user interfaces for service administration would support the maintenance of the configuration and filter specifications, since this can only be done on development level.
- A remarkable issue was that the open source jspdf library has an internal bug. Because of this bug, the pdf creation sometimes aborts, which probably is caused by content size. The *createPdf()* function adds canvases (screen shots) to the report. By using bootstrap the representation of the report preview was dependent on the screen size. When using high resolution screens, it occurred that the pdf creation ceased to work.
- Further results and features may be added based on the feedback of the demo users.

4.3.1 Future Extension Possibilities

For supporting the change management process within the hospitals and to increase the attractiveness of the PBM-ABS service the following additional features should be evaluated:

- Calculation of the expected savings when benchmarking the own status with “best-in-class” centres.
- Calculation of the expected improvements by using specific PBM related measures (e.g. potential annual reduction of RBC units by implementing physiological transfusion triggers).
- Calculation of the potential risk and expected need for transfusion prediction of required RBC units for elective surgeries by using the Mercuriali algorithm, which feasibility was shown in [56]. This would allow for more precise ordering and planning of blood bank storage.
- Extending the core data set with information about the patient’s outcome and implement additional analytics for outcomes research.

5 References

[1] E. Anthes, “Evidence-based medicine: Save blood, save lives,” *Nature News,* 31 march 2015.

[2] H. Gombotz, K. Zacharowski and D. R. Spahn, Pateint Blood Management, Thieme, 2015, pp. 2-10.

[3] T. Oveland Apelseth, L. Molnar, E. Arnold and N. M. Heddle, "Bechmarking: Applications to Transfusion Medicine," *Transfusion medicine reviews,* vol. 26, no. 4, pp. 321-332, 2012.

[4] Austrian Institute of Technology, "EU-PBM Leaflet," May 2014. [Online]. Available: www.europe-pbm.eu. [Accessed 14 March 2016].

[5] P. Kastner, N. Breznik, H. Gombotz, A. Hofmann and G. Schreier, "Implementation and Validation of a Conceptual Benchmarking Framework for Patient Blood Management," *EHealth2015–Health Informatics Meets EHealth: Innovative Health Perspectives: Personalized Health,* p. 190, 2015.

[6] A. Ettorchi-Tardy, M. Levif und P. Michel, „Benchmarking: a method for continuous quality improvement in health,“ *Healthcare policy,* Bd. 4, Nr. 7, p. e101, 2012.

[7] European Commission, "An EU-wide overview of the market of blood, blood components and plasma derivatives focusing on their availability for patients," April 2015. [Online]. Available: http://ec.europa.eu/health/blood_tissues_organs. [Accessed March 2016].

[8] M. van Kraaij, "Patient Blood Management in the Netherlands:Between practice and evidence," Sanquin blood supply, Radboud University Medical Center, Nijmegen, 2015.

[9] A. Shander, A. Hofmann, O. Sherri, O. M. Theusinger, H. Gombotz and D. R. Spahn, "Activity-based costs of blood transfusions in surgical patients at four hospitals," *Transfusion,* vol. 50, no. 4, p. 753–765, April 2010.

[10] R. L. Barty, K. Gagliardi, W. Owens, D. Lauzon, S. Scheuermann, Y. Liu, G. Wang, M. Pai und N. M. Heddle, „A benchmarking program to reduce red blood cell outdating: implementation, evaluation, and a conceptual framework,“ *Transfusion,* Bd. 55, Nr. 7, pp. 1621-1627, 2015.

[11] A. Winter, E. Ammenwerth, O. Bott, B. Brigl, A. Buchauer, S. Gräber, A. Grant, A. Häber, W. Hasselbring, R. Haux, A. Heinrich, H. Janssen, I. Kock, O.-S. Penger, H.-U. Prokosch, A. Terstappen und A. Winter, „Strategic information management plans: the basis for systematic information management in hospitals,“ *Internaltional Journal of Medical Informatics,* Bd. 2, Nr. 64, pp. 99-109, 2001.

[12] E. Ammenswerth, R. Haux, P. Knaup-Gregori und A. Winter, IT-Projektmanagement im Gesundheitswesen: Lehrbuch und Projektleitfaden Taktisches Management von Informationssystemen, 2 Hrsg., Schattauer Verlag, 2014.

[13] S. Alshawi, F. Missi und T. Eldabi, „Healthcare information management: the integration of patients' data,“ *Logistics Information Management,* Bd. 3/4, Nr. 16, pp. 286-295, 2003.

[14] H. Gombotz, P. H. Rehak, A. Shander and A. Hofmann, “Blood use in elective surgery: the Austrian benchmark study,” *Transfusion,* vol. 47, no. 8, pp. 1468-1480, 2007.

[15] H. Gombotz, P. Rehak and A. Hofmann, "Projekt „Fortsetzung der Studie betreffend Maßnahmen zur Optimierung des Verbrauchs von Blutkomponenten in fachlicher und inhaltlicher Sicht" Modul 1: "Fortsetzung und Erweiterung der Benchmark-Analyse"," Österreichische Bundesgesundheitskommission, Wien, 2009.

[16] P. G. D. Group. [Online]. Available: http://www.postgresql.org/. [Accessed may 2016].

[17] “Zope Foundation,” [Online]. Available: http://www.zope.org/. [Accessed may 2016].

[18] P. S. Foundation. [Online]. Available: https://www.python.org/psf/. [Accessed may 2016].

[19] Zope, [Online]. Available: http://zope2.zope.org/. [Accessed april 2016].

[20] “HIMSS Europe,” [Online]. Available: http://himss.eu/emram. [Accessed april 2016].

[21] “Philips IntelliSpace critical care and anesthesia,” [Online]. Available: http://www.philips.at/healthcare/product/HCNOCTN332/intellispace-critical-care-and-anesthesia. [Accessed may 2016].

[22] “Cemsiis AKIM,” [Online]. Available: http://cemsiis-akim.meduniwien.ac.at/allgemeines/. [Accessed may 2016].

[23] Medizinische Universität Wien; AKH EDV; Allgemeines Krankenhaus der Stadt Wien, "Allgemeine AKIM-Projektinfo," 2011.

[24] T. Wrba, *Comprehensive Cancer Center Vienna-Tumordatenbank in der AKIM RDA-Plattform,* Zentrum für Medizinische Statistik, Informatik und Intelligente Systeme, 2011.

[25] Philips Health Care, *Enhanced care for your acute patients,* Netherlands: Philips, 2015.

[26] “Microsoft,” [Online]. Available: www.microsoft.com/. [Accessed may 2016].

[27] JavaScript, may 2016. [Online]. Available: www.javascript.com.

[28] “ObjectiveIT,” [Online]. Available: www.objectiveit.com/blog/the-advantages-and-disadvantages-of-web-apps. [Accessed april 2016].

[29] “Visual Studio,” [Online]. Available: www.visualstudio.com. [Accessed may 2016].

[30] Visual Studio, [Online]. Available: visualstudio.com. [Accessed april 2016].

[31] B. Dayley and D. Brendan, AngularJS, JavaScript and JQuery All in One, Sams, 2016.

[32] jQuery. [Online]. Available: https://jquery.com/. [Accessed may 2016].

[33] “AngularJS,” [Online]. Available: https://angularjs.org/. [Accessed may 2016].

[34] B. Green und S. Seshadri, AngularJS, O'Reilly Media, Inc., 2013.

[35] “Google,” [Online]. Available: https://www.google.com/chrome. [Accessed may 2016].

[36] “Mozilla Firefox,” [Online]. Available: https://www.mozilla.org/de/firefox/. [Accessed may 2016].

[37] “Node.js,” [Online]. Available: https://nodejs.org/en/. [Accessed may 2016].

[38] “MySQL,” [Online]. Available: https://www.mysql.com/. [Accessed may 2016].

[39] R. Cattell, “Scalable SQL and NoSQL data stores,” *ACM SIGMOD Record,* vol. 4, no. 39, pp. 12-27, 2011.

[40] H. Andrade, B. Gedik, M. J. Hirzel, R. J. Soule, H. Wang, K.-L. Wu und Q. Zou, „Proxying open database connectivity (ODBC) calls“. Patent US Patent 8,321,443, november 2012.

[41] Microsoft Office Support, [Online]. Available: support.office.com. [Accessed april 2016].

[42] H. Burnus, “Datenbanksystem Microsoft® Access,” in *Datenbankentwicklung in IT-Berufen: eine praktisch orientierte Einführung mit MS Access und MySQL*, Springer Science & Business Media, 2007, pp. 53-80.

[43] J. L. Harrington, “Introduction to SQL,” in *SQL Clearly Explained*, Elsevier Sience, 2003.

[44] E. Schicker, Datenbanken und SQL: Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL, 17 ed., Springer-Verlag, 2014.

[45] “Edureka,” [Online]. Available: www.edureka.co/angular-jsSlide. [Accessed april 2016].

[46] S. Seshadri and B. Green, AngularJS: Up and Running: Enhanced Productivity with Structured Web Apps, O'Reilly Media, Inc., 2014.

[47] “Bootstrap,” [Online]. Available: http://getbootstrap.com/. [Accessed may 2016].

[48] J. Fielding, Beginning Responsive Web Design with HTML5 and CSS3, Apress, 2014.

[49] “Highcharts,” [Online]. Available: http://www.highcharts.com/. [Accessed may 2016].

[50] B. Shahid, Highcharts Essentials, Packt Publishing Ltd, 2014.

[51] “jsPDF on GitHub,” [Online]. Available: https://github.com/MrRio/jsPDF. [Accessed may 2016].

[52] “Niklas von Hertzen on GitHUb,” may 2016. [Online]. Available: https://github.com/niklasvh.

[53] “Gabe Lerner on GitHub,” [Online]. Available: https://github.com/gabelerner. [Accessed may 2016].

[54] “Medical Dictionary,” [Online]. Available: http://medical-dictionary.thefreedictionary.com/. [Accessed april 2016].

[55] T. C. Redman, Data quality for the information age (The Artech House computer science library), 1996.

[56] D. Hayn, K. Kreiner, P. Kastner, N. Breznik, A. Hofmann, H. Gombotz and G. Schreier, “Data Driven Methods for Predicting Blood Transfusion Needs in Elective Surgery,” *eHealth2016- Health Informatics meets eHealth,* 2016.

6 Appendix

To demonstrate the feasibility of the PBM-ABS-Service original data from the EU-PBM data were anonymised.

Appendix-Table 1 shows a small fictional dataset including 7 entries for a demo centre, which was analysed by the PBM-ABS-App. One entry was uploaded to the database and the other 6 were used as input file. Using such a small dataset makes it easier to establish if the results are correct, since the original data included several hundred entries per centre.

Appendix-Table 1: Fictional data for Graz

indication	gender	age	weight	size	hb_preop	hb_end	surgery_date	rbc_transfused	hb_5post	hb_nadir	length_of_stay
CAR	m	57	79	174	16,4	10,2	09.04.2013 00:00	1,00	12,3	10,2	15
CAR	f	63	65	170	15,3	9,3	10.07.2014 00:00	0,00	8,3	8,3	7
CAR	f	61	80	168	7,2	7	12.11.2015 00:00	1,00	10,2	7,2	4
CAR	m	75	85	170		10,2	04.04.2014	2,00	11,3	10,2	5
REC	f	80	77	180	10,2	6,8	10.05.2013 00:00	5,00	7,9	6,8	15
REC	m	52	70	185	10,5	10,2	28.06.2014 00:00	2,00	6,3	6,3	5
REC	m	57	79	170	9,6	11,2	01.01.2015 00:00	0,00	8,3	8,2	10

Appendix pages A2 to A9 are showing the imported PDF-pages of the Demo-centre report produced by the PBM-ABS-App.

PBM-Report for Demo-centre

May 13, 2016 10:02:06 AM

Demographics

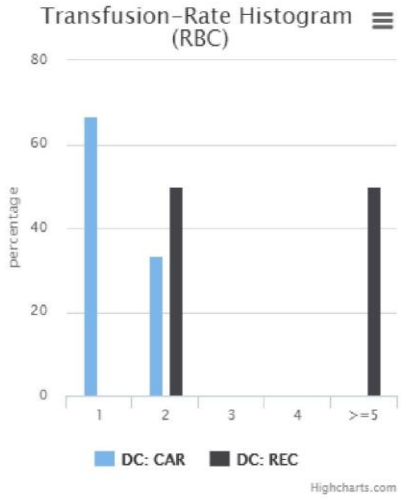
indication	gender	age	weight	size	length_of_stay
CAR	m	66	82	172	10
CAR	f	62	72.5	169	5.5
REC	f	80	77	180	15
REC	m	54.5	74.5	177.5	7.5

Transfusion-Rate (RBC)

indication	all patients	patients transfused	TR
CAR	4	3	75
REC	3	2	66.67

Transfusion-Index (RBC)

indication	patients transfused	rbc-units transfused	TI
CAR	3	4	1.33
REC	2	7	3.5



Transfusion-Rate Histogram (RBC)

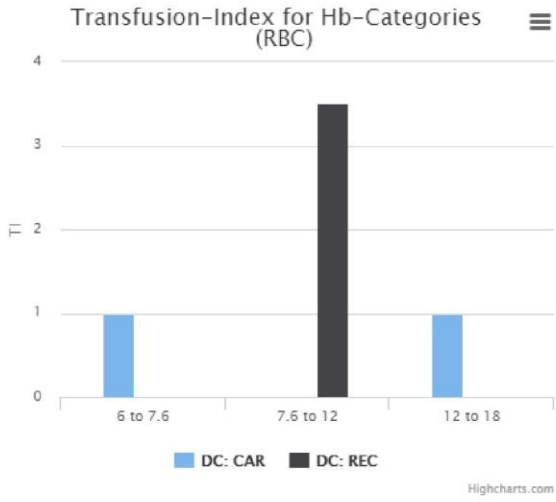
indication	classes	patients	[%]
CAR	1	2	66.67
	2	1	33.33
	3		
	4		
	>=5		
REC	1		
	2	1	50
	3		
	4		
	>=5	1	50

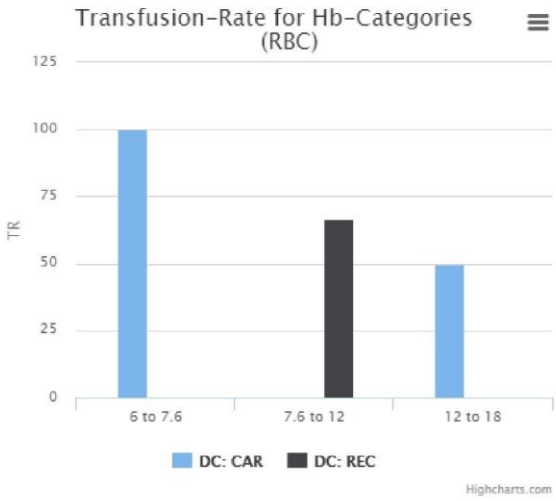
Preoperative Haemoglobine-Value

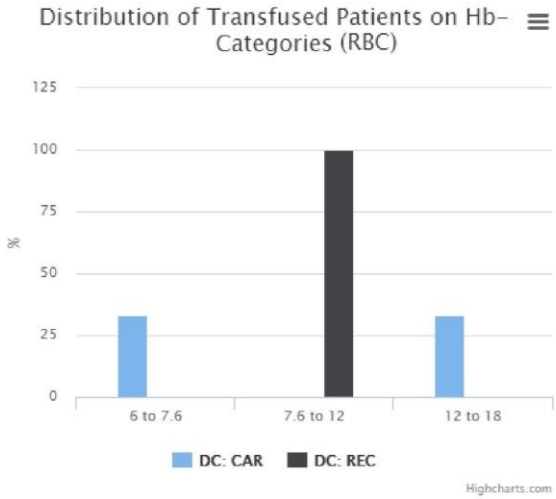
transfused	indication	mean Hb-preop	standard deviation
transfused	CAR	11.8	6.51
not transfused	CAR	15.3	0
transfused	REC	10.35	0.21
not transfused	REC	9.6	0

Haemoglobine-Value on 5th Postoperative Day

transfused	indication	mean Hb post-day5	standard deviation
transfused	CAR	11.27	1.05
not transfused	CAR	8.3	0
transfused	REC	7.1	1.13
not transfused	REC	8.3	0







Internal Benchmarking

phase 1:

from: 09.04.2013 **to:** 21.07.2014

phase 2:

from: 22.07.2014 **to:** 12.11.2015

Demographics

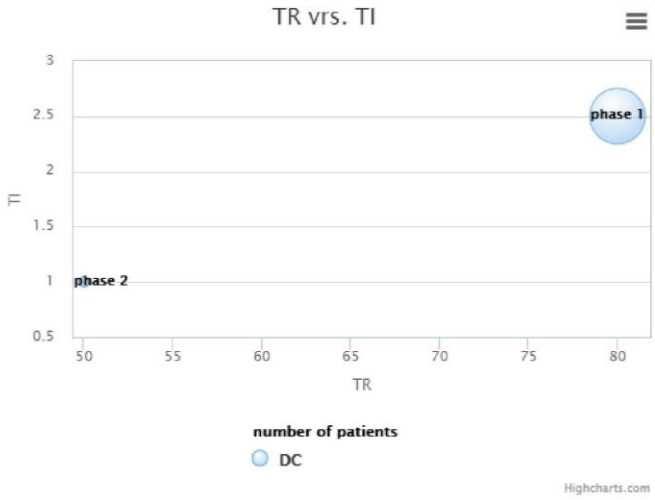
indication	ibm_date	age	weight	size	patients
CAR	phase 1	65	76.33	171.33	3
CAR	phase 2	61	80	168	1
REC	phase 1	66	73.5	182.5	2
REC	phase 2	57	79	170	1

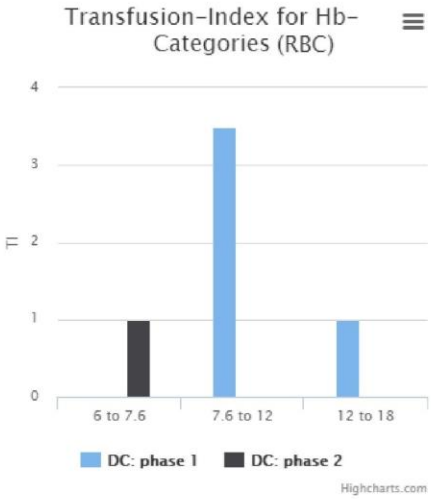
Hb-Values

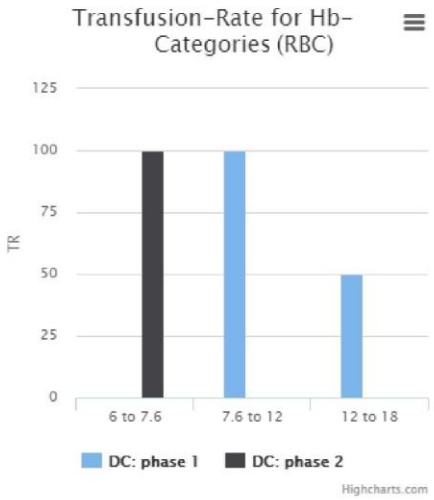
indication	ibm_date	hb_preop	hb_5post	Std hb_preop	Std hb_5post
CAR	phase 1	15.85	10.63	0.78	2.08
CAR	phase 2	7.2	10.2	0	0
REC	phase 1	10.35	7.1	0.21	1.13
REC	phase 2	9.6	8.3	0	0

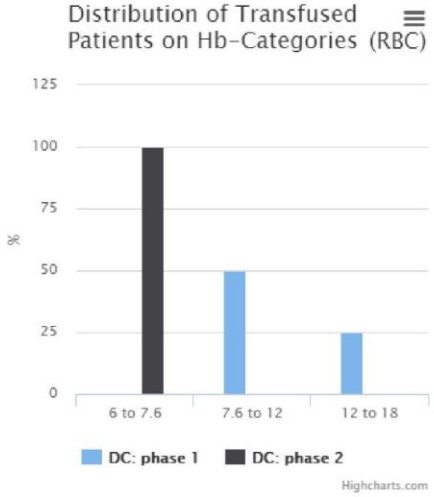
Transfusion-Parameters (RBC)

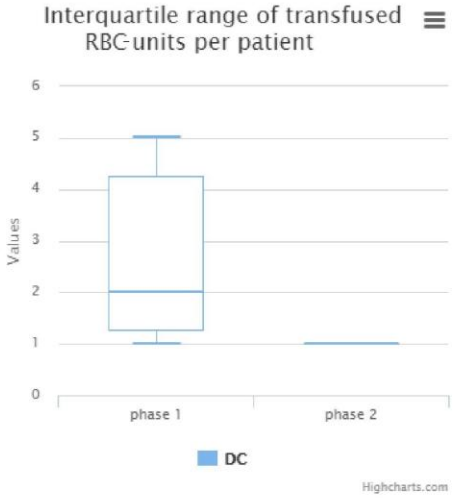
indication	ibm_date	TI	TR	mean bloodloss
CAR	phase 1	1.5	66.67	715.58
CAR	phase 2	1	100	254.03
REC	phase 1	3.5	100	819.76
REC	phase 2	0	0	172.22






Transfusion-Rate for Hb-Categories (RBC)
125
100
75
50
25
0
TR
6 to 7.6
7.6 to 12
12 to 18
DC: phase 1
DC: phase 2
Highcharts.com


Distribution of Transfused Patients on Hb-Categories (RBC)
125
100
75
50
25
0
%
6 to 7.6
7.6 to 12
12 to 18
DC: phase 1
DC: phase 2
Highcharts.com


Interquartile range of transfused RBC-units per patient
6
5
4
3
2
1
0
Values
phase 1
phase 2
DC
Highcharts.com

External Benchmarking

Demographics

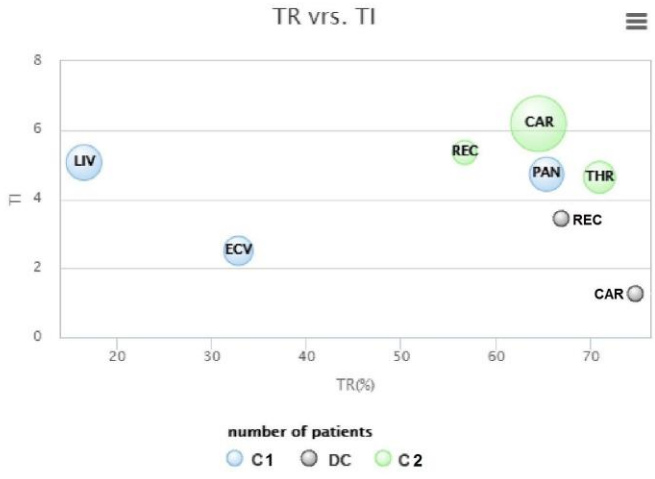
centre	indication	age	weight	size	length_of_stay	patients
C1	ECV	62.72	81.07	175.34	14.6	61
	LIV	64.43	74.77	172.81	11.15	103
	PAN	66.37	72.09	171.84	20.31	98
DC	CAR	64	77.25	170.5	7.75	4
	REC	63	75.33	178.33	10	3
C2	CAR	67.86	80.82	171.7	13.68	346
	REC	62.97	82.13	169.57	25.43	30
	THR	75.03	70.83	165.31	17.87	79

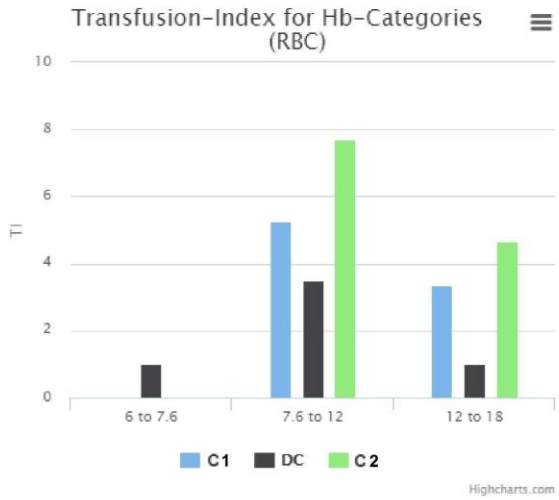
Hb-Values

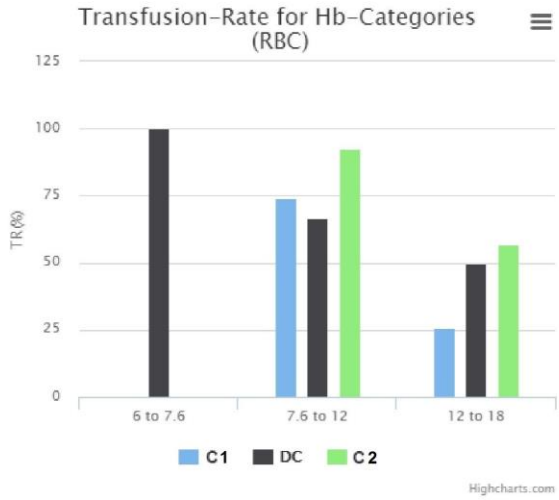
Centre	indication	hb_preop	hb_5post	Std hb_preop	Std hb_5post
C1	ECV	12.96	10.81	2	1.9
	LIV	13.42	10.85	1.73	1.85
	PAN	12.67	10.31	1.82	1.42
DC	CAR	12.97	10.53	5.02	1.71
	REC	10.1	7.5	0.46	1.06
C2	CAR	13.49	9.43	1.65	1.22
	REC	12.06	10.3	1.93	1.45
	THR	12.21	9.39	2.06	1.06

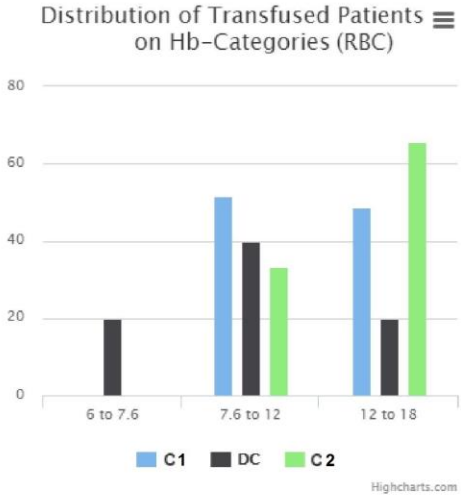
Transfusion-Parameters (RBC)

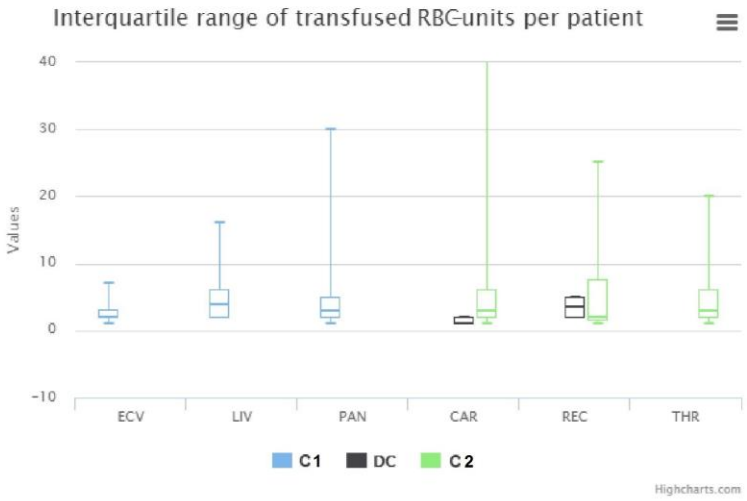
Centre	indication	TI	TR(%)	mean bloodloss
C1	ECV	2.5	32.79	386.59
	LIV	5.06	16.5	485.01
	PAN	4.72	65.31	639.38
DC	CAR	1.33	75	392.38
	REC	3.5	66.67	603.91
C2	CAR	6.18	64.45	935.95
	REC	5.35	56.67	566.08
	THR	4.63	70.89	705.86

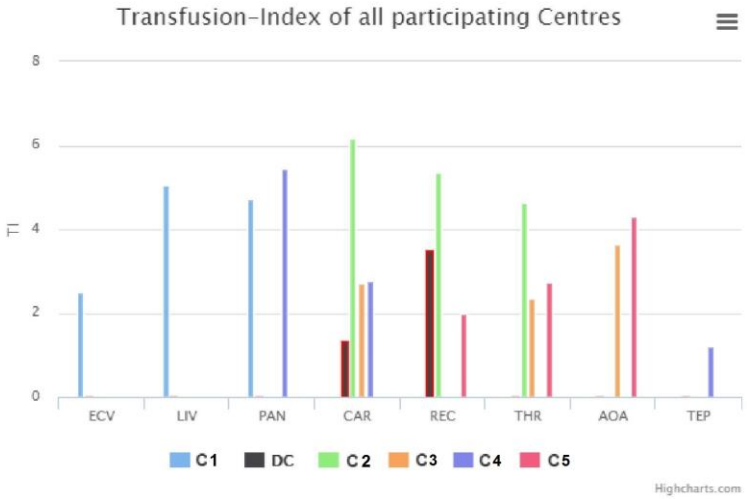


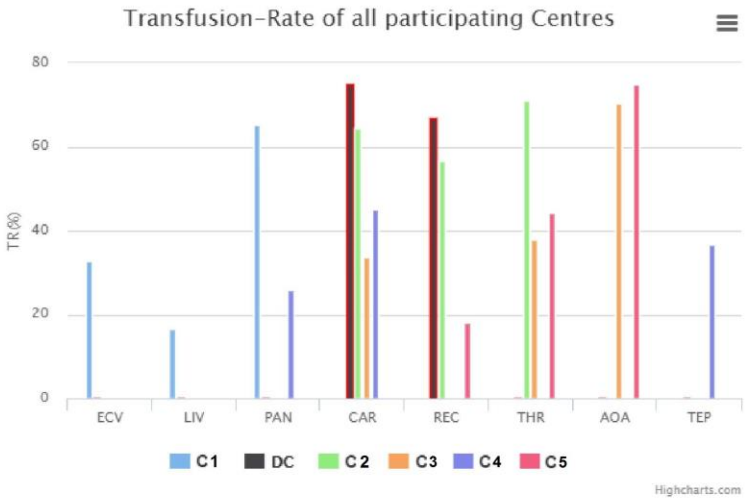

Transfusion-Index for Hb-Categories (RBC)
10
8
6
4
2
0
TI
6 to 7.6
7.6 to 12
12 to 18
C1
DC
C2
Highcharts.com


Transfusion-Rate for Hb-Categories (RBC)
125
100
75
50
25
0
TR (%)
6 to 7.6
7.6 to 12
12 to 18
C1
DC
C2
Highcharts.com


Distribution of Transfused Patients on Hb-Categories (RBC)
80
60
40
20
0
6 to 7.6
7.6 to 12
12 to 18
C1
DC
C2
Highcharts.com


Interquartile range of transfused RBC-units per patient
Values
40
30
20
10
0
-10
ECV
LIV
PAN
CAR
REC
THR
C1
DC
C2
Highcharts.com


Transfusion-Index of all participating Centres
TI
8
6
4
2
0
ECV
LIV
PAN
CAR
REC
THR
AOA
TEP
C1
DC
C2
C3
C4
C5
Highcharts.com


Transfusion-Rate of all participating Centres
TR (%)
80
60
40
20
0
ECV
LIV
PAN
CAR
REC
THR
AOA
TEP
C1
DC
C2
C3
C4
C5
Highcharts.com