TÉCNICO LISBOA

# Umedicine: A System for Clinical Practice Support and Data Analysis

**Nuno Filipe Gonçalves das Lages**

Thesis to obtain the Master of Science Degree in

## Information Systems and Computer Engineering

Supervisors: Prof. Helena Isabel de Jesus Galhardas
Prof. João Carlos Serrenho Dias Pereira
Prof. Manuel João Caneira Monteiro da Fonseca

### Examination Committee

Chairperson: Prof. Luís Manuel Antunes Veiga
Supervisor: Prof. Helena Isabel de Jesus Galhardas
Members of the Committee: Prof. Diogo Manuel Ribeiro Ferreira

**May 2017**

# Acknowledgments

First and foremost, I'd like to thank my parents, Dionil and Adelina, whose support was total, unconditional and fundamental to dive into yet another, very wished-for education venture.

I thank my supervisors, Helena, João and Manuel João, who have been most kind, available and supportive throughout the development of this work. I also thank Bernardo, Daniel and João, who collaborated with me in the development of Umedicine.

I thank my sister Ana, my aunt Lurdes and my friends Tiago, Luís, Desiree, Gabriel, Magda, Filipe and Jayant for their friendship and encouragement.

Finally, I thank Técnico Lisboa and INESC-ID for having me as a student in an environment of excellence.

# Abstract

Healthcare providers collect large amounts of clinical data, including patient personal information and medical histories. For many years, this data was recorded in paper and kept in large archives, making it difficult to search and find quickly and effectively, when required for patient care or scientific research purposes. With the spreading of the use of computers, healthcare facilities began to use information systems to collect this data. However, current software solutions collect and store unstructured data through free-form fields, making it cumbersome to retrieve and very difficult to process, thus limiting its usefulness. Hence, new software solutions are needed to collect patient information in a structured manner, enabling quick retrieval and presentation of the information, using techniques that enable quick and effective understanding by physicians, be it to inform diagnosis, treatment decisions or research.

The Umedicine information system was developed to respond to these demands, providing easy collection and always-on access to medical information, such as symptoms, examination results and prescribed treatments. However, it lacked crucial security features; in clinical information systems, it is of paramount importance to ensure that a patient's data can only be accessed by the patient themselves or by physicians. Additionally, the system did not provide means to analyze the wealth of data collected by clinical facilities, which might otherwise be used by physicians in tasks such as diagnosis, treatment decision or research.

To overcome these limitations, in this thesis we developed security and clinical decision support modules for Umedicine. The security module comprehends an authentication mechanism for four types of users: non-administrator physicians, administrator physicians, patients and clerks (administrative personnel). Each type of user has its own user interface, providing easy access to their corresponding functionalities. The clinical decision support module integrates an off-the-shelf clustering algorithm for time-series data, CCC-Biclustering, which takes advantage of the contiguity between time points to group patients efficiently. We also designed and implemented a visualization mechanism for the clustering results that shows patient data in linecharts and enables the discovery of important characteristics and statistics of each group.

We conducted performance tests for the clustering algorithm and experiments to evaluate the usability of the visualization mechanism. The performance of the clustering algorithm is satisfactory for realistic dataset sizes. The results of the visualization mechanism usability tests are promising, as the

system performed well and users were able to complete successfully all tasks, being mostly satisfied with the user interface provided. In summary, with these security and clinical decision support modules, Umedicine is an application for physicians, patients and administrative staff that keeps medical information up to date, provides easy access to a large amount of clinical data and is empowered with promising data analysis and visualization capabilities.

# Keywords

clinical information system, data analysis, clustering algorithm, data visualization

# Resumo

Os serviços de saúde recolhem grandes quantidades de dados clínicos, incluindo informação pessoal de pacientes e as suas histórias médicas. Durante muitos anos, estes dados foram registados em papel e guardados em arquivos grandes, tornando difícil procurá-los e encontrá-los com facilidade quando necessários para assistir pacientes ou para fins de investigação científica. Com a difusão do uso de computadores, os serviços de saúde começaram a usar sistemas de informação para guardar os dados. No entanto, as soluções de software actuais guardam dados não estruturados recolhidos em formulários com campos de preenchimento livre, o que torna difícil recuperá-los e muito difícil processá-los, limitando assim a sua utilidade. Portanto, são necessárias novas soluções de software para recolher informação de pacientes de forma estruturada, possibilitando a sua rápida recuperação e apresentação, utilizado técnicas que permitam uma compreensão da mesma de modo rápido e eficaz pelos médicos, seja para fins de diagnóstico, decisões sobre tratamentos ou investigação.

O sistema de informação Umedicine foi desenvolvido para responder a esta necessidade, permitindo uma recolha fácil dos dados e acesso constante a informação médica, nomeadamente sintomas, resultados de exames e tratamentos receitados. Contudo, faltavam ao Umedicine condições fundamentais de segurança; em sistemas de informação clínica, é da maior importância assegurar que dados de um paciente apenas podem ser acedidos pelo próprio paciente ou por médicos. Além disso, o sistema não tinha meios para analisar a profusão de dados recolhida pelos serviços de saúde, e que poderiam ser utilizados pelos médicos como apoio para o diagnóstico, decisão sobre tratamentos ou investigação.

Para superar estas limitações, nesta tese desenvolvemos módulos de segurança e de apoio à decisão clínica para o sistema Umedicine. O módulo de segurança contém um mecanismo de autenticação para quatro tipos de utilizador: médico não administrador, médico administrador, paciente e funcionário. Cada tipo de utilizador tem a sua própria interface, que lhe dá acesso às funcionalidades que lhe correspondem. O módulo de apoio à decisão clínica integra um algoritmo de *clustering* proposto na literatura, CCC-Biclustering, que tira partido da sequencialidade das series temporais para encontrar grupos de pacientes com eficiência. Também desenhámos e implementámos um mecanismo de visualização para os resultados do algoritmo de *clustering* que expõe os resultados em gráficos de linhas e possibilita a descoberta de características e estatísticas de cada grupo.

Realizámos testes de desempenho para o algoritmo de *clustering* e experiências para avaliar a us-

abilidade do mecanismo de visualização. O desempenho do algoritmo de *clustering* é satisfatório para conjuntos de dados de tamanho realista. Os resultados dos testes de usabilidade do mecanismo de visualização são encorajadores, uma vez que o sistema revelou um bom desempenho e os utilizadores foram capazes de completar todas as tarefas com sucesso, ficando geralmente satisfeitos com a interface. Em resumo, com os módulos de segurança e de apoio à decisão clínica, temos que o Umedicine é uma aplicação para médicos, pacientes e pessoal administrativo que mantém informação médica actualizada, dá acesso a uma grande quantidade de dados clínicos e tem capacidades prometedoras de análise e visualização de dados.

# Palavras Chave

Sistema de informação médico, análise de dados, algoritmo de *clustering*, visualização de dados

# Contents


**1 Introduction** **1**

1.1 Umedicine: A System for Clinical Practice Support . . . 3

1.2 Objectives . . . 4

1.3 Contributions . . . 4

1.4 Document Outline . . . 5

**2 Related Work** **7**

2.1 Online Analytical Processing in Clinical Information Systems . . . 9

2.1.1 Data Warehousing . . . 9

2.1.2 The Intermount Healthcare Enterprise Data Warehouse . . . 10

2.1.3 A Data Warehouse for Clinical Research on Obesity . . . 12

2.1.4 Conclusion . . . 12

2.2 Data Analysis for Clinical Decision Support . . . 13

2.2.1 Logistic Regression . . . 15

2.2.2 Bayesian Networks . . . 16

2.2.3 Artificial Neural Networks . . . 18

2.2.4 Production Rules . . . 19

2.2.5 Conclusion . . . 20

2.3 Information Visualization for Clinical Diagnosis . . . 21

2.3.1 Visualizing Personal Histories . . . 22

2.3.2 Visualizing Patient Population Data . . . 23

2.3.3 Conclusion . . . 25

**3 The Umedicine Application** **27**

3.1 Requirements . . . 29

3.2 Architecture . . . 30

3.2.1 Front end . . . 31

3.2.2 Back-end Server . . . 34

3.2.3 Relational Database . . . 36

3.3 Implementation . . . 38
3.4 Conclusion . . . 39

**4 Umedicine: Security Module** **41**
4.1 User Types . . . 45
4.2 Authentication . . . 47
4.3 Database Schema . . . 49
4.4 User Interface . . . 50
4.5 Implementation Details . . . 51
4.6 Conclusion . . . 53

**5 Umedicine: Clinical Decision Support Module** **55**
5.1 A Clustering Approach for Clinical Time Series . . . 57
5.1.1 CCC-Biclustering . . . 58
5.1.2 Implementation Details . . . 60
5.2 Visualization of Clinical Time-Series Data Clusters . . . 61
5.2.1 Implementation . . . 63
5.3 Discussion . . . 65

**6 Validation** **67**
6.1 Clustering and Visualization Performance Evaluation . . . 69
6.1.1 Experimental Setup . . . 69
6.1.2 Results and Discussion . . . 70
6.2 Usability Tests . . . 71
6.2.1 Experimental Setup . . . 72
6.2.2 Results and Discussion . . . 72

**7 Conclusion and Future Work** **77**
7.1 Summary . . . 79
7.2 Limitations and Future Work . . . 80

**A Usability Test For the Biclustering Visualization Mechanism** **89**

# List of Figures


2.1 Architecture of the data warehouse at Intermount Healthcare. Figure from Evans *et al.* [10]. EDW is an acronym for Enterprise Data Warehouse. . . . . 11
2.2 Star schema designed by Narra *et al.* [28]. . . . . 13
2.3 An example of a ROC curve with the true positive and false negative rates represented as percentages. Figure from www.medcalc.org/manual/roc-curves.php. . . . . 15
2.4 An example of a Bayesian network, from Binder *et al.* [3]. (a) is the corresponding Directed Acyclic Graph (DAG) and (b) is the conditional probability table for the attribute Emphysema, which depends on the attributes Smoker and CoalMiner. . . . . 16
2.5 HMM . . . . . 17
2.6 ANN . . . . . 18
2.7 The LifeLines environment [31]. Image from Rind *et al.* [35]. . . . . 21
2.8 The WBIVS environment [30], which shows aligned line and matrix plots to follow the evolution of a lung transplant patient. Parts of the visualization are magnified on the right for a better view of the details. Image from Rind *et al.* [35]. . . . . 23
2.9 The LifeLines2 environment [42,43] for visualization of multiple records containing temporal categorical data. (a) shows a stack of records, (b) shows the histogram for a selected event within a chosen time window, (c) contains the controls for different operations and (d) shows controls for event distribution, grouping and other options. Image from Rind *et al.* [35]. . . . . 24
2.10 Gravi++ [17]. The squared icons represent attributes and the circular icons represent patients. Three patients are represented at several time points during an animation. Circles around the attribute icons represent attribute values for the patients. Users can choose patients and attributes on the left and select options on the right. Image from Rind *et al.* [35]. 25

3.1 Architecture of the Umedicine system. . . . . 31
3.2 Physician homepage. . . . . 31
3.3 Patient search page available to the physician. . . . . 32

3.4 Umedicine's patient search results page. . . . . 32
3.5 A patient's information page as viewed by a doctor using Umedicine. . . . . 32
3.6 View of a patient's treatment history in Umedicine. . . . . 33
3.7 A patient's information page. . . . . 34
3.8 Umedicine's server component architecture. . . . . 35
3.9 Parameter Volume of type FLOAT and unit milliliter (mL) . . . . . 38

4.1 Architecture of the Umedicine system presented in this chapter. The modules in blue are new. The *Decision Support* and *Decision Support Pages* modules are described in Chapter 5. . . . . 44
4.2 Form to create a physician account.If the checkbox is checked, the physician will be an administrator. The form to create a clerk account is the same except for the checkbox, which is absent. . . . . 46
4.3 Form where clerk users create patient accounts. For patients a password of no less than 8 characters must be provided when the account is created. . . . . 46
4.4 After creating a patient account, clerks are asked if they want to introduce information about the patient. . . . . 47
4.5 Form where a clerk can add limited personal information about a just-added patient user. 47
4.6 Umedicine's new login page, which has a button that navigates to a form where users can request a new password (Figure 4.7). . . . . 48
4.7 Page where users with an email address can request that a new password is sent to them. 48
4.8 Umedicine's password change form, which includes a checkbox that displays the password as text on the screen when checked. . . . . 49
4.9 Homepage of administrator physicians. The page has buttons that lead to pages where physicians and clerks can be added to the system. The homepage of non-administrator physicians doesn't show these 2 buttons (the middle line). . . . . 51
4.10 Patient information form with user-guidance techniques. The inserted date ('Data') does not respect the required format, the telephone number ('Telefone') does not meet the minimum length and the user has navigated away to the 'profession' field ('Profissão'). . . 52

5.1 (Left) Example of a discretized matrix, with maximal CCC-Biclusters with at least two rows (B1 to B4) identified by circles. The strings N, UDU, U, and UN correspond to the patterns of the maximal CCC-Biclusters B1, B2, B3 and B4, respectively. The dollar sign is just a string termination marker. (Right) Suffix tree for the symbolic matrix shown on the left. For clarity, the figure does not show the leaves that represent string terminators that are direct children of the root. Figure from [24]. . . . . 60

5.2 Visualization of time series clustering in Umedicine applied to the IPSS, a diagnosis parameter for BPH. . . . . 61
5.3 Example of information displayed over a plot (from figure 5.2) representing a cluster of patients with similar IPSS evolution. . . . . 63
5.4 Example of information displayed when a user clicks the button 'Ver mais' (meaning 'get more information') on plot such as the one in Figure 5.3. . . . . 64

6.1 Performance measurements for the Contiguous Column Coherent Biclustering (CCC-Biclustering) algorithm and the cluster visualization mechanism. Left panel: results for different numbers of patients for a fixed number of 5 time points per patient. Right panel: results for different numbers of time points per patient, for a fixed number of 1000 patients. . . . . 70

# List of Tables

2.1 Data Warehousing Approaches for Clinical Decision Support . . . . . 13
2.2 Data Mining Approaches for Clinical Decision Support . . . . . 20
2.3 InfoViz Systems for Clinical Decision Support . . . . . 26

6.1 Usability test results . . . . . 73
6.2 Usability test results: patient improvement according to the users . . . . . 74

# Acronyms

**ANN** Artificial Neural Network

**Ajax** Asynchronous JavaScript and XML

**ANNs** Artificial Neural Networks

**API** Application Programming Interface

**BPH** Benign Prostatic Hyperplasia

**CCC-Biclustering** Contiguous Column Coherent Biclustering

**CCC-Biclusters** Contiguous Column Coeherent Biclusters

**CDS** Clinical Decision Support

**CSV** comma-separated values

**DAG** Directed Acyclic Graph

**DAO** Data Access Object

**DAOs** Data Access Objects

**DBMS** Database Management System

**DOM** Document Object Model

**EHRs** Electronic Health Records

**EM** Expectation Maximization

**ETL** Extraction, Transformation and Loading

**FPR** False Positive Rate

**GMM** Gaussian Mixture Distribution Model

| | |
|---|---|
| **HL-GOF** | Hosmer-Lemeshow Goodness-of-Fit |
| **HMM** | Hidden Markov Model |
| **HTML** | HyperText Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **IH** | Intermount Healthcare |
| **InfoViz** | Information Visualization |
| **IoC** | Inversion of Control |
| **IPSS** | International Prostate Symptom Score |
| **JDBC** | Java Database Connectivity |
| **JSP** | JavaServer Pages |
| **ML** | Machine Learning |
| **MVC** | Model-View-Controler |
| **OLAP** | Online Analytical Processing |
| **OLTP** | Online Transaction Processing |
| **ROC** | Receiver Operating Characteristic |
| **SL4J** | Simple Logging Facade for Java |
| **SQL** | Structured Query Language |
| **SVG** | Scalable Vector Graphics |
| **TLS** | Transport Layer Security |
| **TPR** | True Positive Rate |
| **URL** | Uniform Resource Locator |
| **VTE** | Venous Thromboembolism |
| **WBIVS** | Web-based Interactive Visualization System |

# 1

# Introduction

## Contents


1.1 Umedicine: A System for Clinical Practice Support . . . . . 3
1.2 Objectives . . . . . 4
1.3 Contributions . . . . . 4
1.4 Document Outline . . . . . 5

For health care providers, recording every patient's clinical information comprehensively is of paramount importance. However, for many years, this information has been recorded in paper, and kept in large dedicated archives, making it difficult to retrieve relevant past clinical information quickly and effectively, when required for patient care or for scientific research purposes.

With the widespread use of computers, new software tools were developed for clinical staff to record information about patients. This easy access to clinical data promises a significant impact in clinical practice. In particular, continuous patient monitoring can be ubiquitous, enabling fast response by clinical staff and quick situation assessment by physicians; clinical research can benefit from much larger, easier-to-access data sources, that will accelerate the discovery of new medical knowledge; and health care managers will be able to make more informed decisions regarding institutional policies.

However, existing medical information systems are often too complex to use, and/or only provide free-form fields for collecting patient's medical information, which therefore is not stored as structured data. For this reason, many patient's records are still in paper and, in the cases where the information is in the system, it is not possible to gather statistical information about a population of patients, thus limiting the usefulness of data. As a consequence, physicians often give up searching for relevant information to support medical decisions because the task is too time-consuming [7, 35]. Hence, there is still a need for tools to enable integration and analysis of clinical data in an effective manner [35, 37] and to make it useful in everyday practice. Moreover, physicians should have means — Clinical Decision Support (CDS) systems — to explore existing patient data quickly and thus support their diagnosis, treatment decisions and research.

## 1.1 Umedicine: A System for Clinical Practice Support

Umedicine is a medical information system designed and developed at INESC-ID[1], under the supervision of an Urology practitioner. It is a web-based application for physicians, patients and administrative staff with an appealing and easy-to-use graphical interface, that aims to address the current limitations of clinical information systems.

Umedicine is designed to meet the requirements of physicians. By using it, physicians and patients keep medical information (e.g., symptoms, clinical examination results, and prescribed treatments) up to date on a platform that provides easy, quick and always-on access to a large amount of clinical data. The medical information is kept persistently in an electronic and structured format, thus enabling the visualization of medical data per patient.

However, the Umedicine system, prior to the work presented in this thesis, still lacked support for use by hospital administrative staff and didn't meet the security standards required for a medical information

[1] Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento em Lisboa, http://www.inesc-id.pt/

system. Additionally, the fact that Umedicine stores structured medical data offers an unexplored potential: structured data facilitates the application of data mining techniques to extract interesting knowledge from the collected medical data. In this work, we address these limitations, we redesign Umedicine's user types and interface, implement security requirements and explore Umedicine's structured data potential to build a CDS module for Umedicine with data analysis capabilities.

## 1.2 Objectives

Considering the original state of the Umedicine application and the presented literature survey, the objectives of the present work are twofold.

The first objective is to complement Umedicine with security and usability features that are necessary for production use in clinical facilities, but were missed in the current version of the system. These features include: supporting different user types (administrator and non-administrator physician, clerk and patient), providing each with an appropriate user interface and ensuring that each has access to, and only to, the respective functionalities.

The second objective is to implement a CDS module for the application. Specifically, provide data analysis techniques to find and explore the characteristics of groups of patients with similar medical histories, namely what treatments were successful or unsuccessful.

## 1.3 Contributions

In the context of this thesis, we solved several shortcomings of the system and integrated a data analysis module comprising a clustering algorithm for patient time-series data and a visualization mechanism to explore the results of this clustering algorithm. First, we supplied the application with a previously-absent set of features required for its use in production, namely appropriate user types, authentication, management of user sessions and password generation. Then, we developed a data analysis module to equip Umedicine with CDS capabilities, which is one of the differentiating features of the system. Specifically, our main contributions are:

1. Defined and implemented appropriate user types in the system and interfaces for each user type
2. Designed and implemented various security features that prevent breaches of data confidentiality
3. Developed a CDS module to enable analysis and exploration of medical time-series data:
   (a) Added a clustering algorithm to the application, especially suited for time-series data, applied to patient data.

(b) Implemented a visualization mechanism to explore the results of the clustering algorithm.

(c) Conducted performance experiments and usability tests to evaluate the clustering algorithm and the visualization mechanism.

We performed evaluations of the clustering algorithm's performance and of the visualization mechanism's usability. For the former, we conducted experiments where we measured time of execution for various algorithm input sizes. For the latter, we asked users to perform several task scenarios, collected usability measures and asked them to fill in a satisfaction questionnaire to qualitatively assess the proposed visualization mechanism. The evaluation results are promising, as the system performed well for realistic input sizes and users were able to complete successfully all tasks, and were mostly satisfied with the interface provided.

The work presented in this thesis is the basis for an research paper submitted to DMAH 2017 (The Third International Workshop on Data Management and Analytics for Medicine and Healthcare), to be held in Munich, Germany, on September 1, 2017, in conjunction with the VLDB conference (43$^{rd}$ International Conference on Very Large Data Bases).

## 1.4 Document Outline

This thesis is organized as follows.

In Chapter 2 we describe existing research work in three fields of CDS: databases for clinical data analysis, clinical data mining methods and clinical information visualization systems.

In Chapter 3, we describe the requirements set for the Umedicine system, elaborated in collaboration with a physician practitioner and its original architecture and functionalities, as well as the details of its implementation.

In Chapter 4, we propose a new version of the Umedicine system, supporting multiple user types, enhanced security, with an upgraded database schema and a user interface adapted to the new user types.

In Chapter 5, we present the CDS module integrated in the application. The data mining capabilities comprise a clustering algorithm to group medical time-series data and a visualization mechanism that enables the exploration of the results of the clustering approach. We also provide a description of the technologies used and details about the implementation of the CDS module, and a discussion that wraps the chapter.

In Chapter 6, we evaluate the data analysis approach proposed in Chapter 5. Specifically, we describe experiments to measure the performance of both the clustering algorithm and the visualization mechanism available to users to explore the results, and the results of usability tests for the visualization mechanism.

Finally, Chapter 7 provides a conclusion for the thesis, with a summary of the developed work and an analysis of its limitations and possible future lines of work.

# 2

# Related Work

## Contents


2.1 Online Analytical Processing in Clinical Information Systems . . . . . 9

2.2 Data Analysis for Clinical Decision Support . . . . . 13

2.3 Information Visualization for Clinical Diagnosis . . . . . 21

In this chapter, we review relevant published work in CDS. The text is organized in three sections corresponding to research areas that are provide the basis for solutions in CDS information systems: online analytical processing/data warehousing (Section 2.1), data mining methods (Section 2.2) and information visualization (Section 2.3). In each section, we review the fundamentals of the field and contributions from the research literature.

## 2.1 Online Analytical Processing in Clinical Information Systems

The ultimate purpose of a CDS system is to provide practitioners useful information to aid them in making decisions that result in the best outcome for a patient. The development of suitable strategies for handling data in the context of the broader field of decision support has been driven by the *business intelligence* needs of organizations, where technologies exist to transform raw data into meaningful information that managers can use for decision making [40]. This transformation process prepares the data appropriately for analysis, possibly for further processing by data mining algorithms. It involves data extraction (possibly from different sources), transformation, integration and cleaning. Data processed this way is usually stored in a data warehouse, a data repository designed to facilitate access to data through complex queries [40]. Typically, data warehouses enable interactive querying and automatic aggregation of data [40]. In this chapter we review data warehousing concepts (Section 2.1.1) and reference data warehousing approaches for CDS (Sections 2.1.2 and 2.1.3).

### 2.1.1 Data Warehousing

Data warehouses are built with basis on the Online Analytical Processing (OLAP) paradigm. OLAP is designed for heavy analytical query loads rather than heavy transaction load and concurrency, for which relational databases, with basis on the Online Transaction Processing (OLTP) paradigm, are more suitable. Analytical queries often involve reading many records and aggregation operations, requiring different indexing and denormalization strategies from those used in OLTP systems [38].

OLAP systems are based on a multidimensional data model, where the data structure is represented as a hypercube. Hypercube edges represent dimensions corresponding to the various perspectives that data is to be analyzed from. Hypercube cells, also named *facts*, contain the *measures* to be analyzed. Under each dimension, there may exist different levels of granularity named *dimension levels* [38].

Data warehouses are usually designed to store consolidated data obtained from various source databases. It is often the case, however, that a department of the organization only needs a subset of this data for its regular activities. A data warehouse with a subset of the information of an organization that is designed to serve the needs of a specific department or business area is known as a *data mart*. For example, a sales department may have its own data mart with sales data; a human resources

department may have its own data mart with employee data and so on [38]. Hence, an organization may be served by a collection of data marts. The organization's data warehousing system may be organized following a bottom up approach, if the data marts are developed somewhat independently and integrated later, or a top-down approach, if a common data warehouse is developed first and serves as basis for the data marts. In the latter case, the data marts may be mere logical views of a single warehouse [38].

The collection of data from different sources is a key aspect of the development of a data warehouse. The process of taking data from data sources, transforming it to fit the warehouse model and loading it into the warehouse is called Extraction, Transformation and Loading (ETL). This is a complex and costly process that requires careful design [39].

The data processed in the ETL process is stored in the warehouse according to a predefined schema that is also designed with analytical queries in mind. These schemas may be of two kinds: *star schemas* or *snow flake schemas*. Both are composed of relational tables, albeit organized in different ways. In star schemas there is a unique table for each hypercube dimension, which may imply denormalization. Snowflake schemas use only normalized tables [40]. In both cases, a single special table exists, the *fact table*, which contains foreign keys of *dimension tables* and measures. A *measure* is a specific attribute of interest or the result of applying an aggregation function. Fact tables are usually normalized and their primary key is the concatenation of the foreign keys. The primary key determines all the measures as there are no functional dependencies between the foreign keys [41]. Variations of these two schema structures exist as well. A *starflake schema* is a hybrid solution where some dimensions are normalized and other are not. A *constellation schema* allows for multiple fact tables sharing dimension tables, which may be normalized or not [41].

Having given a overview of fundamentals in data warehousing, we are now ready to review relevant applications of these concepts in medical information systems.

### 2.1.2 The Intermount Healthcare Enterprise Data Warehouse

Intermount Healthcare (IH)[1] is an organization in the United States under which 22 hospitals and over 179 clinics operate. Evans and colleagues described the enterprise data warehouse in production at IH and two application case studies [10]. IH's databases accounted for 80 billion rows of data in 9000 tables at the time of writing of their paper. The data warehouse supports not only patient medical data but also all kinds of data from the organization's business. The warehouse is updated every night with patient data, administrative data and data from relevant US national pharmaceutical and genetic databases. It serves all kinds of users, from database administrators to clinicians and executives, through reports and dashboards. The system's architecture is illustrated in Figure 2.1. The system is composed of several data marts designed to meet a variety of clinical decision support activities. The warehouse also includes

[1] https://intermountainhealthcare.org/

three additional repositories: a Master Reference Data repository, which stores keys that identify different actors such as patient and hospital IDs, a metadata repository, for easy access to relevant metadata, and a repository designed specifically for security and auditing purposes. The authors do not give details on the inner architecture of these solutions. The technology used to build both the relational databases and the warehouse is from Oracle, with many custom Java applications to extract information from the latter.

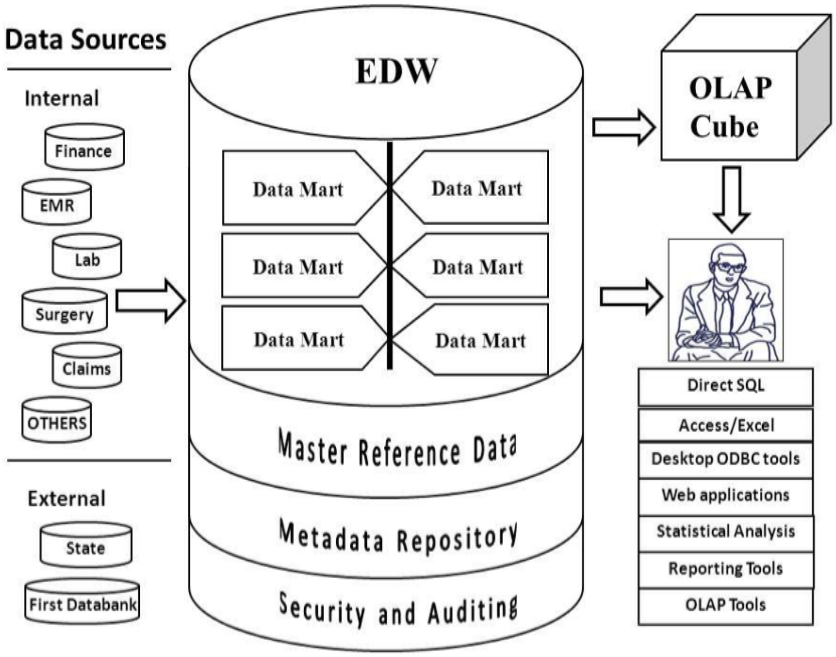



**Figure 2.1:** Architecture of the data warehouse at Intermount Healthcare. Figure from Evans *et al.* [10]. EDW is an acronym for Enterprise Data Warehouse.

Evans and colleagues illustrate the capabilities of the system with two use cases.

In the first use case, a data mart was developed to store data from patients infected with multi-drug resistant microorganisms. Upon admission of a patient with a previous infection by certain microorganisms, the admission information is collected into the data warehouse. When this happens, the system sends a phone text message to infection preventionists with relevant information to help them take immediate measures to protect signaled patients, other patients and hospital staff. The authors do not provide an assessment of the effectiveness of the application in preventing the spread of infections, they only note that infection preventionists were frequently not aware of the presence of infected patients prior to deployment of the application.

In the second use case, a data mart was developed to collect the information of risk for Venous Thromboembolism (VTE). Every night, database tables would be scanned for specific data (surgery duration, body mass index, age, bed rest, among others) from which a risk score would be computed. Patients at high risk for VTE would be signaled. Signaling could happen at admission or after some

time, depending on patient evolution. The identification of high-risk patients was evaluated to have a sensitivity of 98%. An extra functionality was added to alert the pharmacist and check if these patients' medications satisfied the appropriate dosages.

### 2.1.3 A Data Warehouse for Clinical Research on Obesity

Narra *et al.* proposed a data warehouse model to study various health issues and applied it to an obesity research case study based on a dataset from the Australian Bureau of statistics [28].

The original data used in the study was organized in spreadsheets and presented people's weight with respect to subcategories of nutrition, physical activity, gender and age. The choice of a data warehousing architecture was conditioned by the entire study being focused on a single factor – the weight of subjects. Narra *et al.* designed a star schema, shown in Figure 2.2, with a fact table containing measurable attributes and attributes that link to multiple dimension tables. The dimensions tables contain data pertaining to different factors with suspected relation to obesity (physical exercise, kind of milk consumed, geographical location, fruit intake and others, as shown in Figure 2.2).

The authors chose Microsoft SQL Server to implement the hypercube corresponding to the star schema and extract, transform and load the data from the spreadsheets into the warehouse, and generate reports. The system also generated reports including bar charts, line charts and tables for data visualization. The authors show the usefulness of the approach by identifying correlations between different variables and obesity. For example, they observed a strong correlation between obesity and physical exercise but not between obesity and the type of milk consumed.

### 2.1.4 Conclusion

In this section we introduced the fundamentals of data warehousing and reviewed two examples of data warehouses used in CDS systems. These two examples are summarized in Table 2.1.

In the process of integrating data mining methods with a medical information system, it is important that the input data for the algorithms is organized in a way that enables efficient querying. Data warehouses are designed with this goal in mind and we described relevant examples from the literature. From Narra *et al.*'s work, we learned how a relatively simple data warehouse can be an effective tool for research on specific clinical subjects. From the IH experience we learned how a system that supports a variety of patient care tasks can scale up to accommodate a growing amount of data, by using several data marts and a few complimentary repositories.

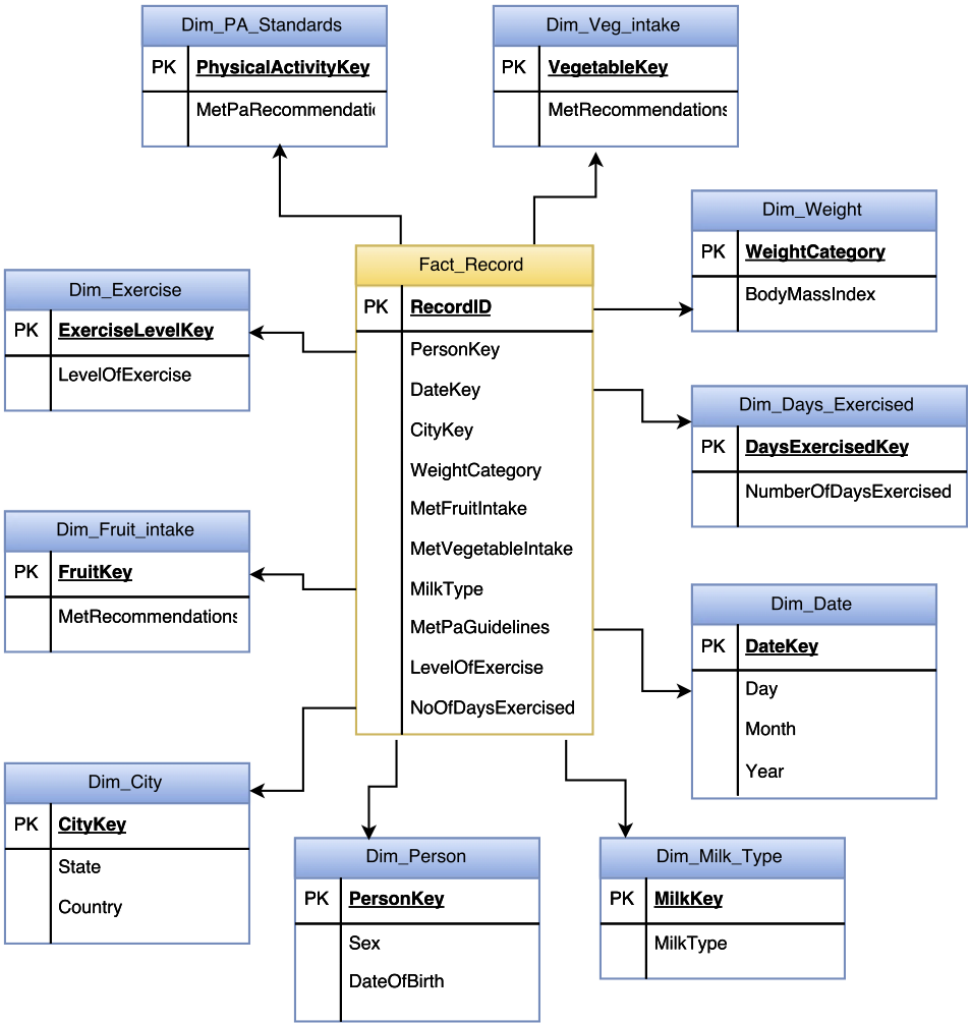



**Figure 2.2:** Star schema designed by Narra *et al.* [28].

**Table 2.1:** Data Warehousing Approaches for Clinical Decision Support

| | Intermount Healthcare | Narra *et al.* |
|---|---|---|
| Technology | Oracle | Microsoft SQL Server |
| Architecture | Several data marts; master reference data, metadata and security/auditing repositories | Single repository with star schema |
| Main purpose | Support all aspects of a large healthcare organization business; support patient care and decision making by the organization's staff | Research on a specific health issue, oriented toward statistical analysis, visualization and report generation |

## 2.2 Data Analysis for Clinical Decision Support

To make the best use of the wealth of data from patient data for diagnosis, physicians should have quick means to explore the information therein for the benefit of patients. In the context of diagno-

sis, it is useful to be able to relate the data from patients to identifiable patterns in Electronic Health Records (EHRs). In order to identify patterns in large amounts of data or accurately compare a patient's symptoms to a specific group of patients or pattern in the EHRs, *Data Mining* methods are useful. According to Witten *et al.*, 'data mining is the extraction of implicit, previously unknown, and potentially useful information from data' [45], using statistical and Machine Learning (ML) methods. According to Matheny and Ohno-Machado [25], 'data mining techniques are pattern recognition techniques intended to find correlations and relationships in the plethora of data.' This section provides an overview of the use of statistical and ML methods in the context of CDS.

The data mining method of choice for a specific task depends on the nature and goal of the task at hand. The task may be mostly exploratory and aiming at finding patterns without guidance based on previous knowledge. Methods falling under this category do not attempt to match an object to a predetermined group of labeled examples, but rather try to find groups without basis on a previous classification. These are said to be *unsupervised learning* methods and clustering algorithms such as K-means are well-known examples [46]. On the other hand, *supervised learning* methods produce models with basis on a set of labeled data examples – the training set. The produced models may then be used to label new objects. In other words, supervised learning methods are used to solve classification problems. Supervised learning methods have been more extensively used for CDS than unsupervised methods, notably Bayes networks, decision trees and Artificial Neural Networks (ANNs). In general, there has been a clear preference for the simpler, more intuitive data mining methods than for the most sophisticated ones, such as Support Vector Machines.

Once an ML algorithm is chosen, appropriate criteria must be used to assess its performance. In the case of classification problems, we are interested in measuring how accurately the algorithm classifies objects. In other words, how often it classifies an object correctly. Several approaches can be used for this purpose, including the holdout method, cross-validation, the bootstrap method, statistical significance tests and ROC curves [12]. Receiver Operating Characteristic (ROC) curves are an intuitive visual means to determine the quality of a classifier models [14]. In a ROC curve plot, the $y$ axis represents the True Positive Rate (TPR) or sensitivity. The TPR is the ratio of true positives (objects correctly labeled as belonging to a class) over all positives (all objects classified as belonging to a class, correctly or not). The $x$ axis of a ROC curve plot represents the False Positive Rate (FPR), the ratio of false positives (objects incorrectly labeled as belonging to a class) over all positives. The higher the area under a ROC curve, the higher is the number of correctly classified objects [12]. Figure 2.3 shows an example of a ROC curve.

The rest of this section reviews recent applications of data mining methods to CDS systems.

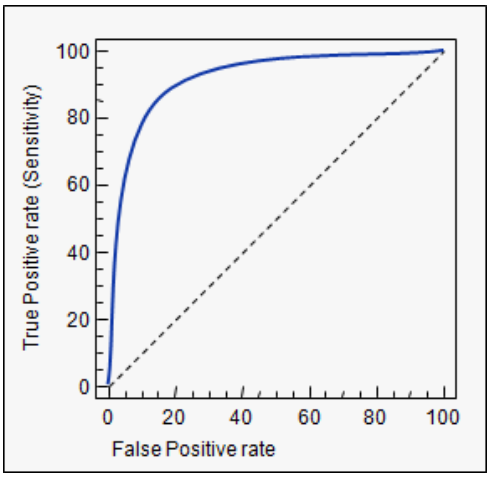



**Figure 2.3:** An example of a ROC curve with the true positive and false negative rates represented as percentages. Figure from www.medcalc.org/manual/roc-curves.php.

### 2.2.1 Logistic Regression

The logistic regression has been the most widely used method for predictive modeling in medicine [22, 25]. It has been valuable both on its own and in conjunction with more sophisticated methods such as ANNs. This method is based on the logistic function given by Equation 2.1.

$$Y = \frac{1}{1 + e^{-(\beta \mathbf{x} + c)}} \tag{2.1}$$

In equation 2.1, $Y$ represents the predicted outcome and **x** represents a vector of variables that are attributes in the dataset. $c$ is a constant that calibrates the baseline rate of the outcome of interest [25]. This equation corresponds to a sigmoid curve that describes two intervals with approximately constant values and a sharp transition between them. This behavior makes the continuous logistic function suitable to approximate binary variables, which can translate to a decision such as the patient having ($Y = 1$) or not having ($Y = 0$) a certain condition [25].

The performance of a logistic regression is usually assessed with basis on two parameters, discrimination and calibration. Discrimination refers to the capability of the model to discriminate predictions with the outcome from predictions without the outcome. The area under the ROC curve is a measure of discrimination [2, 25]. Calibration refers to how close the estimated probability of a prediction is from the probability of the actual outcome and it can be determined with a Hosmer-Lemeshow Goodness-of-Fit (HL-GOF) test [18, 25].

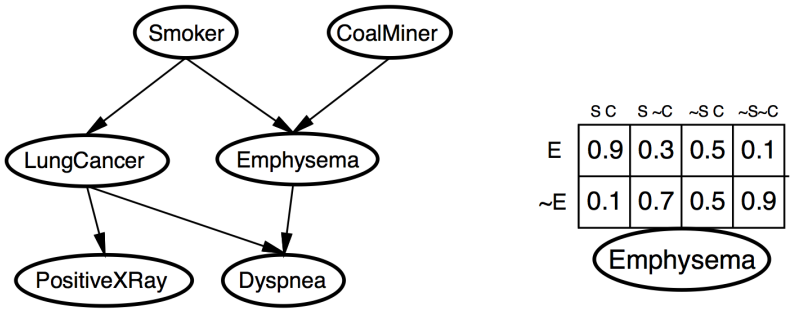
An interesting example of the use of logistic regression in the health care field comes from an attempt to predict whether a patient will or will not be readmitted in a hospital up to 30 days after discharge [9]. For this purpose, the HOSPITAL score was proposed, based on a logistic regression model [9]. The name of the score is derived from the variables used to compute the prediction: Hemoglobin level, discharge from

Oncology service, Sodium level, number of Procedures during admission, Type of admission, number of Admissions in the last year and Length of stay. The authors applied their method to a real dataset from a hospital in Boston, Massachusetts, and claimed to have predicted a readmission risk of 18% against an observed readmission risk of 18.2%, with good discriminatory power (C statistic, 0.71)[2] and calibration (non-significant p-value in the HL-GOF test).

### 2.2.2 Bayesian Networks

Bayesian Networks were among the first data mining methods employed in CDS systems, in the 1960s. Probabilities were estimated from patient data with the help of experts [21, 25]. However, these methods did not become widely adopted beyond the realm of research [25].

A Bayesian network is defined by a DAG and a set of conditional probability tables, one table for each attribute. For an example, see Figure 2.4. In the DAG each node is an attribute and edges define dependency relationships between attributes, with the child node attributes depending (only) on the parent node attributes. For example, in Figure 2.4, the probability of a patient having dyspnea depends on them having lung cancer and emphysema but it does not depend on them being a smoker, being a coal miner or a having a positive X-ray [13, 46].




**(a)** **(b)**

**Figure 2.4:** An example of a Bayesian network, from Binder *et al.* [3]. (a) is the corresponding DAG and (b) is the conditional probability table for the attribute Emphysema, which depends on the attributes Smoker and CoalMiner.

Given the class label of a record, the values of the attributes are assumed to be conditionally independent of one another. Under these conditions, let $X = (x_1, ..., x_n)$ be a data tuple (a record) with $n$ attributes, we can write equation 2.2 [13, 46]:

$$P(x_1, ..., x_n) = \prod_{1}^{n} P(x_i | x_i\text{'s parents}) \tag{2.2}$$

[2]The C statistic is equivalent to the area under the ROC curve [15]

Various algorithms can be used to infer Bayes networks, which can be used to return a class in a classification problem or a probability distribution that gives the probability of each class [13].

Recently, a special case of Bayesian network, a Hidden Markov Model (HMM) [27], was the basis for a new unsupervised learning approach with clinical application. Najjar and colleagues proposed a method based on Gaussian Mixture Distribution Model (GMM) clustering and HMM to mine EHRs and discover groups of similar patients in an unsupervised fashion [27]. Najjar's method is able to handle numerical and categorical attributes present in the records. It works in two steps. The first step handles numeric and categorical attributes with finite, unordered domain. The second step handles multivalued categorical attributes, whose domain comprises sequences of categorical values instead of a single value. The results of the two steps are combined to yield groups that account for all three kinds of attributes.

In the first step, records are clustered into K groups, considering numerical and univalued categorical attributes. Each group is characterized by a probability distribution function with a distinct parametrization. Numerical attributes are modeled as Gaussian distributions and categorical attributes are modeled with a multinomial density function. The parameters of the mixture model are determined with the Expectation Maximization (EM) algorithm. At the end of this first step, each object is a member of one of the K groups.

The second step handles multivalued categorical attributes. This step trains an HMM for each multivalued attributes in each cluster obtained in the first step. Briefly, the HMM used by Najjar and colleagues is a Bayes network where the elements of the sequence (value) of a categorical attribute correspond to the nodes of the network. The actual values in the sequence are unknown, hence the name *hidden* Markov model [34]. Each node is, however, connected to yet another node that is *observable* – a value in a sequence from a record. The probability that a value is observed given the state of the corresponding hidden node is called the emission probability. A simple illustration is given in Figure 2.5.

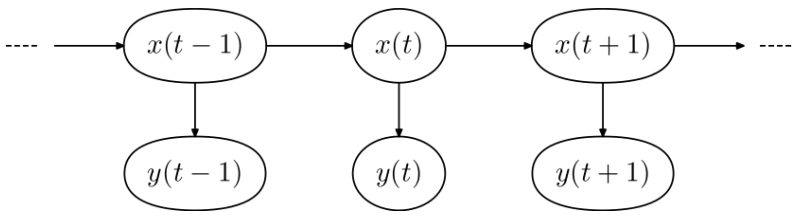



**Figure 2.5:** A Hidden Markov Model for a multivalued categorical attribute. $t$ is the index of the value; $x$ represents the values in the hidden states; $y$ represents the observed value; the transition between each $x(i)$ and $y(i)$ represents emission. Figure from Wikimedia Commons (commons.wikimedia.org/w/index.php?curid=4352728).

Najjar *et al.* trained an HMM by finding the maximum log-likelihood for each multivalued attributes in each cluster from the first step. For training, they used the Baum-Welch algorithm [34]. Then they computed the emission probabilities using the forward algorithm [34]. The lower the emission probabilities,

the lowest the quality of the clusters found in the first step for the multivalued attributes.

The first and second steps of the algorithm were then combined so that the probability of a record belonging to the corresponding cluster reflected the three kinds of attributes. This was done by multiplying the probability obtained for each record in the first step by the corresponding probability computed in the second step. This product is then used as input to reestimate the GMM in the first step. The algorithm iterates a number of times over these two steps until satisfactory convergence is reached. In a sentence, this method tries to balance the contributions of numerical, categorical and multivalued categorical attributes when grouping the individuals, by combining the outputs of appropriate models for each kind of attribute.

The Najjar method was applied to a dataset from the healthcare system of Québec, Canada, comprising records from people over 65 years old with heart diseases. They found groups of hospital stays corresponding to pathologies such as cataracts, prostate disease and mental disorders, among others. Najjar *et al.* compared their 2-step method to the EM algorithm alone and found that their algorithm achieved better results.

### 2.2.3 Artificial Neural Networks

ANNs are flexible models inspired by the structure of biological nervous systems. Such models are composed of several processing units (neurons), usually organized in more than one connected layer. Figure 2.6 shows an example of a simple Artificial Neural Network (ANN).

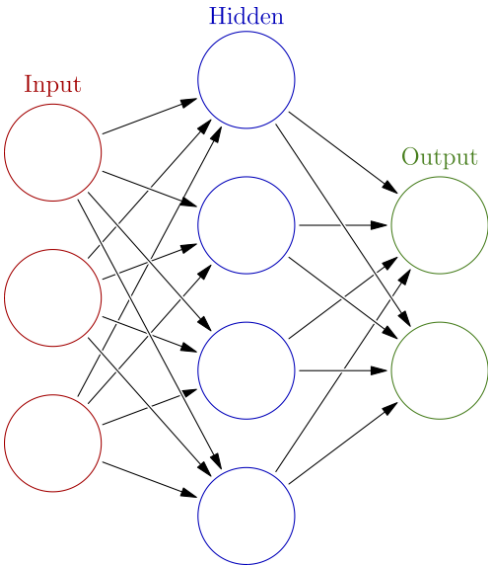



**Figure 2.6:** A simple ANN. Neurons are represented by circles. Input neurons are in red, hidden neurons are in blue and output neurons are in green. Arrows represent connections through which information propagates. Figure from Wikimedia Commons (commons.wikimedia.org/wiki/File:Colored_neural_network.svg).

The processing units propagate information forward, or not, depending on the input they receive and

an activation function. The most used activation function has been the logistic function introduced in Subsection 2.2.1, which is able to propagate the information only when a threshold is achieved in a manner similar to how neuron cells propagate nervous impulses. The intermediate (hidden) layer (or layers) of neuron units in an ANN enables a flexibility that is an extra advantage over the use of logistic regression. Additionally, unlike the logistic function, an ANN can have multiple output units, thus it is capable of handling more than a single classification problem with a single model. [25].

Che *et al.* applied *deep* neural networks to detect characteristic physiological patterns in clinical time series data from variables such as blood glucose level, blood pH and body temperature among others [5]. *Deep* neural networks are ANNs with multiple layers of hidden non-linear processing units. According to the authors, deep ANNs are especially suited to learn representations of human physiology due to the abundance of variables with complex nonlinear relationships. In their paper, Che *et al.* proposed an incremental training procedure that yields better results than the common full training of ANNs. The idea behind incremental training is to reuse the top layers of a network. Taking as example Figure 2.6, but assuming the existence of several hidden layers rather than a single one, a number of the few left-most layers of neurons would not be recomputed, it would instead be reused for larger networks, when larger networks are given the same inputs and initial subsequence in the time series. They show that this approach allows for faster training while retaining the classification capability of the network.

Che *et al.*'s method was evaluated using two clinical time series datasets from intensive care units. They observe that their method with the incremental training procedure both produces more effective classifiers and speeds up training when compared to the usual full training method of deep neural networks. As examples, they show that their method correctly established a causal association between high blood pressure, high heart rate and low pH and circulatory disease.

### 2.2.4 Production Rules

Production rules map characteristics of a situation to behavior or conclusions that should follow from that situation in the form of *IF condition THEN action* statements [19]. As an example, for the case of a diagnosis task, from some symptoms a production rule might output the name of a disease or recommend a treatment. The *action* may be arbitrarily complex and consist of several smaller actions. The process of decision with production rules is an iterative one in which rules are *matched* to available data, *selected* and *executed* until a conclusion is reached. Likely, there are stages of conflict resolution between rules [19]. Application of production rules may proceed in two ways: *forward chaining* or *backward chaining*. In forward chaining, the match-select-execute cycle starts by matching data elements to rules. The selected actions are executed and the results may serve as input to the next match-select-execute cycle. This process repeats until a conclusion is reached. Backward chaining works in reverse: departing from a conclusion intended to be *satisfied*, the algorithm works its way backwards to the data. If the

conclusion cannot be supported by the rules with basis on the existing data, the conclusion is rejected. Backward chaining is more suitable to inform the clinician if her decision is supported by the system and forward chaining is more appropriate to use data to point to possible clinical decisions [19].

MYCIN was a seminal CDS system (developed in the 1970's) based primarily on production rules with backward chaining [19, 36]. Its purpose was to advise practitioners in diagnosing and treating infectious diseases. Other production-rule based systems were developed later, notably the HELP system [16] at LDS Hospital in Salt Lake City and the Regenstrief Medical Record System at Indiana University [26].

### 2.2.5 Conclusion

In this section we introduced concepts of data mining, a field that uses methods from Statistics and Machine Learning to extract useful information from data. We reviewed reference applications of data mining methods to CDS systems. In particular, we focused on state-of-the-art work on use of Logistic Regression, Bayesian Networks, Artificial Neural Networks and a seminal system based on Production Rules. The reviewed methods are summarized in Table 2.2.

**Table 2.2:** Data Mining Approaches for Clinical Decision Support

| | Donzé *et al.* | Najjar *et al.* | Che *et al.* | MYCIN |
|---|---|---|---|---|
| Method | Logistic Regression | Hidden Markov Model (Bayes Network), Expectation Maximization/ Gaussian Mixture Distribution (clustering) | Deep Neural Networks (Artificial Neural Networks) | Production Rules |
| Category | Supervised Learning | Unsupervised Learning | Supervised Learning | Non-applicable |
| Purpose | Predict patient readmission | Find groups of similar patients | Detect time-dependent physiological patterns | Diagnosis, Suggest treatments |

The work by Donzé *et al.* provides an example of how Logistic Regression can be used to make predictions about patient outcome. The contribution by Che *et al.* shows how Bayes networks can be used to detect temporal patterns in time-series data, which could be used to predict the evolution of a patient over time. Donzé *et al.* and Che *et al.*'s are examples of work that aims to predict the occurrence of medical conditions. The MYCIN system is a case of success in diagnosis support that also suggests possible treatments. Finally, Najjar *et al.*'s paper describes a powerful clustering approach that can be used to find groups of similar patients, which is useful to guide further, more focused data analysis in a research context.

## 2.3 Information Visualization for Clinical Diagnosis

Appropriate visual representations of datasets have the power to improve the understanding of complex data and provide effective tools for physicians to find the information they need. Visualization in computers can be further enhanced with techniques of interaction. Interaction plays a very important role in computational Information Visualization (InfoViz) by providing users with the power to dynamically change the mapping of data (e.g., color, shape, size), the view of the mapping (e.g., zoom, pan, rank), or the scope of the data being visualized (e.g., search, filter) [35]. A choice of visualization techniques should support practitioners in finding the information they need quickly and effectively. However, research in medical InfoViz is lagging behind InfoViz research in other areas [1]. In this section we review some approaches used in InfoViz for CDS.

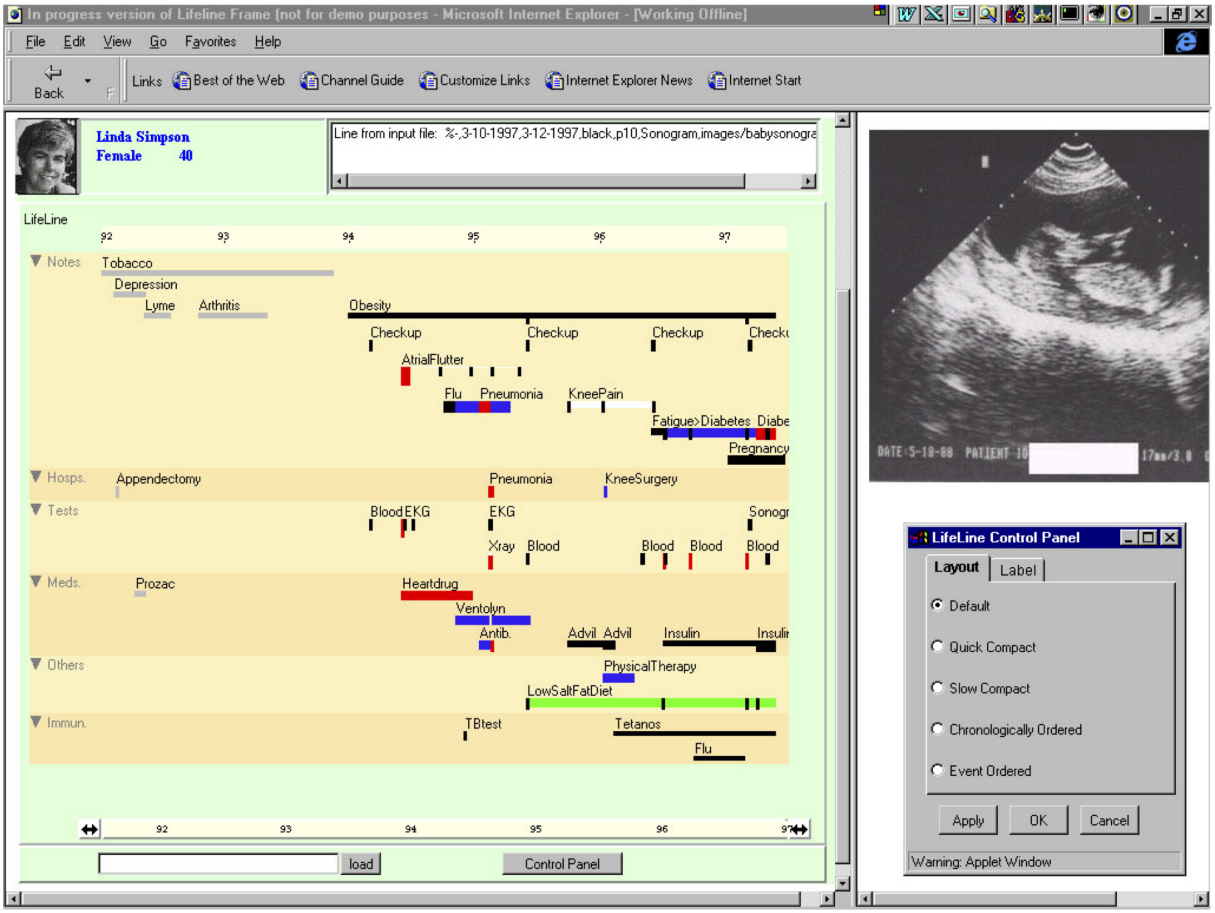



**Figure 2.7:** The LifeLines environment [31]. Image from Rind *et al.* [35].

Most InfoViz systems for medicine have been designed for exclusive analysis of either single or multiple patient data. This has been an aspect with major impact on the design of systems. Visualization of single patient data usually focuses on the patient's history, including information such as effect of

treatments, identification of significant events or missing data. These visualizations involve time-oriented data. On the other hand, visualization of multiple patient data tends to deliver less detail on each patient, enabling the discovery and exploration of relevant groups of patients [35]. In this section, we review four influential medical visualization systems. We will start by reviewing LifeLines [31], a seminal work for visualizing single patient history, and Web-based Interactive Visualization System (WBIVS) [30], a system to monitor data following lung transplant. While LifeLines has been a reference in patient history visualization [31, 44], WBIVS was a landmark in visualization of postsurgery data. Then we move on to describe LifeLines2 and Gravi++, two systems that implement techniques for visualization of multiple patient data. LifeLines2 focuses on event sequences in multiple records. Gravi++ clusters patients visually according to their attribute values.

### 2.3.1 Visualizing Personal Histories

LifeLines [31], shown in Figure 2.7, was a seminal work developed in the late 1990's for visualization of a patient's medical history. LifeLines provides a view with two panels. The left panel shows information from categorical data with color representing normal or abnormal states and presents line segments along the horizontal time axis that represent episodes in the patient's record. Line color, height and captions provide further information. Events of the same type (vaccines, medications, tests and other) are grouped along the vertical axis. Group areas may be folded and unfolded and line segments can be zoomed, panned and clicked for detailed inspection. The current date is marked with a vertical line (not visible in Figure 2.7), separating past events and events scheduled for the future [35]. Items on the left panel can be selected in order to present additional related information on the right upper area, such as X-ray images.

The WBIVS environment, shown in Figure 2.8, was designed to show numerical and categorical pulmonary data from a lung transplant patient over time. Numerical attributes are displayed in line plots and categorical attributes are displayed in matrix plots. The line plots are vertically aligned with the matrix plots so that they share the same horizontal axis, which represents time. When data is highlighted in one of the plots, the corresponding time period is also highlighted in other plots with color and horizontal lines.

The rate of change between two points can be visualized by selecting those points. As a result, the program draws a thick black line connecting the selected points. On mouse hovering over a point, WBIVS shows relevant measures in a tooltip. The user can also choose to display a five-day moving average to highlight trends.

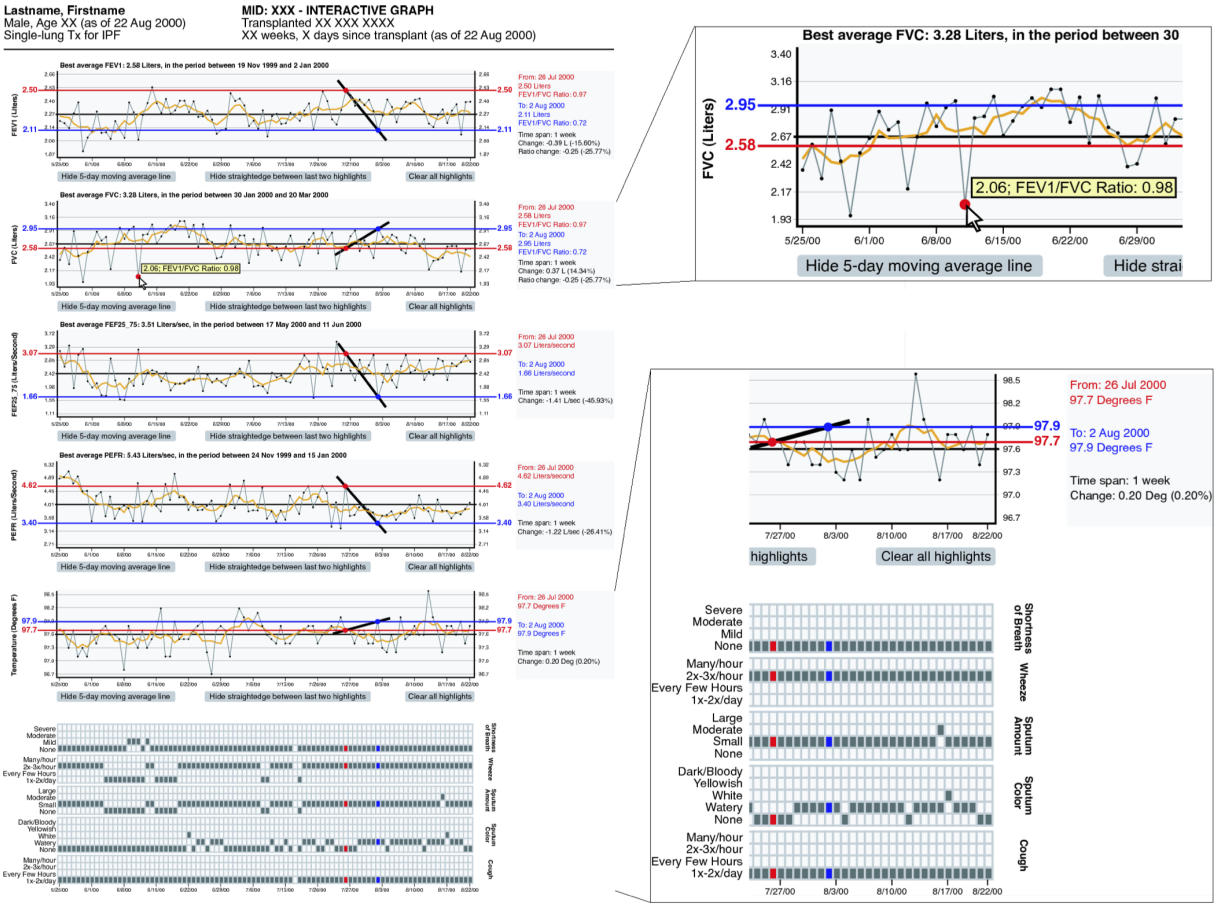



**Figure 2.8:** The WBIVS environment [30], which shows aligned line and matrix plots to follow the evolution of a lung transplant patient. Parts of the visualization are magnified on the right for a better view of the details. Image from Rind *et al.* [35].

### 2.3.2 Visualizing Patient Population Data

LifeLines2, shown in Figure 2.9 is a system designed to search and explore event sequences in temporal categorical data from multiple records of patient data [35, 42, 43]. Each record is shown in its own area with its own list of categorical attributes. Individual record areas are stacked vertically on a common horizontal time axis. Each attribute category is represented by an icon with a specific color. Icons representing the same attribute always fall on the same horizontal line.

LifeLines2 allows users to align the displayed records by a specific event. For example, if the user decides to align records by 'Radiology Contrast', all records will be aligned vertically by occurrence of the first event with that name. Records that do not have a 'Radiology Contrast' event will be filtered out. When the alignment is done, the time axis scale is transformed into an appropriate relative scale. This approach allows the user to discover trends in the timing of events relative to the alignment event. The system is also equipped with record sorting, filtering and search capabilities by attributes and derived measures. The authors also included a panel that displays a histogram representing the frequency of an

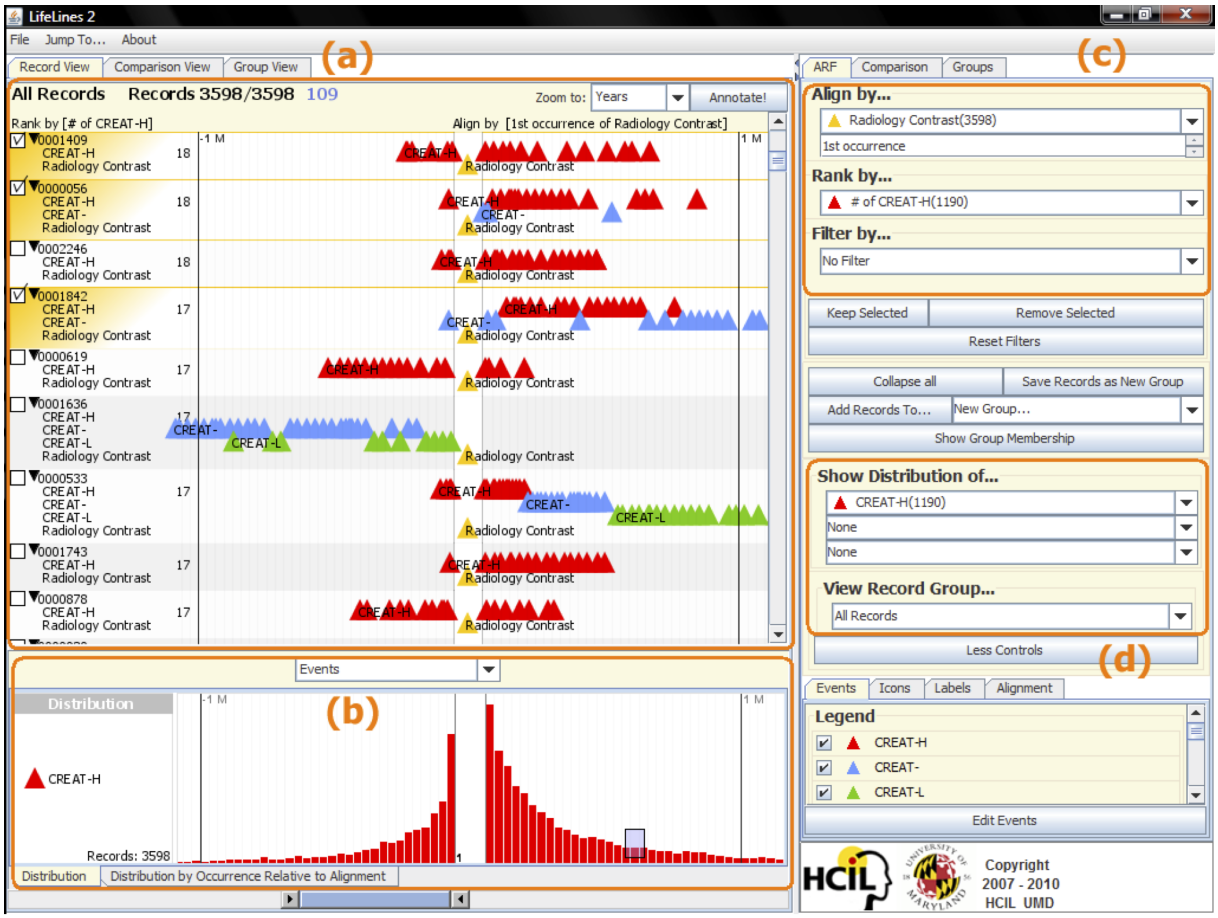



**Figure 2.9:** The LifeLines2 environment [42,43] for visualization of multiple records containing temporal categorical data. (a) shows a stack of records, (b) shows the histogram for a selected event within a chosen time window, (c) contains the controls for different operations and (d) shows controls for event distribution, grouping and other options. Image from Rind *et al.* [35].

attribute in a time window around the vertically aligned event. The histogram in Figure 2.9, for example, shows the distribution of occurrence of high level of creatine in a time window around the first 'Radiology Contrast' event.

Gravi++ [17] (shown in Figure 2.10) is an application that clusters patients according to the similarity of their ordinal categorical attribute values. Attributes and patients are represented by (different) icons. Attribute icons are placed near the limits of the visualization area. The positions of patient icons are determined by their attribute values: the higher the value of an attribute, the closer the patient's icon will be to that attribute's icon. Patients with similar attribute values are expected to cluster together and be easily identified visually [35].

Icon size can also encode additional patient attributes such as age or weight. Icon color can encode gender or therapeutic outcome. Attribute values for a specific patient are shown with circles around the attribute's icon upon selection of a patient [35].

Gravi++ supports visualizing the evolution of patients through time with an animation controllable by the user. The user can play the animation and watch the patient icons move in the visualization area according to the variation of their attribute values [35].

**Figure 2.10:** Gravi++ [17]. The squared icons represent attributes and the circular icons represent patients. Three patients are represented at several time points during an animation. Circles around the attribute icons represent attribute values for the patients. Users can choose patients and attributes on the left and select options on the right. Image from Rind *et al.* [35].

### 2.3.3 Conclusion

The implementation of InfoViz techniques is imperative to enable the exploration of data and analysis results by the end users. In this section, we reviewed reference work in InfoViz in clinical systems, describing four influential medical visualization systems from the research literature. These systems are summarized in Table 2.3.

Clinical InfoViz systems have traditionally been designed to support either visualization of single patient data or visualization of multiple patient data. LifeLines and WBIVS are reference systems to visualize the evolution of a patient's medical parameters over time, hence helping the physician to un-

**Table 2.3:** InfoViz Systems for Clinical Decision Support

| | LifeLines | WBIVS | LifeLines2 | Gravi++ |
|---|---|---|---|---|
| System type | Single patient | Single patient | Multiple patients | Multiple patients |
| Attribute types | Categorical | Numerical and Categorical | Categorical | Ordinal Categorical |
| Main techniques | Aligned time lines, Zooming, Panning | Line plots, Matrix plots | Aligned time lines, Histogram, Zooming, Panning, Sorting, Filtering and Search | Positioning in 2D area, Time-dependent animation |
| Purpose | Visualize events in patient's history | Visualize numerical and categorical time series | Explore/compare patient histories | Find groups of similar patients |

derstand the patient's history and present condition. LifeLines2 and Gravi++ allow physicians to explore collections of patient histories: they provide means to visualize data for groups of patients using techniques such as histograms and time lines (LifeLines2) or spatial organization (Gravi++), which is an effective means to enable discovery of patterns in the evolution of patient conditions that may point to typical strategies of therapeutic failure or success.

# 3

# The Umedicine Application

## Contents


3.1 Requirements . . . . . 29
3.2 Architecture . . . . . 30
3.3 Implementation . . . . . 38
3.4 Conclusion . . . . . 39

Umedicine is a software application that aims to support Urology practitioners in clinical practice and medical research. On one hand, it aims to provide an easy means of collecting and accessing individual patient information during medical appointments. On the other hand, it aims to be a tool for physicians to explore their patient population data and potentially discover new relevant medical insights. With these general aims in mind, a set of requirements was established. The development of the application was already ongoing when the work described in this thesis started and some of the requirements listed below had been addressed. However, new requirements were produced under the guidance of the team's physician and the application needed to be extended to accommodate these new requirements.

Section 3.1 lists the complete set of requirements, highlighting those that are addressed in the present work. Section 3.2 describes the initial architecture of the application, with details about the front-end, the back-end server and the design of the relational database and Section 3.3 describes the technology used for the development of the application. Finally, Section 3.4 provides a conclusion to the chapter.

## 3.1 Requirements

This section lists the complete set of requirements for the Umedicine application, including both requirements addressed before the present work and the requirements addressed in this thesis.

The application has four types of users: administrator physicians, non-administrator physicians, patients and administrative personnel (clerks). Administrative personnel is expected to help patients to start using the application. Each user has access to different functionalities, depending on their type, as listed below.

1. Functionalities for non-administrator physicians:
   (a) Add new patients to the system, as well as their personal and medical information;
   (b) Search for the personal or medical information of a specific patient by name or process number;
   (c) Have full access to patient personal and medical information.
2. Functionalities for administrator physicians:
   (a) All requirements of non-administrator physicians;
   (b) Add new clerks and physicians (both administrator and non-administrator) to the system.
3. Functionalities for patients:
   (a) View own personal information;

(b) View own exam results and history;

(c) Fill medical questionnaires.

4. Functionalities for clerks:

   (a) Add new patients to the system;

   (b) Add limited patient personal information (name, birth date, contact and profession);

   (c) View limited patient personal information;

5. Security requirements:

   (a) Ensure that each type of user can only use the functionalities and have access to information as described in the requirements above;

   (b) Enforce minimal password strength requirements;

   (c) Provide to all users the possibility of changing passwords;

   (d) Ensure data confidentiality in communications between clients and server.

6. Data mining requirements:

   (a) Find groups of similar patients with basis on their personal and medical history;

   (b) Enable exploration of the characteristics of the patients in each group;

   (c) Explore the effect of treatments on diseases.

The Umedicine application, at the beginning of this work, fulfilled requirement groups 1 and 3: it allowed only for one generic physician and multiple patients as users. The remaining of this chapter describes the application and how these requirements had been addressed before the start of the work for this thesis. In this thesis, we address requirement groups 2, 4, 5 and 6: we added the possibility of having multiple physicians using the system, the distinction between administrator an non-administrator physicians, the *clerk* user type and the security and data mining requirements.

## 3.2 Architecture

Umedicine has a client-server style architecture with three main components (Figure 3.1): (i) a relational database, (ii) a back-end server, and (iii) a front end available to the user through a web browser client. Users interact with the system through a web browser, which submits HTTP requests to the server. The server returns an HTML page. Data may be asynchronously requested from the server through Asynchronous JavaScript and XML (Ajax) requests made by the browser. To respond to Ajax requests, the server queries a relational database and returns the data to the client in the form of JSON

objects. The use of Ajax allows to exchange only new necessary data to fill the page, minimizing data transfers from the server and improving user waiting time. The server is developed in Java with extensive use of Spring Framework[1]. Web pages are generated with the JavaServer Pages (JSP) and Apache Tiles[2] technologies — details are given in Section 3.3.

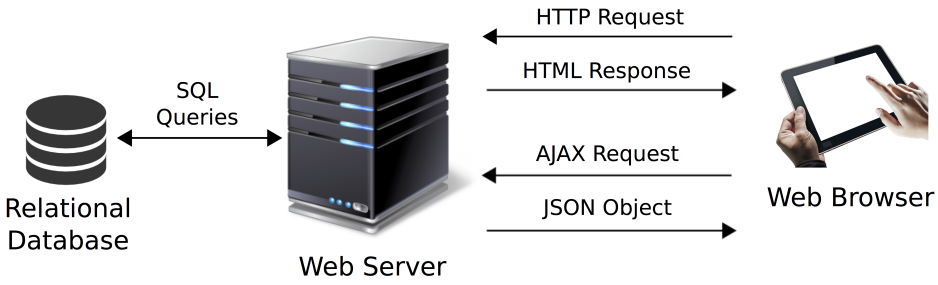



**Figure 3.1:** Architecture of the Umedicine system.

### 3.2.1 Front end

The system provides distinct interfaces for the physician and the patients. The physician's area (Figure 3.2 shows the physician's entry page) provides a button to view patient information and a button to add a new patient to the system.

**Figure 3.2:** Physician homepage.

Upon choosing the 'view patient' button, the physician is brought to a search page where existing patients can be searched using their names or process numbers (Figure 3.3).

On this page the physician can find a summary of the information about the patient corresponding to the inserted process number (Figure 3.4). The user can navigate to the patient's information page (Figure 3.5) by clicking on the corresponding row.

The patient information page (Figure 3.5) is organized in 6 parts:

- Personal information: area where physicians can view and modify information such as name, birth date, contact and habits of the patient;

[1] https://spring.io/
[2] https://tiles.apache.org/

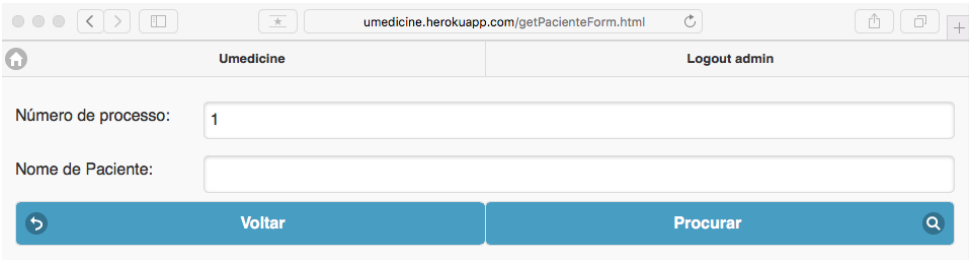



**Figure 3.3:** Patient search page available to the physician.

**Figure 3.4:** Umedicine's patient search results page.

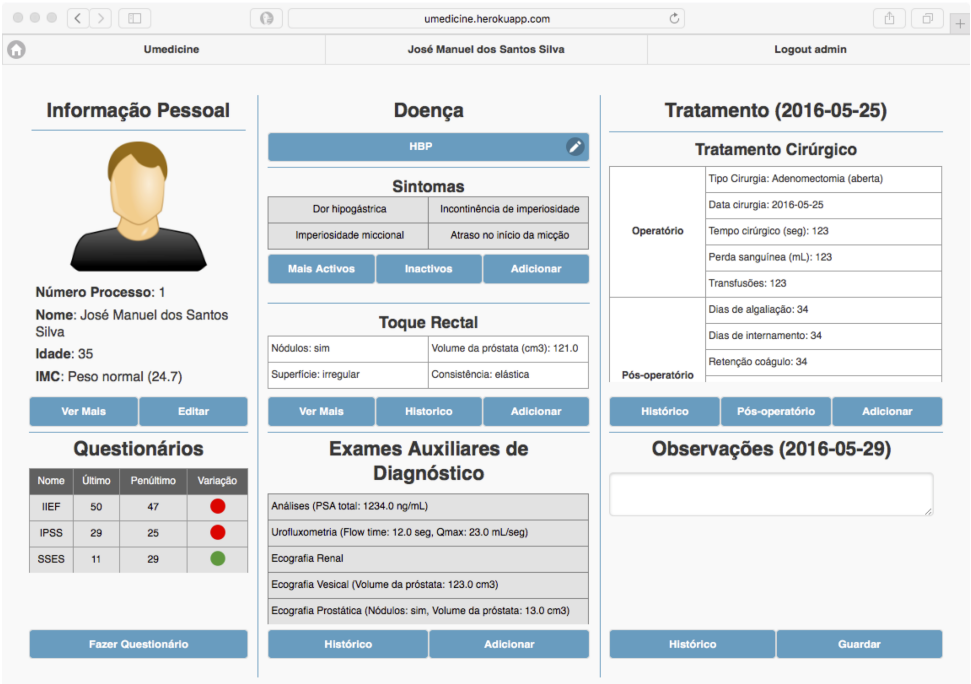



**Figure 3.5:** A patient's information page as viewed by a doctor using Umedicine.

- Disease: area where physicians can select a medical condition and view or modify information about symptoms; following the development team's physician advice, this area also includes the

rectal examination information; users can view symptom and rectal examination histories and add new symptoms and examination results;

- Treatment: this area displays information about ongoing treatments and provides means to view treatment history and add new treatments;

- Questionnaires: this area provides access to 3 diagnosis support questionnaires that a physician can have a patient answer during a consultation; these questionnaires are used to compute scores that represent the severity of a patient's condition; Umedicine currently supports three relevant standard questionnaire-based scores: International Index of Erectile Function (IIEF), International Prostate Symptom Score (IPSS) and State Self Esteem Scale (SSES);

- Diagnosis support examinations; this area shows the last results for several kinds of medical examinations and laboratory analyses; the user can add new results and access the patient's examination and analysis histories;

- Notes: physicians can write a text note about the patient's condition in the day of the consultation and view notes from previous consultations.

Figure 3.6 shows an example of a patient's treatment history view. Treatments are organized in a table where each row corresponds to a treatment and shows the treatment's kind and date. Treatments are listed in the table from the latest, at the top, to the oldest, at the bottom. For details about a treatment, the user can click the corresponding line.

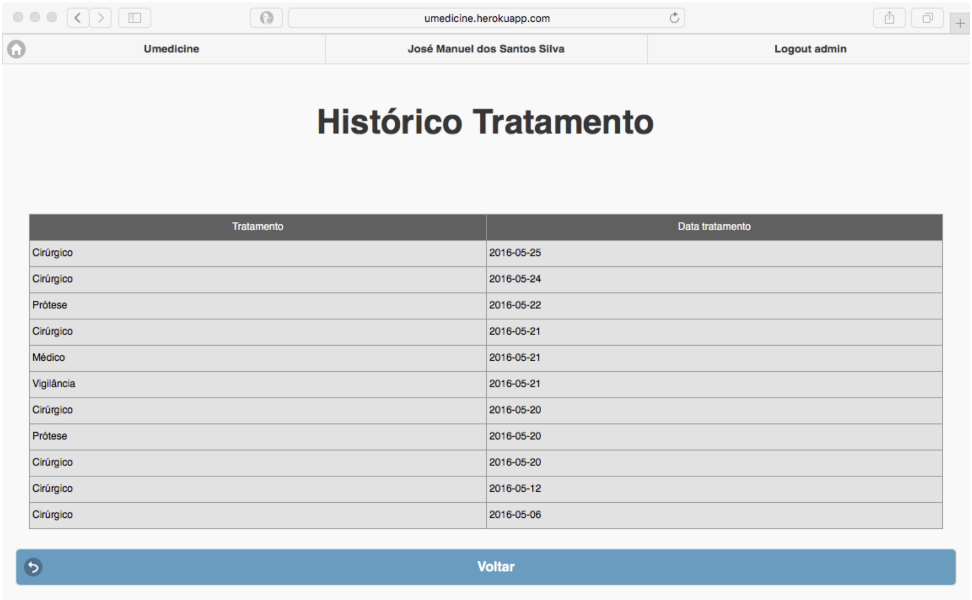



**Figure 3.6:** View of a patient's treatment history in Umedicine.

The patient information page (Figure 3.7) has an area on the left where they can view their current photo in the system and alerts requiring attention. The middle area shows their current personal information. On the right, the patient can view information about examinations, prescriptions and past surgeries. At the bottom, there are buttons to edit the patient's personal information or answer a diagnosis support questionnaire.

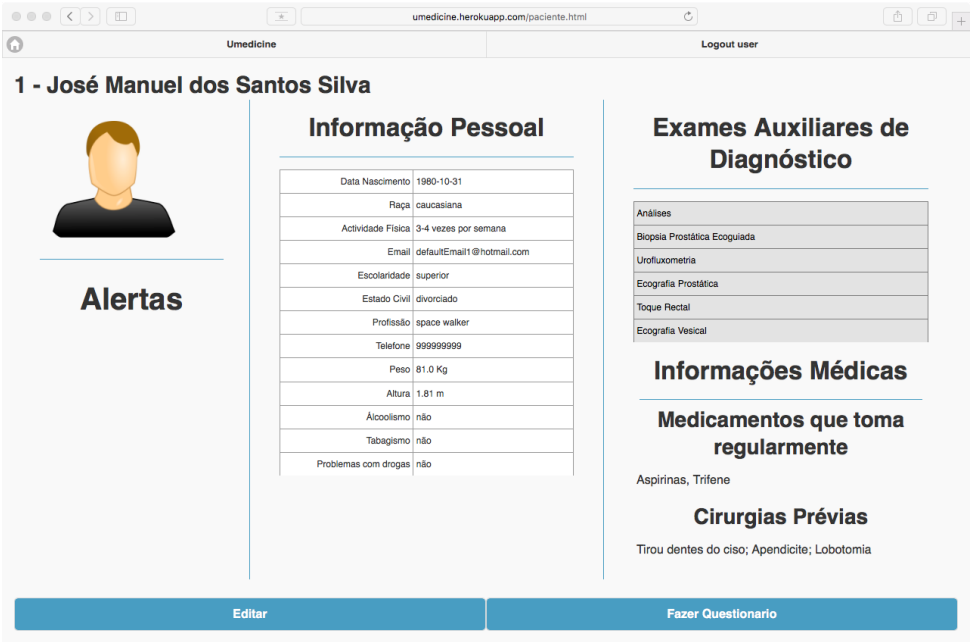



**Figure 3.7:** A patient's information page.

The version of Umedicine described in this chapter does not provide an interface for administrative personnel, and does not have separate administrator and non-administrator physicians, which have access to different functionalities in some cases. These are new contributions of the present work described in Chapters 4 and 5.

### 3.2.2 Back-end Server

The back-end server replies to the requests sent by the browsers, performs operations on the data and stores or retrieves data as necessary from the Database Management System (DBMS). It is responsible for ensuring that each user has access to, and only to, the data it needs, and for loading the data or web pages requested by the users.

The server is implemented as a Java application with extensive use of the Spring Framework ecosystem. It follows the Model-View-Controler (MVC) design pattern and is composed of three main layers — from top to bottom: controllers, services and Data Access Objects (DAOs) (Figure 3.8).

*Controllers* are the components responsible for the interaction with the clients. They receive client

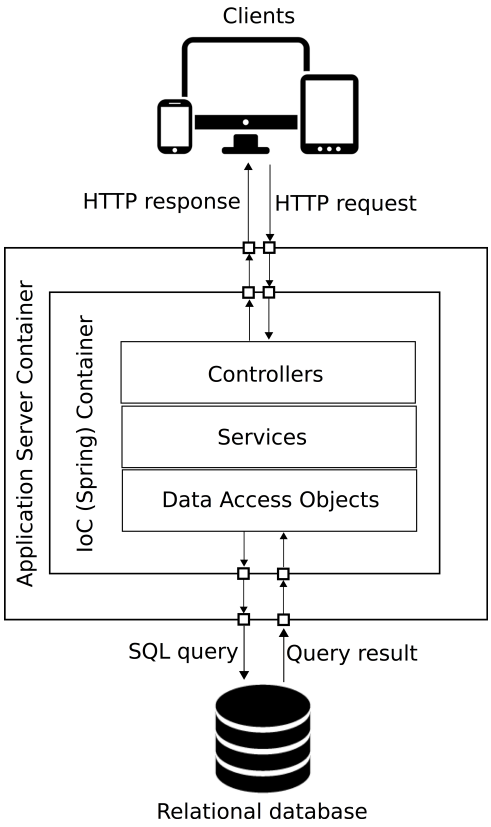



**Figure 3.8:** Umedicine's server component architecture.

requests, invoke the appropriate business logic methods and send a response if required. To interact with the server, clients need to log into the application. A simple login configuration that authorizes physicians and patients to access their respective resources was in place in the initial prototype, although without separate sessions, password hashing or client-server communication encryption. Once users are logged in, they can request web pages or data from the controllers.

Controller code, as well as service and Data Access Object (DAO) code, is, as much as possible, organized to maximize separation of concerns. In other words, a controller is written to support a specific set of coherent functionalities. For example, the code responsible for handling requests related to user account creation is in methods of the class *UserController*; requests from pages where users fill questionnaires are handled by the *QuestionnaireController* class; and so on.

Any business logic-related processing of the data retrieved from the database or sent by the clients takes place in the *service* layer. Service methods are invoked by controllers (or other services) and return data that controllers use to respond to the requests issued by the client. Service code is also organized according to functionalities. *UserService*, for instance, contains the code that related to user account creation, performing duties such as invoking the DAO methods that store user data in the database (after doing any required data processing). In a similar fashion there is a *PatientService* class

that handles patient personal data, a *QuestionnaireService* class that handles questionnaire data, and so on. Services also organize data retrieved from the database via DAOs into objects suitable to be used by the controllers, which may return it to the browser.

The bottom layer, DAOs, contains the code that enables the interaction between the server and the database. This code uses the Java Database Connectivity (JDBC) Application Programming Interface (API) and includes the SQL queries that insert new rows or update existing rows and the queries that retrieve data from the database tables. Information retrieved from the database is converted to Java objects and passed on to the upper service layer for further processing.

### 3.2.3 Relational Database

The Umedicine architecture encloses a relational database to store three kinds of data persistently: (i) clinical data inserted by patients or physicians; (ii) user authentication data; and (iii) metadata concerning a specific medical field. The current version of the application has a database that is set up for Urology but its schema is designed to be adaptable to other medical specialties. For this end, the metadata that corresponds to domain knowledge – drug names, disease names, symptoms, medical examinations and diagnosis support questionnaires – is stored in the database rather than in the back-end server and is accessed when the application loads. With this approach, different domain knowledge from other medical specialties can be replaced into the database and plugged into the server with minimal modification of server and client code (ideally no modification will be needed whatsoever). This approach also enables switching easily from metadata in Portuguese to metadata in another language.

To illustrate the Umedicine database model, we present the subset of the database relational schema that model medical examinations. Primary keys of relations are underlined and foreign keys are specified by FK:

*Clinical metadata:*

ExaminationType (typeName, subTypeName)
Parameter (paramName, paramUnit, paramType)
ExaminationParameter (typeName, subTypeName, paramName)
    typeName, subTypeName: FK (ExaminationType)
    paramName: FK (Parameter)

*Clinical data:*

Patient (pNumber)
PerformedExamination (pExamID, pNumber, typeName, subTypeName, date)
    pNumber: FK (Patient)
    typeName, subTypeName: FK (ExaminationType)
PerformedExaminationValue (pExamID, paramName, value)
    pExamID: FK (PerformedExamination)
    paramName: FK (Parameter)

There are two types of data modeled in this relational schema: (i) clinical metadata and (ii) clinical data. ExaminationType, Parameter and ExaminationParameter are relations that model metadata. This metadata is accessed when the application starts and is used to determine the information that is shown in the user interface. The ExaminationType table stores all the possible types and subtypes of medical examinations. The tuples: *<Urofluxometry, - >* and *<Blood Tests, Liver function>* are examples of records stored in this table. Urofloxometry is a medical examination by itself while there are several kinds of blood tests so the subtype field is required. The Parameter table stores the existing parameters, their measurement units and expected types, i.e., String, Integer, Float or enumerated (in the case the type is ENUM, there is another table not represented here that stores the possible values). The tuples: *<Creatinine, mg/dl, float>* and *<Volume, ml, float>* are examples of records of the Parameter table. The table ExaminationParameter stores the correspondence between an examination and the its parameters. In accordance with the previous examples, the following tuples: *<Urofluxometry, -, Volume>* and *<Blood Tests, Liver function, Creatinine>* are examples of tuples of the ExaminationParameter table.

The tables Patient, PerformedExamination and PerformedExaminationValue store clinical data collected through the usage of the application. In particular, Patient stores data about patients (*<P1>* and *<P2>* are examples of tuples); PerformedExamination stores the examinations performed by each patient ( *<PE1, P1, Blood Tests, Liver function, 27/6/2016>* and *<PE2, P2, Urofluxometry, -, 28/7/2016>*

are examples of tuples); and PerformedExaminationValue stores data about the filled parameters of the examination performed by a patient (examples of tuples are: *<PE1, Creatinine, 1.02>* and *<PE2, Volume, 157>*).

Figure 3.9 presents the application screen shown for the examination parameter Volume that is of type float and is measured in milliliters. The metadata stored in the database table Parameter, in particular, supplies the (meta)data to be shown in the forms presented in the front-end user interface. The clinical data that will be filled in by the application users is then stored in the database tables that store clinical data and is further available for future visualization and analysis.

**Figure 3.9:** Parameter Volume of type FLOAT and unit milliliter (mL)

## 3.3 Implementation

The Umedicine application makes use of a variety of technologies based on the Java language[3]. In particular, it makes extensive use of the Spring Framework[4] ecosystem and its Dependency Injection technique. When using Dependency Injection, creation and location of objects in the application is a duty of the framework. When an object needs another object of a certain type, it is initialized and provided — *injected* — by the framework. This dependency object can be reused as many times as needed by other objects in the application. Therefore, the responsibility of creating and wiring the objects has been *inverted*: it is no longer responsibility of an object to create the objects it depends on [11, 47]. This concept is known as Inversion of Control (IoC).

Dependency Injection brings advantages to the design of the application. Since the framework is responsible for wiring the dependency to a dependent object, a single dependency object may be created and reused by different dependent objects. It encourages implementations of dependents and dependencies that fulfill their contracts but reduce coupling. Testing classes also becomes easier because under the Dependency Injection design pattern the state of tested objects should be independent of the state of other objects. On the other hand, ambiguities may result from object wiring no longer being stated explicitly, potentially causing errors. Dependency injection may also add unneeded complexity to an application and even coupling of the application's code to the framework's [20]. Umedicine benefits from Dependency Injection's advantages in a manner that outweighs its disadvantages: it reuses framework-managed objects extensively, avoiding the overhead of object creation, destruction and a potential increase of memory usage.

[3] https://www.oracle.com/java/index.html
[4] https://spring.io/

The use of Dependency Injection means that the objects of the Umedicine application live inside an IoC container managed by Spring Framework. In turn, this container lives inside an application server such as Tomcat[5], capable of receiving and responding to HTTP requests. When a request is received by the application server, the request is forwarded to the IoC container, which determines which servlet, a controller, is mapped to the Uniform Resource Locator (URL) associated to the request. The response from the controller is forwarded by Spring to the application server to be sent to the client — Figure 3.8.

Most Umedicine web pages are built dynamically with JSP[6], a technology that allows inserting Java code inside HyperText Markup Language (HTML) pages by making use of special tags. With JSP, webpages can be built dynamically by the application, enabling dynamic addition of specific data, HTML and JavaScript code to the page. Additionally, JSP technology also enables reutilization of front-end code through definition of custom (developer-defined) tags or fragment files. Custom tags wrap code that can be reused by declaring them in a JSP page; fragment files contain code that is directly inserted in a JSP page where the fragment is imported. When the application receives a request for a JSP page, the corresponding JSP is compiled to a Java class that prints the HTML file with the intended HTML, JavaScript code and data to be returned to the browser.

The elements of the interface are designed and implemented to allow appropriate scaling in desktop computers and mobile devices, mostly without the need to scroll, with extensive use of the JQuery Mobile[7] technology. Ajax calls are used to avoid that the user experiences long waiting times, as much as possible.

The relational database is implemented in Oracle's MySQL database management system[8].

## 3.4 Conclusion

The initial Umedicine prototype implemented a set of architectural decisions and an interface design that shapes the enhancements proposed in the present work. However, this version of the application didn't meet the security standard required to handle medical data, it didn't support simultaneous use by several physicians, it didn't provide access to administrative personnel (clerks) and there wasn't a distinction between administrator and non-administrator physicians. Additionally, it didn't provide any data analysis capabilities beyond visualization of crude patient information stored in the database. The following chapters (Chapters 4 and 5) describe enhancements to the Umedicine application that address these requirements.

[5] http://tomcat.apache.org/
[6] http://www.oracle.com/technetwork/java/javaee/jsp/index.html
[7] http://jquerymobile.com/
[8] https://www.mysql.com/

# 4

# Umedicine: Security Module

## Contents


4.1 User Types . . . . . 45
4.2 Authentication . . . . . 47
4.3 Database Schema . . . . . 49
4.4 User Interface . . . . . 50
4.5 Implementation Details . . . . . 51
4.6 Conclusion . . . . . 53

The Umedicine application is intended to be a system for storage and analysis of patient data. The version of the application presented in Chapter 3 fulfills several of the requirements stated in collaboration with a Urology practitioner: it provides a means to store, search and visualize individual patient data, and a means for patients to keep their personal information up to date and view their medical history, including examination results. However, Umedicine from Chapter 3 does not fulfill the remaining requirements. In this Chapter, we describe a new version of the application, upgraded to meet these requirements.

Likely, in everyday clinical practice, patients are assisted by more than one physician. It is also likely that tasks such as patient account creation are not performed by physicians but by hospital administrative staff. Thus, we recognized the need to create an extra user type in the application, the *clerk* user type, which is intended to respond to the need to have administrative personnel creating patient accounts and adding certain patient personal information to the system. The creation of the *clerk* user type raised the question of whether all physicians should be able to add clerk accounts to the system. This question was part of the larger task of providing Umedicine with a mechanism to add new users, which was lacking (Requirement groups 1, 2 and 4). To address this issue, it was decided to have two kinds of physician users: physicians with administration privileges and physicians without administration privileges. The difference between the two kinds of physicians is that only those with administration privileges are allowed to create new accounts for (both kinds of) physicians and for clerks. To keep assistance to patients as efficient as possible, all physicians and clerks are able to create patient accounts. Unlike clerks, however, all physicians are allowed to edit any patient information.

In an application intended to serve a potentially large population of users, users must be confident that the information they edit or view cannot be accessed without the right privileges, and even less by third parties using the same network (Requirement group 5). This is especially sensitive when it comes to medical data: patients rely on healthcare providers to keep their personal and medical data confidential. Hence, in a medical information system, security is an issue of paramount importance. In particular, in Umedicine, we must guarantee that only physicians have access to a patient's information, besides the patient themselves.

Finally, Umedicine is intended to provide tools to support tasks beyond accessing a patient's personal information and medical history: it should provide tools to explore the wealth of data gathered by the application to support diagnosis and research through the use of data mining techniques (Requirement group 6). This requirement is addressed in Chapter 5.

In this chapter, we describe our solutions for the security and user management requirements outlined above. Figure 4.1 illustrates the architecture of the new version of the Umedicine application. We added an authentication mechanism, which guarantees that users access (only) the functionalities intended for their user types. We added session management to the server, which isolates data ex-

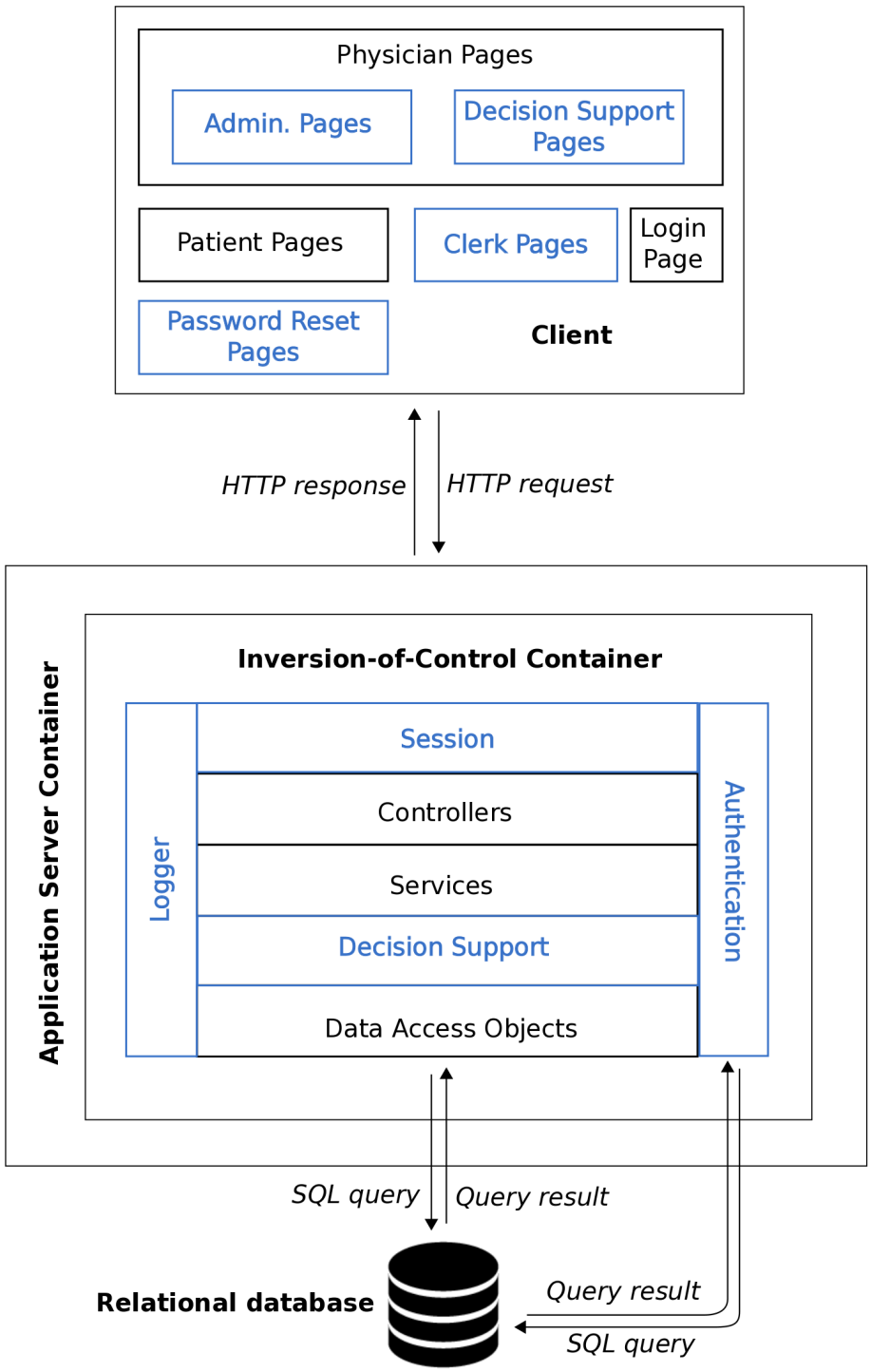



**Figure 4.1:** Architecture of the Umedicine system presented in this chapter. The modules in blue are new. The *Decision Support* and *Decision Support Pages* modules are described in Chapter 5.

changed by each client with the server from other clients. We implemented password generation and reset mechanisms that are available to all users. We also implemented the *administrator physician*, *non-administrator physician* and *clerk* user types, and provided each with an interface designed according to the corresponding functionalities.

In the remainder of this chapter, we describe the new version of the Umedicine application, by explaining the new additions and the differences with respect to the previous version from Chapter 3. We begin by explaining the user types in Section 4.1, then we explain the authentication mechanism in Section 4.2. We proceed to describe the database schema in Section 4.3 and the user interface in Section 4.4. Then we provide a section describing the technologies used and the implementation details, Section 4.5, and, at the end, a conclusion for the chapter in Section 4.6.

## 4.1 User Types

Two critical features that were missing in Umedicine were the possibility of adding physicians to the system and allow administrative staff (clerks) to create patient accounts and add limited patient personal information. To solve this problem, we define four *user types* with access to different features of the application:

- Non-administrator physician: this type of user can add users of type patient and can view and edit patient information.
- Administrator Physician: in addition to having access to the same features as non-administrator physicians, this type of user can add non-administrator physician, administrator physician and clerk users to the system.
- Clerk: a clerk user can add patient users to the system, as well as the patient's name, contact information and profession.
- Patient: a patient user can add and view limited information about themselves to the system.

Two kinds of physician users can now be created: administrator physicians, who can add other physician and clerk accounts, and non-administrator physicians, who have access to the same features that administrators have, except creating physician and clerk accounts. Moreover, the development team's physician expressed the need to allow other clinical staff to create accounts for patients and introduce limited patient personal information. To address this need, the *clerk* user type was created.

The creation of physician accounts is done through the form shown in Figure 4.2. The only information that is required about the new user is an email address. Upon submission of the form, a password is

generated for the new user. If the email and password are successfully written to the database, a message containing the password is automatically sent to the new account's email address. The creation of a clerk account is similar, but without the option to grant administration privileges.

**Figure 4.2:** Form to create a physician account.If the checkbox is checked, the physician will be an administrator. The form to create a clerk account is the same except for the checkbox, which is absent.

To create a new patient account, the physician or clerk user submits the form in Figure 4.3. If the submission is successful and the user creating the account is a clerk, the clerk is asked if they want to introduce patient information (Figure 4.4), potentially speeding up the availability of this information to physicians. To fill this form (Figure 4.5) clerks only need to ask patients their name and birth date. Optionally, they can also ask the patient's telephone number, email address and profession. Clerk users are restricted to creating accounts for patients and introducing personal information for them as part of the process. They cannot access medical information.

**Figure 4.3:** Form where clerk users create patient accounts. For patients a password of no less than 8 characters must be provided when the account is created.

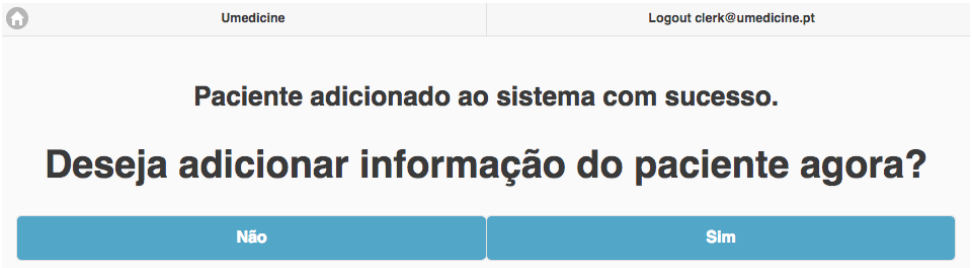



**Figure 4.4:** After creating a patient account, clerks are asked if they want to introduce information about the patient.

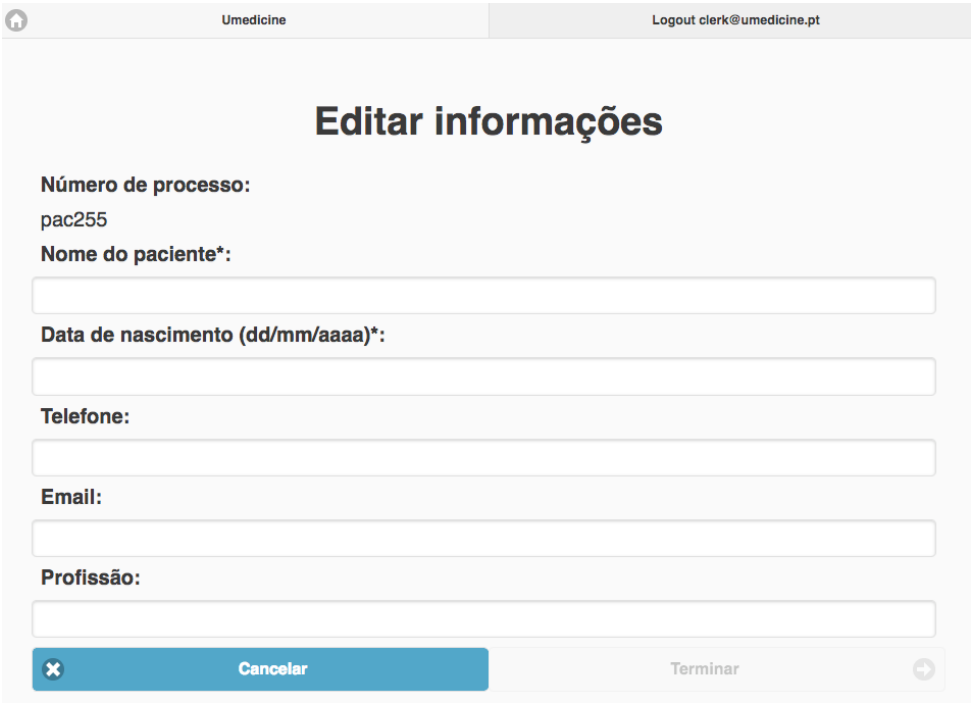



**Figure 4.5:** Form where a clerk can add limited personal information about a just-added patient user.

## 4.2 Authentication

In a medical information system, patient data confidentiality is of paramount importance. In a client-server system like Umedicine, where users interact with the application through a web browser that constantly exchanges data with a remote server across a network, software engineers must guarantee that the information traveling in the network cannot be seen or modified by third parties.

Umedicine, as described in Chapter 3, did not provide connection encryption or session isolation between clients making requests from the server at the same time. This means that any patient information kept in the server memory would be shared between connections and the information would be transmitted over the network as readable plain text. To address these problems, we enabled the *Transport Layer Security (TLS)* cryptographic protocol [8], enforcing encryption of data transmitted between server

and clients, and configured the application to open a *session* for each client, isolating in-memory user data from other connected clients.

**Figure 4.6:** Umedicine's new login page, which has a button that navigates to a form where users can request a new password (Figure 4.7).

**Figure 4.7:** Page where users with an email address can request that a new password is sent to them.

Password generation, change and retrieval are other security-related features that were added to the current version of the system. The way passwords are generated depends on the kind of user. For physicians and clerks, a strong password is automatically generated when the user account is created in the system. This password is emailed to the user. Physicians and clerks must use their emails as usernames and the provided passwords to log into the system. For a new patient, a password of at least 8 alphanumeric characters must be introduced as input at the moment of creation of the account. This password is handed to the patient manually, for example in paper, so that he can log into the system using their medical process number as username. Since patients are handed their passwords in a less secure way, they are asked to change their passwords the first time they log into Umedicine. Users with an email can also request a new password (if, for instance, they forget their current one). They can

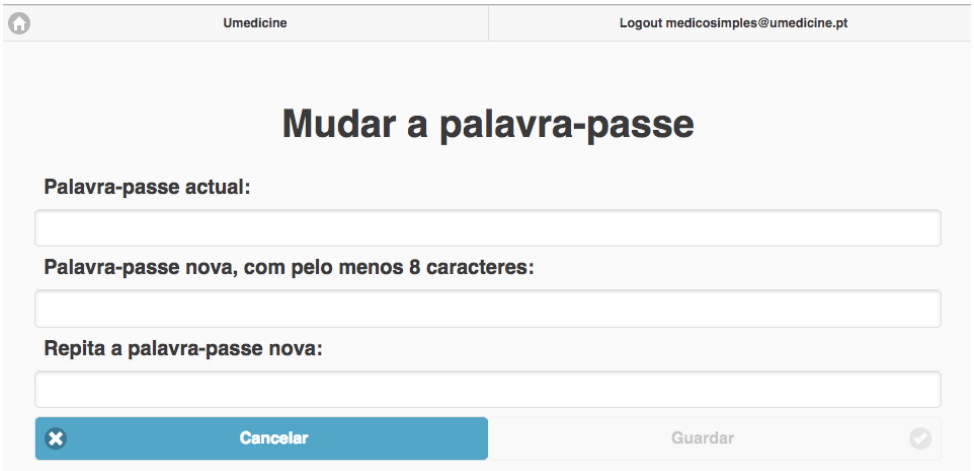



**Figure 4.8:** Umedicine's password change form, which includes a checkbox that displays the password as text on the screen when checked.

access the form for this purpose (Figure 4.7) from the login page (Figure 4.6). An user only needs to introduce their email to trigger the generation of a new password. The newly-generated password is then sent to the user by email. All users may change their password whenever they wish, to a string with at least 8 alphanumeric characters, using the form in Figure 4.8, accessible from their homepage.

## 4.3 Database Schema

The tables that store user information were modified to accommodate the changes introduced by the new user types and authentication mechanism. User information involves storing login credentials — usernames and passwords — and the type of each user. This data is stored in two tables, Users, which holds usernames and passwords, and UserTypes, which holds the type (or types) of each user in the system. This is the subset of the database relational schema that models user authentication information. Primary keys of relations are underlined and foreign keys are specified by FK.

*Authentication:*

Users (userid, username, pNumber, password, active)
    nprocess: FK (Patient)
UserTypes (*userid, type*)
    userid: FK (Users)
Patient (pNumber)

The possible types are *physician*, *patient*, *clerk* and *administrator*. Note that each user may have more than one type. In practice, currently, only physicians may have two types: they may or may not have the *administrator* type as well. Each username must be unique, have a corresponding password and have a *true* or *false active* value. If the *active* value of a user is *false*, the user is not granted access to the system. Patient users also have a unique clinical process number, stored in attribute *pNumber* of table Users, which is a foreign key to table Patient. Each tuple in UserTypes also holds a foreign key to table Users. The tuples *<1, aPhysician@somewhere.com, -, abcdefgh, 1 >* and *<2, aPatient, aPatient, ijklmnop, 1 >* are examples of tuples of table Users. (In the actual database, passwords are hashed.) A physician's username is their email address, while a patient's username is also their identifier from the clinical facility, which is used as the primary key of the *Patient* table, which holds their personal information. For physicians, the *pNumber* attribute is not required. The tuples *<1, 'physician' >* and *<2, 'patient' >* are examples of tuples of table UserTypes that associate a user type to the tuples indicated above from table Users.

## 4.4 User Interface

Upon a successful login by any user, the system fetches the user type information from the database and redirects the user to the corresponding homepage. From there, the user can only navigate to pages they will be granted access to.

When an administrator physician logs into Umedicine, their homepage (illustrated in Figure 4.9) shows six buttons. The top-left button navigates to the pages where the physician can access and modify information about existing patients; the top-right and the two middle buttons lead to the creation of accounts for patients, physicians and clerks; the bottom-left button leads to the pages dedicated to data analysis (described in Chapter 5) and the bottom-right button navigates to the 'change password' page (described in Section 4.2). The middle line, with the buttons to create physician and clerk accounts, is absent in the homepage for non-administrator physicians. The homepage of clerk users shows two buttons: one to add a patient user and another to change the clerk's password.

Patient data is added to the system through forms that are designed to fit in a window, without scrolling. When more room is needed, the remaining input fields are moved to a different view to which the user may proceed after filling the current one. The forms provide features to guide users when filling them in: the button that allows the user to submit the form is disabled until all the mandatory fields are filled correctly; mandatory fields are marked with a star (*); if the user navigates away from a field and the information entered does not meet the requirements, the application highlights the field with a red border and pops up a message warning the user about the problem. When all mandatory fields and any filled non-mandatory fields are filled correctly, the submission button is enabled to allow the user to

**Figure 4.9:** Homepage of administrator physicians. The page has buttons that lead to pages where physicians and clerks can be added to the system. The homepage of non-administrator physicians doesn't show these 2 buttons (the middle line).

proceed. Figure 4.10 provides an example of a form with these features

Across the application, the user interface uses graphical elements that scale according to screen dimensions, with a consistent color scheme and font style in all pages.

## 4.5 Implementation Details

Umedicine, as described in Chapter 3, was implemented as a collection of singleton beans running in a Spring container, in turn running inside a Tomcat container. The information exchanged between Tomcat and client browsers was not encrypted and there was no session isolation between clients making requests from the server at the same time. This means that any patient information kept in the server memory would be shared between connections and the information would be transmitted over the network as readable plain text, hence not guaranteeing confidentiality. To address these problems, we enabled the *TLS* cryptographic protocol [8], enforcing encryption of data transmitted between Tomcat and the clients, and configured the application to open a *session* for each client, isolating in-memory user data from other connected clients. To isolate client sessions, beans where user-specific information is stored are annotated with the *@Session* annotation. This Java annotation, provided by Spring, generates a bean of the annotated class for each browser-associated session, guaranteeing that when user-specific information is stored in bean fields, it cannot be read or written through requests from a different browser.

The access to specific functionalities and information in the application is tied to URL patterns and is controlled by Spring Security. For example, URLs with the pattern "/medico/*" can only be accessed by physicians. Pages for exclusive access by patient or clerk users have the pattern "/paciente/*" and

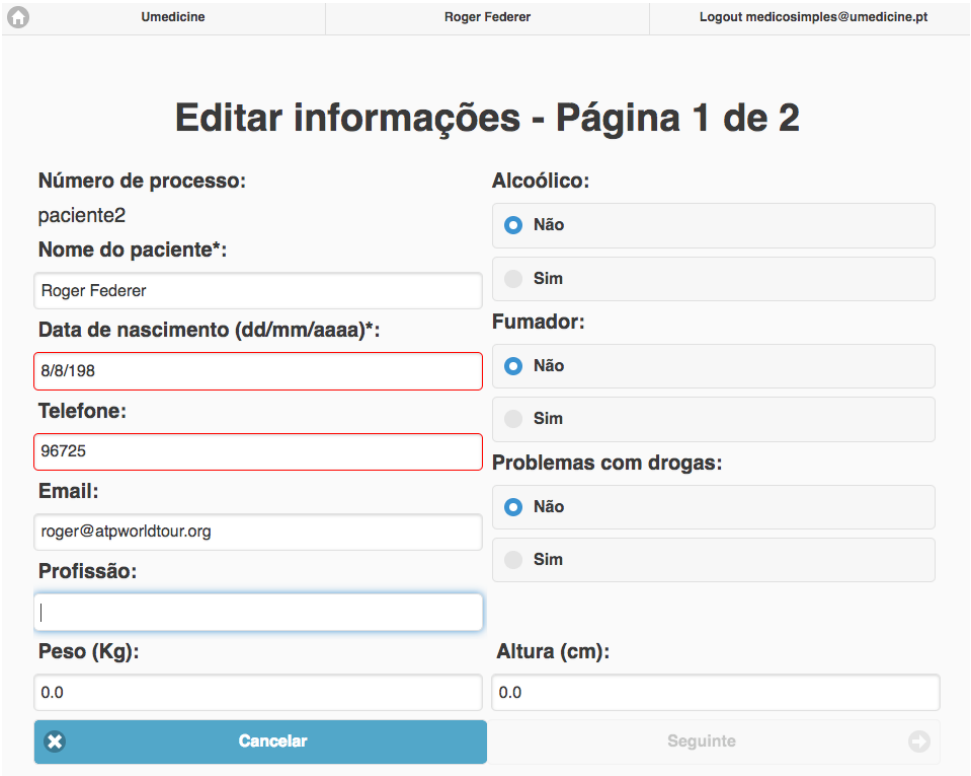



**Figure 4.10:** Patient information form with user-guidance techniques. The inserted date ('Data') does not respect the required format, the telephone number ('Telefone') does not meet the minimum length and the user has navigated away to the 'profession' field ('Profissão').

"/clerk/*", respectively. In the case of patients, the application enforces that they can only access or modify information bound to their process number, which is also their Umedicine username.

Password generation in Umedicine begins with the generation of a pseudorandom large integer (BigInteger), using the SecureRandom class from Spring Security (Spring Framework's subproject that provides authentication and authorization services). This integer is then converted to the corresponding string in a given base (by default, 32), which is stored in the database as the user's initial password. All passwords are hashed with the BCrypt class from Spring Security, which implements OpenBSD-style Blowfish password hashing using the scheme described in [33]. Once the generated password is successfully stored in the database, it is sent to the user using Spring Framework's MailSender interface.

Umedicine's interface is built with extensive use of JSP technology (introduced in Section 3.3). JSP fragment files and tags are used to abstract away and reuse interface elements, ensuring that the look of the interface, including element size, colorscheme and fonts, is the same across the different areas of the application.

Two other enhancements were introduced in the application. First, the implementation of transactional operations, in general, was simplified by using Spring's @Transactional annotation. One example is the saving of a patient's information to the database, which involves three INSERT Structured Query

Language (SQL) statements. By using the @Transactional annotation, the operation is rolled back if an error occurs in the database or in the method, without need for extra SQL code. Second, a logger was configured application-wide to log any occurring exceptions to a file. This file is be available for developers to look for information to solve problems reported by users. The logger makes use of the Simple Logging Facade for Java (SL4J) and the standard logging capabilities of Spring.

## 4.6 Conclusion

In this chapter we described the new version of the Umedicine application. It provides four different kinds of user types that reflects the needs of use by physicians (with or without administration privileges), patients and administrative staff. It has enhanced security, enforcing connection encryption, separate sessions for the clients and a password management module that allows users to change their passwords. Finally, we describe the changes made to the interface to accommodate this new functionalities and how we used Spring Framework and other technologies to implement the features described in the previous sections. In the next chapter, we describe Umedicine's new data analysis module.

# 5

# Umedicine: Clinical Decision Support Module

## Contents


5.1 A Clustering Approach for Clinical Time Series . . . . . 57

5.2 Visualization of Clinical Time-Series Data Clusters . . . . . 61

5.3 Discussion . . . . . 65

A major advantage of medical information systems is that they are able to store data from a large population of patients, hence providing a large source of data for discovering trends in disease progression that might be difficult to uncover by physicians in their daily clinical practice. In order to make such datasets useful, a medical information system needs to provide means to find groups of similar patients with basis on their personal and medical history and enable the exploration of the characteristics of the patients in these groups (Requirement group 6 in Section 3.1). This chapter proposes an approach to address this requirement. This approach makes use of an off-the-shelf state-of-the-art clustering algorithm for time series, to find groups of patients with similar variation of relevant clinical parameters. We complemented the clustering algorithm by designing and implementing a visualization mechanism to enable the exploration of the results for each group of patients discovered by the algorithm. This approach can be applied to any medical parameter that varies over time. We illustrate our approach with the International Prostate Symptom Score (IPSS), a parameter used by urologists to measure the severity of Benign Prostatic Hyperplasia (BPH) symptoms. In this chapter, we begin by describing the clustering algorithm in Section 5.1, and then we describe the visualization mechanism in Section 5.2. Finally, Section 5.3 discusses the main achievements presented in this chapter as well as the limitations of the developed work.

## 5.1 A Clustering Approach for Clinical Time Series

Clustering algorithms are widely used in biological research for a variety of applications [23, 32]. Notably, these algorithms are employed to find groups of genes with similar temporal expression patterns in Genomics experiments. Typically, these experiments involve measuring the expression of several thousand genes of an organism. Hence, Computational Biology research literature is fertile in studies where clustering algorithms were evaluated on data with the approximate size and dimensionality that we find in clinical research: the number of genes and time points in a Genomics experiment is comparable to the number of patients and time points for clinical parameters in clinical databases.

The use of classical clustering techniques such as $K$-means have important limitations when used with biological or medical data:

1. These algorithms tend to find clusters of similar size, hence they may not find interesting, relatively small clusters.

2. Each element (patient) is assigned to one and only one cluster, while it may display behavior similar to more than one (albeit at different intervals in time), or to none.

3. These algorithms compute clusters using all dimensions (which in time series data translates to using all time points) at once, hence they may not cluster together time series that are similar in all

but one or two time points (outliers).

Biclustering algorithms [6] are a family of algorithms that overcome these disadvantages in genomic datasets. Due to the similar size of genomic and medical datasets, we expect that biclustering's benefits apply to medical time series as well. Biclustering algorithms find groups of patients with basis on a subset of the time points instead of using all time points at once. In other words, they produce a local model instead of a global model [24]. This problem can be formulated as finding clusters of rows in a data matrix that are similar across a subset of the columns. In this matrix, each row represents a patient, each column represents a time point and each matrix element holds a value for a medical parameter. The complexity of an algorithm to find these clusters depends on the criteria of similarity between rows, but most formulations are NP-hard [29]. Hence, most biclustering algorithms resort to heuristics without guaranteeing optimal solutions, or have prohibitive running times. The problem becomes tractable, however, in cases where the measured medical parameter is discrete and the search for similar series is restricted to contiguous time points. Subsection 5.1.1 describes an algorithm that explores these restrictions to compute clusters in linear time, which we integrated with Umedicine to find clusters of patients with similar history.

### 5.1.1 CCC-Biclustering

CCC-Biclustering [24] is a biclustering algorithm that explores the idea that, when dealing with time series, analysis is concerned with a sequence of events in time and, in particular, with patterns observed for contiguous time points (or columns in the data matrix). This algorithm explores this restriction of contiguity between time points to find all maximal Contiguous Column Coeherent Biclusters (CCC-Biclusters) in time linear in the size of the data matrix.

The algorithm takes as input a matrix of patient data where each row represents a patient, each column represents a time point and matrix elements represent values for a discrete medical parameter. Many medical parameters of interest, however, are not discrete. As for the genomic data for which the algorithm was developed [24], the medical parameter to be used as basis for the clustering must be discretized using an appropriate strategy. In the original paper, authors are concerned with changes in gene expression, which are measured in a continuous positive scale with zero as fixed minimum. Hence, the authors normalize the scale by taking the ratio of the measured gene expression values with respect to a reference value and computing the logarithm of the ratio, to get a meaningful measure of change. They proceed to discretize this scale in three levels: significant increase, significant decrease and no change of gene expression. This strategy reflects their goal of finding groups of genes with similar variation patterns. For other problems, care must be taken to choose an appropriate discretization approach as well.

We illustrate the application of CCC-Biclustering to IPSS time series data. IPSS is a score calculated

from a standardized questionnaire given to certain Urology patients, such as patients who suffer from BPH. Physicians request that BPH patients fill in the questionnaire several times during the treatment, obtaining scores that reflect the patient's health status evolution in time. IPSS has a uniform scale of integer values between 0 (best-case scenario for the patient) and 35 (worst-case scenario for the patient). It is, thus, discrete in itself. However, imposing such a fine discrete scale of 36 values would rarely cluster patients together: only when they had a sequence of exactly equal scores would the algorithm consider them part of the same cluster. For this reason, it makes sense to use a coarser discretized scale: for example, a scale with 6 discretization levels would allow for a difference of up to 6 points in IPSS while still being able to capture increasing or decreasing trends, as variations of the IPSS value would often result in a change to a different level.

The library containing the CCC-Biclustering algorithm, excluding the preparation of the discretized matrix, was obtained from the algorithm's authors webpage[1]. As required by the algorithm, we prepare a matrix representing the discretized data: each row represents a patient, columns represent sequential time points and each matrix element holds a character representing the discretization level in which the original matrix element falls. For example, if we choose to discretize the IPSS scale into 6 levels, we need 6 symbols: 'A' for [0, 5], 'B' for [6, 11], 'C' for [12, 17], 'D' for [18, 23], 'E' for [24, 29] and 'F' for [30, 35]. A row corresponding to a patient with a sequence of IPSS values {34, 29, 21, 14, 10, 3} would be converted to {F, E, D, C, B, A} in the discretized, symbolic matrix.

CCC-Biclustering builds a suffix tree from the symbolic matrix. Figure 5.1, left panel, shows an example of such a tree, built from the matrix shown in the right panel. In the suffix tree building process, the column number is appended to each symbol. As shown in the example, every internal node in the tree corresponds to a one row-maximal[2], right-maximal CCC-Bicluster in the matrix[3]. Internal nodes in the suffix tree correspond to CCC-Biclusters, which are maximal if and only if there is no incoming link (dashed arrows in the figure) from another node with the same number of leaves in its subtree.

After executing the clustering algorithm, the CCC-Biclustering library computes statistics from each cluster. One of the most important of these statistics is a *p*-value that reflects how similar the patients in the same group are among themselves. This *p*-value is calculated with basis on a hypothesis test in which the null hypothesis assumes that a cluster, with its size and patient data, was randomly generated. The lower the value of the *p*-value, the smaller is the probability of finding this group of patients under this null hypothesis. Hence, patients in groups with lower *p*-values can be expected to be more similar.

[1] http://kdbio.inesc-id.pt/software/ccc-biclustering

[2] A CCC-Bicluster is row-maximal if we cannot add more rows to its set of rows and mantain the same subset of equal contiguous column values.

[3] A CCC-Bicluster is right-maximal if we cannot extend it to the right by adding a contiguous symbol at its end without removing a row from its set of rows.

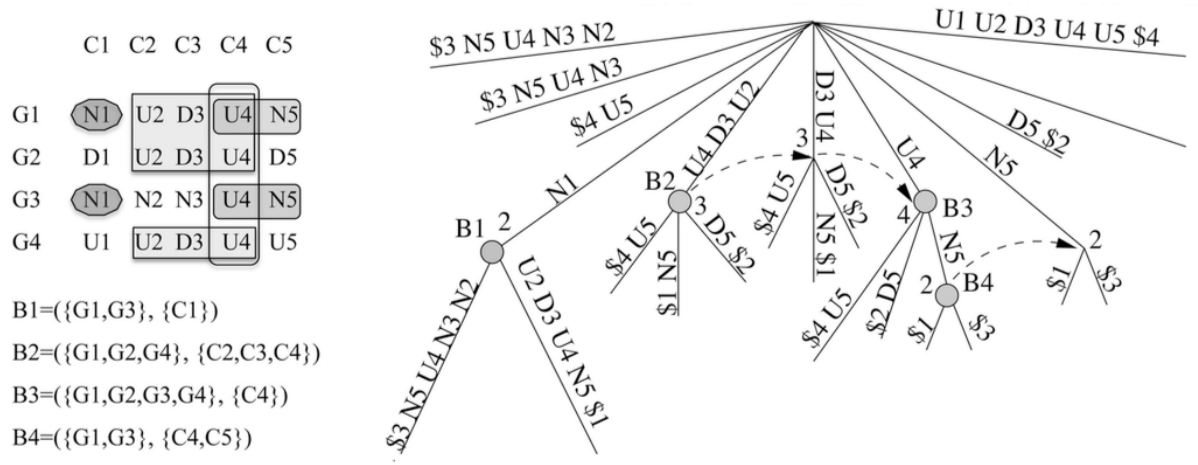



**Figure 5.1:** (Left) Example of a discretized matrix, with maximal CCC-Biclusters with at least two rows (B1 to B4) identified by circles. The strings N, UDU, U, and UN correspond to the patterns of the maximal CCC-Biclusters B1, B2, B3 and B4, respectively. The dollar sign is just a string termination marker. (Right) Suffix tree for the symbolic matrix shown on the left. For clarity, the figure does not show the leaves that represent string terminators that are direct children of the root. Figure from [24].

### 5.1.2 Implementation Details

The library containing the CCC-Biclustering algorithm exposes an API that we use to integrate it with the Umedicine application. The code that uses the library is in a dedicated class named *TimeClustering*, designed to be reusable with any medical time-series data stored by the application. The *TimeClustering* class constructor receives two arguments: the path to a comma-separated values (CSV) text file containing the time-series data to be analyzed, and the number of discretization levels to be used in the execution of the algorithm. (Writing a DAO that reads the data from the database remains, at this time, as future work.) The constructor reads the data from the CSV file into a Java array. The *TimeClustering* class instance also stores the identifiers for the lines and columns of the original data matrix.

The *TimeClustering* class is invoked from the *DataAnalysisService* class in Umedicine's server, which uses methods from the CCC-Biclustering library to map the data array to a symbolic matrix and to compute the clusters and related statistics. These tasks are wrapped in a method of the *DataAnalysisService* class that accepts the input arguments of the clustering algorithm.

The clustering functionality as a whole is exposed to clients through a controller, the DataAnalysisController class, which accepts an Hypertext Transfer Protocol (HTTP) GET request containing the arguments needed to invoke DataAnalysisService methods. The controller returns an HTTP response that contains the information needed by the client to show the results to the user as described in Section 5.2.

## 5.2 Visualization of Clinical Time-Series Data Clusters

We developed an interactive visualization mechanism and incorporated it into the Umedicine application to enable the exploration of time-series data as analyzed by the biclustering algorithm described in Section 5.1. Physicians can navigate to the page with this visualization functionality from their homepage (illustrated in Figure 4.9).

The visualization is a matrix of line charts (represented in Figure 5.2), where each chart corresponds to a group of patients computed with the CCC-Biclustering algorithm, described in Section 5.1. Each chart also indicates the number of patients in the corresponding group.

**Figure 5.2:** Visualization of time series clustering in Umedicine applied to the IPSS, a diagnosis parameter for BPH.

Users can set six different parameters of the CCC-Biclustering algorithm and visualization mechanism. The top 4 parameters correspond to parameters of the CCC-Biclustering algorithm (explained in Subsection 5.1.1) but were renamed for a more intuitive use by users with little knowledge of statistics. The bottom 2 parameters are not related to the algorithm but to the visualization itself. In the same order as in Figure 5.2, they are:

1. **Allowed similarity between groups** (*Semelhança permitida entre grupos*): this parameter corresponds to the maximum overlap between clusters of the CCC-Biclustering algorithm. In other words, it is defined as the maximum percentage of patients in a group that can also be part of another group. The user can choose 5 levels of similarity: very small, small, intermediate, large and very large, corresponding to 1%, 5%, 10%, 25% and 50% maximum overlap.

2. **Number of IPSS levels** (*Número de níveis do IPSS*): corresponds to the number of intervals/symbols chosen to discretize the IPSS scale. The larger the number of IPSS levels, the narrower they are. Hence, a larger number of levels leads to clusters where patients are more similar to each other. The user has 5 options: very small, small, intermediate, large and very large, corresponding to 3, 6, 9, 12 and 18 levels of discretization.

3. **Minimum group size** (*Tamanho mínimo dos grupos*): the minimum number of patients in each cluster/chart to be displayed. Charts with a number of patients inferior to the number defined in this option are not shown. The user has 5 options: very small, small, intermediate, large and very large. The 'very small' option sets the minimum number of patients per cluster/chart to 2. The other options are computed as a function of the total number of patients in the dataset: 'small' corresponds to 10% of the total number of patients, 'intermediate' corresponds to 20%, 'large' to 40% and 'very large' to 60%.

4. **Group homogeneity** (*Homogeneidade dos grupos*): the larger this parameter, the smaller the maximum *p*-value of the clusters — as explained in Section 5.1.1; clusters whose *p*-values exceed this value are not shown. The user chooses among the options 'very small', 'small', 'intermediate', 'large' and 'very large', which correspond to *p*-values 0.25, 0.125, 0.083, 0.0625 and 0.05, respectively.

5. **Maximum number of clusters**/charts to show (*Número máximo de grupos a visualizar*)

6. **Presentation order** (*Ordem de apresentação*): users can choose between displaying charts ordered from the one with most patients to the one with less patients, from the one with best overall change in IPSS to the one with worst overall change of IPSS or from the one with worst overall change in IPSS to the one with best overall change of IPSS. The overall change in IPSS here is the simple difference between IPSS at the final time point and the IPSS at the initial time point.

After setting these parameters as desired, the user generates the clusters and the corresponding visualization by clicking/tapping the 'Calculate' (*Calcular*) button.

Each chart can be selected with a mouse click or finger tap. This action overlays selected information about the corresponding group of patients, as illustrated in Figure 5.3. The information shown includes the three most prevalent characteristics in the group (*Características mais comuns*) and the treatment

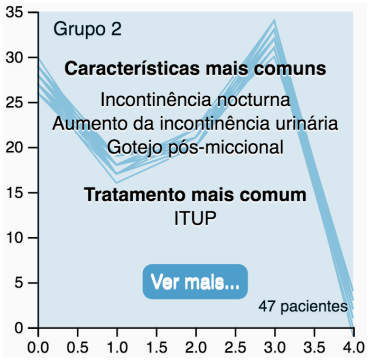



**Figure 5.3:** Example of information displayed over a plot (from figure 5.2) representing a cluster of patients with similar IPSS evolution.

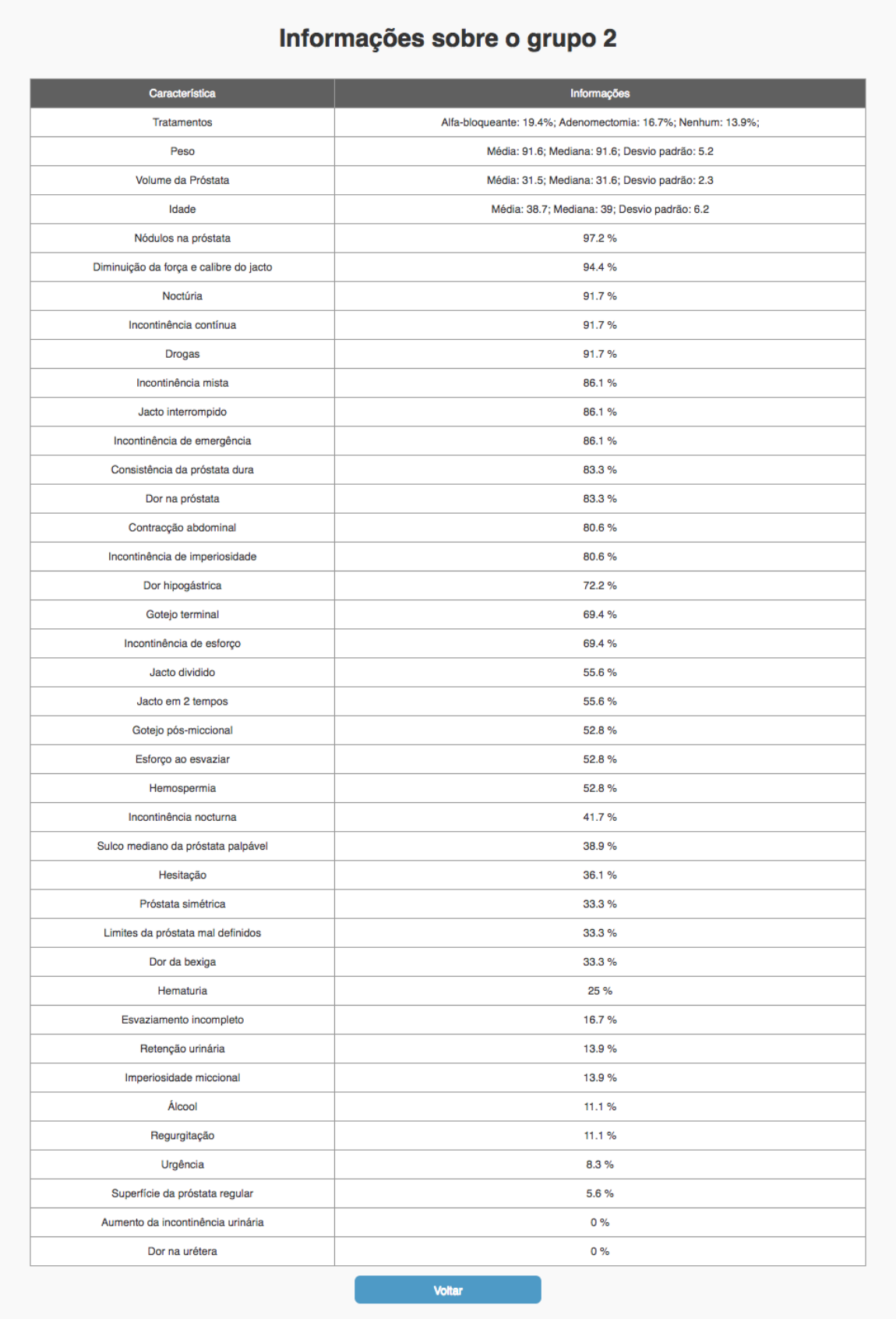
given to most patients (*Tratamento mais comum*). The button at the bottom (*Ver mais...*) hides the charts and shows a table with more information about the selected group of patients, as illustrated in Figure 5.4 for group 2. The first line shows the percentages of patients that were subject to the most common treatments for this group. The second, third and fourth lines show statistical information about weight, prostate volume and age of these patients. The lines that follow show the percentage of patients that display certain symptoms of characteristics, sorted from highest to lowest – the top three of these symptoms and characteristics are the same shown in Figure 5.3. At the bottom of the table, a button (*Voltar*) allows users to close the table and return to the view with the charts (as in Figure 5.2).

### 5.2.1 Implementation

When the user generates charts by pressing the *Calculate* button (*Calcular* in Figure 5.2), the front end gathers the options chosen by the user and sends an Ajax request to the back-end server, where the user options correspond to arguments of a method of the DataAnalysisController class, as described in Subsection 5.1.2. The controller returns clusters, *p*-values and patient data back to the browser.

The charts and tables were developed with the D3[4] [4] library, version 4. D3 stands for *Data-Driven Documents*. It is a data visualization language developed with three priorities in mind: *compatibility* with other JavaScript libraries, ease of *debugging* and *performance*. With D3, developers can *select* specific Document Object Model (DOM) elements, apply *operators* and bind data to the elements, producing data-dependent visual representations in Scalable Vector Graphics (SVG). *Transition* effects may be applied in place of operators to produce smooth interpolation between element attributes and styles over time, instead of obtaining an instantaneous effect. D3 comprises event handlers to respond to user input and trigger operators or transitions, enabling interaction. The library provides various geometric *shapes*; it provides *scales* to support ordinal and quantitative (linear, logarithmic, exponential, quantile) values, as well as color gradients; it provides *layouts*, which are mathematical structures that facilitate

[4] http://d3js.org



## Informações sobre o grupo 2

| Característica | Informações |
|---|---|
| Tratamentos | Alfa-bloqueante: 19.4%; Adenomectomia: 16.7%; Nenhum: 13.9%; |
| Peso | Média: 91.6; Mediana: 91.6; Desvio padrão: 5.2 |
| Volume da Próstata | Média: 31.5; Mediana: 31.6; Desvio padrão: 2.3 |
| Idade | Média: 38.7; Mediana: 39; Desvio padrão: 6.2 |
| Nódulos na próstata | 97.2 % |
| Diminuição da força e calibre do jacto | 94.4 % |
| Noctúria | 91.7 % |
| Incontinência contínua | 91.7 % |
| Drogas | 91.7 % |
| Incontinência mista | 86.1 % |
| Jacto interrompido | 86.1 % |
| Incontinência de emergência | 86.1 % |
| Consistência da próstata dura | 83.3 % |
| Dor na próstata | 83.3 % |
| Contracção abdominal | 80.6 % |
| Incontinência de imperiosidade | 80.6 % |
| Dor hipogástrica | 72.2 % |
| Gotejo terminal | 69.4 % |
| Incontinência de esforço | 69.4 % |
| Jacto dividido | 55.6 % |
| Jacto em 2 tempos | 55.6 % |
| Gotejo pós-miccional | 52.8 % |
| Esforço ao esvaziar | 52.8 % |
| Hemospermia | 52.8 % |
| Incontinência nocturna | 41.7 % |
| Sulco mediano da próstata palpável | 38.9 % |
| Hesitação | 36.1 % |
| Próstata simétrica | 33.3 % |
| Limites da próstata mal definidos | 33.3 % |
| Dor da bexiga | 33.3 % |
| Hematuria | 25 % |
| Esvaziamento incompleto | 16.7 % |
| Retenção urinária | 13.9 % |
| Imperiosidade miccional | 13.9 % |
| Álcool | 11.1 % |
| Regurgitação | 11.1 % |
| Urgência | 8.3 % |
| Superfície da próstata regular | 5.6 % |
| Aumento da incontinência urinária | 0 % |
| Dor na urétera | 0 % |

Voltar

**Figure 5.4:** Example of information displayed when a user clicks the button 'Ver mais' (meaning 'get more information') on plot such as the one in Figure 5.3.

the development of certain visualization techniques; and *behaviors*, which abstract event handling (such as zoom and panning) to facilitate the implementation of interaction techniques.

## 5.3 Discussion

The techniques presented in this chapter provide efficient means to find and visualize patient time series data, as illustrated for the BPH and IPSS questionnaire example. The CCC-Biclustering algorithm proposed in [24] achieves linear running time due to the discretization of the time series data. This is also a disadvantage, since the hard split of the scale of the medical parameter into discrete levels may place close values into different levels. (In the six-level example given in Figure 5.2, if we have a patient with IPSS = 5 and another patient with IPSS = 6 at the same time, they will fall in different levels.) In spite of this problem, for a number of patients of at least a few thousands, the algorithm clusters together enough patients to yield interesting results, as it did for the original intended application (gene expression analysis). Our visualization mechanism with linecharts shows the groups of patients with similar evolution over time. The user can explore the data further through interaction with the linecharts and obtain a summary of characteristics and statistics of patient groups. The validation of the methods described in this chapter is described in Chapter 6.

# 6

# Validation

## Contents

6.1 Clustering and Visualization Performance Evaluation . . . . . 69
6.2 Usability Tests . . . . . 71

In this chapter we validate the approach presented in Chapters 4 and 5 to analyze medical time series data, with respect to execution performance and usability. In Section 6.1 we describe how we investigated the performance of the algorithm, integrated in the server, and the performance of the visualization, in the client. In Section 6.2, we describe the usability tests that we conducted for the visualization.

## 6.1 Clustering and Visualization Performance Evaluation

We measured the performance of the clustering algorithm applied to the medical data and of our approach to present the results to the user. In our experiments we varied two input parameters: number of patients and number of time points considered for each patient, and measured the time to analyze the data and to display the results of the analysis. To evaluate the performance of the data analysis module, we measured the time between receiving the request from the client up to the response from the controller. Hence, it includes reading the data, computing the clusters and related statistics, and organizing the data into the objects that are serialized to be sent to the client. Concerning the performance of displaying the results, we measured the time between receiving the cluster data from the server and displaying the charts on the screen.

### 6.1.1 Experimental Setup

We generated a synthetic dataset to support experimentation with the biclustering algorithm and the corresponding visualization. This synthetic data represents a collection of patients that answered the IPSS questionnaire at several points in time after the beginning of a treatment. We generated a set of time series (each corresponding to a virtual patient) as follows:

1. Generate IPSS values for $t$ time points for $n$ patients. Each value was sampled from a uniform distribution function between 0 and 35 (the limits for IPSS).

2. For each of the $n$ patients, generate $m$ time series with $t$ time points, by sampling $m$ times a normal distribution centered on each time point's value and with standard deviation 1.

The procedure for generation of IPSS time series outlined above generates $n$ sets of $m$ time series, where series in the same sets are drawn from the same normal distribution. Hence, a clustering algorithm is expected to output clusters of time series drawn from the same set.

In addition to the IPSS time series, we generated synthetic data for a variety of parameters relevant to the BPH pathology – for the full list, see Figure 5.4. For each group of time series generated as described above, the following procedure was followed to generate BPH data:

1. For each time series (patient) pick a random treatment associated with BPH.

2. For the parameters 'weight' (*peso*), 'prostate volume' (*Volume da Próstata*) and 'age' (*Idade*):

   (a) For each of the *n* groups of time series, generate a random value within a meaningful range from an uniform distribution.

The input size of the CCC-Biclustering algorithm was varied in two experiments: first, we varied the number of patients, through the value of *m*, for a fixed number of time points, and second, we varied the number of time points, *t*, for a fixed number of patients. When varying *m*, *n* was set to 10 (yielding a total of 10*m* patients) and *t* was set to 5. When varying *t*, *m* was set to 100 and *n* was set to 10, corresponding to a total of 1000 patients (or matrix rows). These parameters were chosen to generate a dataset with realistic size. Synthetic datasets were generated using the NumPy[1] library for numerical computing, for the Python[2] language. The experiment was done on a 2009 MacBook Pro laptop computer with 8 gigabytes of 1066MHz DDR3 memory, a 2.66 GHz Intel Core 2 Duo processor and a SATA hard disk drive.

### 6.1.2 Results and Discussion

The results obtained for both experiments are shown in Figure 6.1. The left panel shows the behavior of our solution when we vary the number of patients and the right panel shows the behavior of our solution when we vary the number of time points per patient.

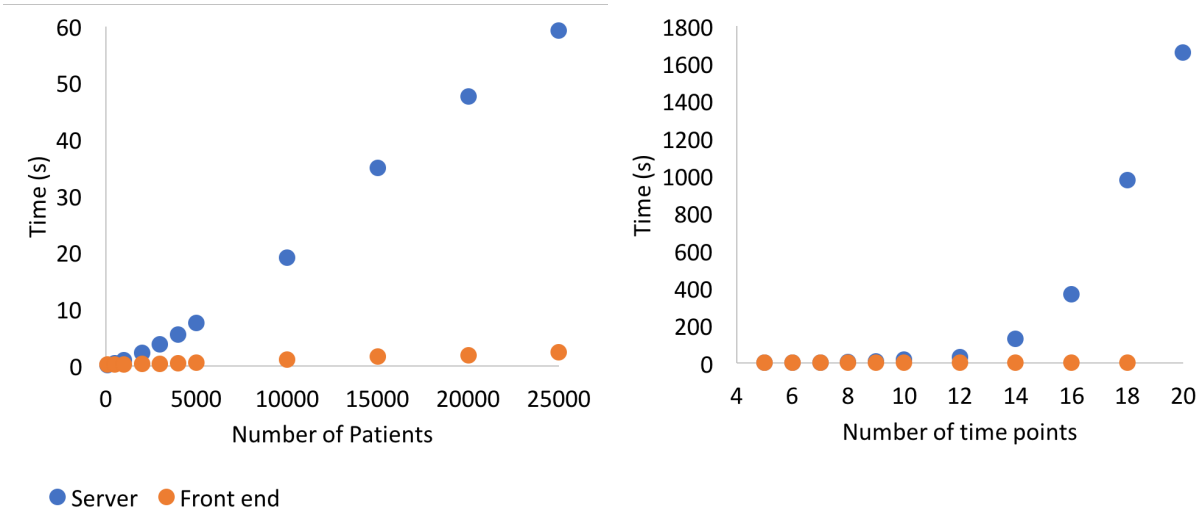



**Figure 6.1:** Performance measurements for the CCC-Biclustering algorithm and the cluster visualization mechanism. Left panel: results for different numbers of patients for a fixed number of 5 time points per patient. Right panel: results for different numbers of time points per patient, for a fixed number of 1000 patients.

[1] http://www.numpy.org
[2] http://www.python.org

The CCC-Biclustering algorithm has a worst-case complexity linear in the size of all input parameters, thus we expect the processing time to increase linearly both with the number of patients (input matrix rows) and with the number of time points per patient (input matrix columns). Our results comply with this expectation for the number of patients (Figure 6.1, left panel), though with a slight deviation from linearity for very small numbers of patients. On the other hand, the performance degrades more than expected with the increase of the number of time points. Since the complexity of the algorithm was mathematically proven and the server only adds read/write and data transfer operation to the operations performed by the CCC-Biclustering library, we speculate that the observed deviation from linearity is caused by data pre- and/or postprocessing steps, such as the computation of *p*-values and other statistics. However, further investigation of the implementation would be necessary to provide an accurate explanation for this result.

Despite the result for large number of time points, we can conclude that the server delivers appropriate performance in most realistic scenarios for IPSS-based studies. It is unlikely that the patient will answer the corresponding questionnaire more than 10 times and our results only show a severe degradation of performance (with waiting times higher than 3 minutes) for cases with more than 14 time points per patient. The performance measured for increasing number of patients is also encouraging: a matrix with 25000 patients and the realistic column size of 5 time points could be analyzed in 1 minute; assuming a linear trend, this means that data from hundreds of thousands of patients could be analyzed in a few minutes. In a pessimistic scenario for the Portuguese health care system, the application would have to accommodate data from hundreds of thousands of patients, which would lead to a waiting time of several minutes. For larger countries, with a high number of patients, this processing time would not be acceptable from the point of view of the user. In such cases, a better approach for on-the-fly medical data analysis would be to do it beforehand, or periodically, and store the result in the database. This way, the results would always be available, without any delay, when requested by the user.

The front end, in turn, added a comparatively small waiting time, typically under half a second, which does not represent a usability concern. This shows that our graphical solution to present medical data analysis results has an excellent performance.

## 6.2 Usability Tests

We conducted usability tests for the visualization that we developed for inspection of clustered time series data (described in Section 5.2). With these tests, we aimed to determine whether the provided features enabled easy data exploration by the users, as well as identify the main difficulties faced by users and common interaction errors while using the application. For this purpose, we assigned a group of volunteers a set of tasks where, after a brief explanation of the usage of the system, they used the

different functionalities provided by our visualization mechanism. We recorded the number of clicks needed to complete each task and took note of the interaction errors of the users.

### 6.2.1 Experimental Setup

10 volunteers participated in the tests, between the ages of 24 and 34, half of which were male and half of which were female. All volunteers have university degrees and work in fields not related to healthcare. The test was composed of a set of 6 tasks and a questionnaire concerning the usage of our visualization mechanism to perform the 6 tasks. These tasks were the following:

1. Generate charts with the groups of patients sorted from the one with most patients to the one with less patients, keeping the default values for the other options.
2. Indicate the groups of patients whose health condition improved.
3. Consider the $4^{th}$ group of patients:
   (a) Indicate the most common characteristics of this group of patients;
   (b) Indicate the average weight of this group of patients;
   (c) Indicate the percentage of patients with *divided urinary stream*.
4. Change the options so that only the first 3 groups are shown.
5. Change the *Minimum group size* to *Large*. What happens? Why?
6. Change the *Allowed similarity between groups* to *Large*. What difference do you observe for the first two groups?

Once each user had finished the tasks, they were asked to answer a questionnaire with 7 questions. In this questionnaire, users rated the application with respect to several usability aspects. Finally, we gave users 3 open-answer questions inquiring about the greatest difficulties, best features and suggested modifications to the visualization. The document given to users with the questions and tasks is included in Appendix A (in Portuguese). The tests were done on a 2009 MacBook Pro laptop computer with 8 gigabytes of 1066MHz DDR3 memory, a 2.66 GHz Intel Core 2 Duo processor and a SATA hard disk drive, with a mouse pointing device.

### 6.2.2 Results and Discussion

Table 6.1 summarizes information about the volunteers and how they rated the visualization (from 1 to 5, 5 being best) on several topics. The total number of clicks was also counted while users were doing

**Table 6.1:** Usability test results

| User | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Average | Standard Error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sex | F | M | F | M | F | M | M | M | F | F | NA | NA |
| Age | 26 | 24 | 33 | 33 | 33 | 26 | 33 | 33 | 34 | 31 | 30.6 | 3.7 |
| YUC | 15 | 16 | 16 | 18 | 20 | 27 | 22 | 20 | 24 | 26 | 20.4 | 4.3 |
| Q1 | 5 | 4 | 4 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 4.5 | 0.5 |
| Q2 | 4 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3.8 | 0.4 |
| Q3 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 3 | 5 | 4.7 | 0.7 |
| Q4 | 4 | 5 | 4 | 5 | 5 | 4 | 4 | 5 | 4 | 5 | 4.5 | 0.5 |
| Q5 | 5 | 5 | 4 | 4 | 5 | 5 | 4 | 5 | 4 | 5 | 4.6 | 0.5 |
| Q6 | 4 | 5 | 4 | 4 | 5 | 3 | 3 | 4 | 4 | 5 | 4.1 | 0.7 |
| Q7 | 5 | 5 | 5 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 4.7 | 0.5 |

YUC: How long the volunteer has been using computers, in years
Q1: The color scheme is appropriate to visualiza the data
Q2: The options given to control patient group generation are easy to understand
Q3: The user can understand when a patient improves or gets worse
Q4: The user can understand that patients in the same group have similar IPSS along time
Q5: The user gets a good understanding of the characteristics of patients in the same group
Q6: It is easy to understand why patients are grouped together
Q7: The data displayed in the visualization table complement well the information shown in the corresponding chart

the test, but it is obvious that a systematic error occurred during the counting process, hence this data was discarded for our analysis.

Overall, the results shown in table 6.1 indicate that the users were pleased with their experience regarding the issues in questions Q1 to Q7: in average, they rated their experience above 4 in all cases except Q2, which concerns the dropdown lists where they can choose values for the parameters of the visualization (described in Section 5.2, shown in Figure 5.2). Indeed, the lack of clarity of the meaning of these parameters was a recurrent complaint during the test: all mentioned, one way or another, that the names are not intuitive and requested further explanation of their meaning. However, 8 of the 10 users recognized that once the meaning of the options was understood, they could use them and interpret how they impact the observed results. Apart from one exception, the users found the options concerned with the presentation (*Maximum number of clusters* and *Presentation order*) easier to understand than the options related to the input of the algorithm. Confusion about the meaning of options became apparent as one user used a wrong option twice for the intended effect (the user *Minimum group size* instead of *Presentation order*) and two other users did the same mistake once. These were the only errors of the users during the test, otherwise they always used the application in the intended way. These errors reflect the need to reconsider how the options are presented to the users.

In general, users were able to complete the tasks successfully. When asked to select groups of patients who had improved their conditions, there was a good agreement between the users (Table 6.2). All interpreted 'improvement' as an overall decrease of IPSS from the first to the last time points. Apart

from group 3, about which there was a split of opinions between on third and two thirds of the users, almost all users had the same opinion about the improvement of patients in the other groups, which was correct: the groups where patients improved were groups 2, 4, 5, 7, 8 and 9.

**Table 6.2:** Usability test results: patient improvement according to the users

| Patient group | Users who said the conditions of the patients improved |
|---|---|
| 1 | 0 % |
| 2 | 90 % |
| 3 | 30 % |
| 4 | 90 % |
| 5 | 100 % |
| 6 | 10 % |
| 7 | 90 % |
| 8 | 100 % |
| 9 | 80 % |

When asked to indicate characteristics for the 4$^{th}$ group, users also selected the expected information – all users read the information overlaid on the chart and the table correctly, although one them read different information due to having selected different options for the visualization. Users were then asked to change *Minimum group size* from *Very small* to *Large* and interpret the result. The effect of the change was that the application couldn't produce any clusters meeting the requirement and the application showed a message conveying this information. 7 users made the correct interpretation, that there were not any groups large enough to be shown, but 3 users struggled to understand the result and asked for help. Finally, users were asked to change the *Similarity between groups* from *Medium* to *Large* and compare the results. All users complained that it was hard to compare the results because they had to switch back and forth between the two values (large and medium), but all concluded correctly that there was no difference between the results.

In the last part of the test, users were asked to describe what, in their opinions, were their main difficulties, the most positive aspects of the visualization and what they would like to have changed. 7 users complained explicitly about the unintuitive options provided. One of such users suggested to add a 'help feature' with text explaining the meaning of the options. One user also disliked not being able to distinguish each line more easily in the charts. The same user suggested to use a different color for each line and use a color gradient to provide information on one more parameter. Another user answered that the main difficulty was not to use the application but to understand the concepts of the medical field. Yet another user complained that they expected the charts to update automatically after changing option values, without the need to press the *Calculate* button. Three users suggested to have the button that closes the table staying visible while the user scrolls up or down to inspect the table; it was hard to find and there was the temptation to use the button to leave the page instead. 2 users also suggested to

use actual numbers in the *Minimum group size* option. 2 users would like to have larger font sizes in the charts.

Users tended to compliment the look and organization of the visualization page: 5 users explicitly praised the look of the charts and options, with two of them mentioning the 'clean' look and two others the spatial organization of the elements as very positive aspects. 5 users identified the most common characteristics of the groups overlaid on the charts and the table with group statistics as the most positive aspects of the visualization. 3 users answered that they liked the interaction experience, especially its responsiveness. 2 users said it was very easy to understand when patients improve or worsen. Other positive comments mentioned the color scheme, element size and how data was easy to interpret.

# 7

# Conclusion and Future Work

## Contents

7.1 Summary . . . . . 79
7.2 Limitations and Future Work . . . . . 80

In this work, we developed a software system to support clinical activity, building on Umedicine, an application with an appealing and easy-to-use graphical interface that physicians and patients can use to keep medical information up to date. As a result, Umedicine is a platform that provides easy, quick and always-on access to a large amount of clinical data. The medical information is persistently kept in an electronic and structured format, thus enabling the visualization of medical data per patient as well as the application of data analysis techniques to extract interesting knowledge from the collected medical data.

We took advantage of Umedicine's existing architecture to pursue two main objectives. On one hand, we upgraded Umedicine with security and usability features that enable its use by physicians, patients and administrative personnel, providing each with an appropriate interface and ensuring that each has access to, and only to, the respective functionalities. On the other hand, we built a CDS module for the system with a clustering algorithm suited to discover groups of patients with similar medical histories and a visualization mechanism to explore the clustering results, showing the main characteristics, treatments and statistics of the patient clusters.

## 7.1 Summary

In Chapter 2, we described approaches from the research literature in the field of CDS systems. We reviewed the main concepts about data warehouses and described examples of their use for clinical practice and research. In particular, we studied the IH enterprise data warehouse, which stores billions of clinical records and business data, and was successfully used to prevent the spreading of hospital infections and identify patients with high risk of VTE. We also described a data warehouse designed to support research on obesity that enabled the discovery of correlations between obesity and a variety of variables. We also reviewed the most common approaches in clinical data mining, namely the logistic regression, Bayesian network, artificial neural networks and rule-based decision, giving examples from the research literature of applications of these approaches to clinical problems. Finally, we described approaches in medical information visualization. We reviewed three systems that focus on the visualization of medical time series (patient histories) and one designed to compare patients and find groups of similar patients, with basis on a variety of parameters.

In Chapter 3, we introduced the Umedicine system, its requirements — laid out under the supervision of a Urology practitioner — and reviewed which requirements had been met and which required attention. Umedicine provided simple storage and visualization of individual patients, but fell short regarding the security requirements. It also could not be used by administrative staff and didn't provide data analysis capabilities. We then describe Umedicine's client-server architecture and components (front end, back-end server and database) as they were before the present work.

We began explaining the contributions from the present work in Chapter 4. We addressed the need for the clerk user type, for administrative personnel, as well as separate non-administrator and administrator physician user types, in addition to the existing patient user type. We redesigned the user interface to accommodate the functionalities specific for each user and provided the application with authentication and session management mechanisms. We designed and implemented a password generation and reset mechanism available to all users. Finally, we added a logging module to record running-time information about the system to aid developers in fixing problems.

In Chapter 5 we describe Umedicine's new CDS module. The CDS capabilities comprise a clustering algorithm suited for time-series data, which we apply to patient history data, and a visualization mechanism to enable intuitive inspection of the results by the users. To perform clustering of patient history data, we use CCC-Biclustering, an algorithm from the research literature that explores the contiguity of data points in time-series to find clusters with basis on a subset (not necessarily all) of time points. This strategy overcomes several disadvantages of classical clustering algorithms (like $K$-means) while achieving linear running time in the size of the input. To visualize the clustering results, we developed a visualization where sers can experiment with the parameters of the algorithm and visualize the obtained groups of patients in a linecharts. We further empower users by providing the choice to overlay treatment and common group characteristics information on the charts, and access to a table with comprehensive statistical information about each group.

We conducted experiments to evaluate the performance of the clustering algorithm and usability tests to collect feedback from users regarding the developed visualization mechanism. These experiments are described in Chapter 6. We made performance measurements both for the clustering algorithm running in the server and the visualization mechanism running in the client. We concluded that the algorithm's performance was adequate for datasets of realistic size and that the performance of the visualization was excellent. The results of the usability tests with the visualization were promising, as users were able to complete with success all the tasks and were mostly satisfied with the user interface provided.

## 7.2 Limitations and Future Work

Regarding the work presented in this thesis, we collected direct feedback from users during the usability experiments done for the visualization mechanism proposed to explore patient time-course data. The options provided are clearly not presented in a satisfactory, understandable way, hence these must be revised in the near future. Likely, this visualization has potential to include new features, such as a color gradient that adds information about an extra numeric parameter, as suggested by one of the user that participated in the tests. Yet another suggestion, made by several users during the test, was to differentiate the label of the button to leave the visualization's page from the button to leave the statistics

table view, and have the latter staying visible in the page, so that the users could always find it and return to the charts.

Still regarding visualization, there are certainly solutions proposed in the systems reviewed in the related work chapter that could be adapted to enrich our CDS module, such as the use of symbols to mark events in patients' timelines, or brushing to select and view the information relative to a subgroup of patients in a cluster.

There are many interesting features that can be incorporated to support medical activities. These can go beyond data analysis and use ML algorithms to suggest diagnosis and treatments to physicians, or perform predictions about the future evolution of patients' conditions. For example, Bayes networks can be used to estimate the probability that a patient develops a certain condition, or improves their condition if prescribed a certain treatment, with basis on data in the system, from patients with similar histories. Logistic regression approaches or the more sophisticated ANNs can be used to make predictions as well, in the context of diagnostics or treatment decisions, again with basis on data from the patient population.

Presently, the clustering algorithm is being applied to data read from an external text file, hence the integration of the CDS module with the server's DAO layer is a priority. Also, as the dataset size increases in the application, it becomes more useful to consider the integration of a data warehouse, to make querying for data analysis, especially data aggregation operations, more efficient. A data warehouse could also be an interesting solution to store pre-computed algorithm results, which would be ready to be analyzed by physicians with nearly-zero waiting time. This could be a suitable approach for the existing biclustering approach: instead of a waiting time of a few minutes, clustering results would be readily retrieved from the data warehouse and presented to the user.

It is also important to note that Umedicine is expected to be tested in a real clinical environment in the near future, with several physicians and their patients using it and providing feedback from their real experience. This experience will certainly guide future decisions regarding new functionalities to be integrated in the system and eventually determine the development lines that will be followed.

# Bibliography


[1] *Health IT and Patient Safety: Building Safer Systems for Better Care*. National Academies Press, Washington, D.C., March 2012.

[2] Peter C. Austin and Ewout W. Steyerberg. Interpreting the concordance statistic of a logistic regression model: Relation to the variance and odds ratio of a continuous explanatory variable. *BMC Medical Research Methodology*, 12:82, 2012.

[3] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive Probabilistic Networks with Hidden Variables. *Machine Learning*, 29(2-3):213–244, November 1997.

[4] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[5] Zhengping Che, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. Deep Computational Phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 507–516, New York, NY, USA, 2015. ACM.

[6] Yizong Cheng and George M. Church. Biclustering of Expression Data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press, 2000.

[7] Tom Christensen and Anders Grimsmo. Instant availability of patient records, but diminished availability of patient information: A multi-method study of GP's use of electronic patient records. *BMC Medical Informatics and Decision Making*, 8:12, 2008.

[8] Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. https://tools.ietf.org/html/rfc5246, August 2008.

[9] Jacques Donzé, Drahomir Aujesky, Deborah Williams, and Jeffrey L. Schnipper. Potentially Avoidable 30-Day Hospital Readmissions in Medical Patients: Derivation and Validation of a Prediction Model. *JAMA Internal Medicine*, 173(8):632, April 2013.

[10] R. Scott Evans, James F. Lloyd, and Lee A. Pierce. Clinical Use of an Enterprise Data Warehouse. *AMIA Annual Symposium Proceedings*, 2012:189–198, November 2012.

[11] Martin Fowler. Inversion of Control Containers and the Dependency Injection pattern. https://martinfowler.com/articles/injection.html, January 2004.

[12] Jiawei Han, Micheline Kamber, and Jian Pei. 8 - Classification: Basic Concepts. In *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 327–391. Morgan Kaufmann, Boston, 2012.

[13] Jiawei Han, Micheline Kamber, and Jian Pei. 9 - Classification: Advanced Methods. In *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 393–442. Morgan Kaufmann, Boston, 2012.

[14] J A Hanley and B J McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, April 1982.

[15] J A Hanley and B J McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, April 1982.

[16] P. J. Haug, R. M. Gardner, K. E. Tate, R. S. Evans, T. D. East, G. Kuperman, T. A. Pryor, S. M. Huff, and H. R. Warner. Decision support in medicine: Examples from the HELP system. *Computers and Biomedical Research, an International Journal*, 27(5):396–418, October 1994.

[17] Klaus Hinum, Silvia Miksch, Wolfgang Aigner, Susanne Ohmann, Christian Popow, Margit Pohl, and Markus Rester. Gravi++: Interactive Information Visualization to Explore Highly Structured Temporal Data. *J. UCS*, 11(11):1792–1805, 2005.

[18] Jr. Hosmer, David W., Stanley Lemeshow, and Rodney X. Sturdivant. Assessing the Fit of the Model. In *Applied Logistic Regression*, pages 153–225. John Wiley & Sons, Inc., 2013.

[19] Robert A. Jenders. Chapter 15 - Decision Rules and Expressions A2 - Greenes, Robert A. In *Clinical Decision Support (Second Edition)*, pages 417–434. Academic Press, Oxford, 2014.

[20] Rod Johnson, Juergen Hoeller, Keith Donald, Colin Sampaleanu, Rob Harrop, Thomas Risberg, Alef Arendsen, Darren Davison, Dmitriy Kopylenko, Mark Pollack, Thierry Templier, Erwin Vervaet, Portia Tung, Ben Hale, Adrian Colyer, John Lewis, Costin Leau, Mark Fisher, Sam Brannen, Ramnivas Laddad, Arjen Poutsma, Chris Beams, Tareq Abedrabbo, Andy Clement, Dave Syer, Oliver Gierke, Rossen Stoyanchev, Phillip Webb, Rob Winch, Brian Clozel, Stephane Nicoll, and Sebastian Deleuze. 7. The IoC container. https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html, March 2017.

[21] Robert S. Ledley and Lee B. Lusted. Reasoning Foundations of Medical Diagnosis. *Science*, 130(3366):9–21, July 1959.

[22] Lemeshow S and Jean-Roger Le. Modeling the severity of illness of icu patients: A systems update. *JAMA*, 272(13):1049–1055, October 1994.

[23] Sara C. Madeira and Arlindo L. Oliveira. Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(1):24–45, January 2004.

[24] Sara C. Madeira, Miguel C. Teixeira, Isabel Sa-Correia, and Arlindo L. Oliveira. Identification of Regulatory Modules in Time Series Gene Expression Data Using a Linear Time Biclustering Algorithm. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 7(1):153–165, January 2010.

[25] Michael E. Matheny and Lucila Ohno-Machado. Chapter 11 - Generation of Knowledge for Clinical Decision Support: Statistical and Machine Learning Techniques A2 - Greenes, Robert A. In *Clinical Decision Support (Second Edition)*, pages 309–337. Academic Press, Oxford, 2014.

[26] Clement J. McDonald, J. Marc Overhage, William M. Tierney, Paul R. Dexter, Douglas K. Martin, Jeffrey G. Suico, Atif Zafar, Gunther Schadow, Lonnie Blevins, Tull Glazener, Jim Meeks-Johnson, Larry Lemmon, Jill Warvel, Brian Porterfield, Jeff Warvel, Pat Cassidy, Don Lindbergh, Anne Belsito, Mark Tucker, Bruce Williams, and Cheryl Wodniak. The Regenstrief Medical Record System: A quarter century experience. *International Journal of Medical Informatics*, 54(3):225–253, June 1999.

[27] A. Najjar, C. Gagné, and D. Reinharz. Two-Step Heterogeneous Finite Mixture Model Clustering for Mining Healthcare Databases. In *2015 IEEE International Conference on Data Mining (ICDM)*, pages 931–936, November 2015.

[28] Lekha Narra, Tony Sahama, and Peta Stapleton. Clinical data warehousing for evidence based decision making. *Studies in Health Technology and Informatics*, 210:329–333, 2015.

[29] René Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651–654, September 2003.

[30] David S. Pieczkiewicz, Stanley M. Finkelstein, and Marshall I. Hertz. Design and Evaluation of a Web-Based Interactive Visualization System for Lung Transplant Home Monitoring Data. *AMIA Annual Symposium Proceedings*, 2007:598–602, 2007.

[31] C. Plaisant, R. Mushlin, A. Snyder, J. Li, D. Heller, and B. Shneiderman. LifeLines: Using visualization to enhance navigation and analysis of patient records. *Proceedings of the AMIA Symposium*, pages 76–80, 1998.

[32] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, May 2006.

[33] Niels Provos and David Mazières. A Future-adaptive Password Scheme. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '99, pages 32–32, Berkeley, CA, USA, 1999. USENIX Association.

[34] Lawrence R. Rabiner. Readings in Speech Recognition. pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[35] Alexander Rind. Interactive Information Visualization to Explore and Query Electronic Health Records. *Foundations and Trends® in Human–Computer Interaction*, 5(3):207–298, 2013.

[36] Edward Hance Shortliffe. *Computer-Based Medical Consultations, MYCIN*. Elsevier, 1976.

[37] Richard Smith. What clinical information do doctors need? *BMJ*, 313(7064):1062–1068, October 1996.

[38] Alejandro Vaisman and Esteban Zimányi. Data Warehouse Concepts. In *Data Warehouse Systems*, Data-Centric Systems and Applications, pages 53–87. Springer Berlin Heidelberg, 2014.

[39] Alejandro Vaisman and Esteban Zimányi. Extraction, Transformation, and Loading. In *Data Warehouse Systems*, Data-Centric Systems and Applications, pages 285–327. Springer Berlin Heidelberg, 2014.

[40] Alejandro Vaisman and Esteban Zimányi. Introduction. In *Data Warehouse Systems*, Data-Centric Systems and Applications, pages 3–11. Springer Berlin Heidelberg, 2014.

[41] Alejandro Vaisman and Esteban Zimányi. Logical Data Warehouse Design. In *Data Warehouse Systems*, Data-Centric Systems and Applications, pages 121–178. Springer Berlin Heidelberg, 2014.

[42] Taowei David Wang, Catherine Plaisant, Alexander J. Quinn, Roman Stanchak, Shawn Murphy, and Ben Shneiderman. Aligning Temporal Data by Sentinel Events: Discovering Patterns in Electronic Health Records. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 457–466, New York, NY, USA, 2008. ACM.

[43] Taowei David Wang, Catherine Plaisant, Ben Shneiderman, Neil Spring, David Roseman, Greg Marchand, Vikramjit Mukherjee, and Mark Smith. Temporal summaries: Supporting temporal categorical searching, aggregation and comparison. *IEEE transactions on visualization and computer graphics*, 15(6):1049–1056, 2009 Nov-Dec.

[44] Vivian L. West, David Borland, and W. Ed Hammond. Innovative information visualization of electronic health record data: A systematic review. *Journal of the American Medical Informatics Association*, 22(2):330–339, March 2015.

[45] Ian H. Witten, Eibe Frank, and Mark A. Hall. Chapter 1 - What's It All About? In *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 3–38. Morgan Kaufmann, Boston, 2011.

[46] Ian H. Witten, Eibe Frank, and Mark A. Hall. Chapter 6 - Implementations: Real Machine Learning Schemes. In *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 191–304. Morgan Kaufmann, Boston, 2011.

[47] Colin Yates, Seth Ladd, Steven Devijver, and Darren Davison. Chapter 2 - Spring Fundamentals. In *Expert Spring MVC and Web Flow*, pages 7–19. Apress, New York, NY, USA, 2006.

# Usability Test For the Biclustering Visualization Mechanism

Here we present the usability test used to evaluate the visualization mechanism of Umedicine's CDS module, in Portuguese.