Medical data visualization module*

Extended Abstract†

Miguel Neves Neto
Instituto Superior Técnico
miguel.neves.neto@tecnico.ulisboa.pt

ABSTRACT

Storing medical information about a patient and displaying that information or its analysis according to a physician's desires through easy to understand graphics during a diagnosis, is very important in order to provide the best possible healthcare service. Nowadays, there are a wide variety of medical software applications to assist physicians during appointments. However, the majority of them do not display medical information through graphics, meaning that the physicians need to search themselves for the information they desire and then perform the analysis as opposing to just look at a graphic where all of that is clear. Therefore, we proposed and developed a medical data visualization module for the *Umedicine* application, which allows the physicians to consult and monitor the evolution of several patient's health parameters. This module offers two functionalities: Graphical representation of the medical data regarding the historic of medical exams; Dashboard offering a graphical representation of the evolution of several health parameters allowing the physician to create graphics and save them to the dashboard. Finally, we validated our solution by conducting user tests. The results of these tests were very satisfatory as only three users commited errors and each one commited just a single one. Regarding the usability questionnaire that the users were asked to fill, 99.3% of the users select one of the two best possible grades for each question presented. These shows the satisfaction of the users with our solution.

KEYWORDS

Medical Software Application, Medical Data Visualization Module, Data Analysis, Medical Graphics, Diagnosis

1 INTRODUCTION

With the ongoing advance of technology, medical institutions in general have been expressing their needs for medical software that would greatly improve the efficiency and quality of the healthcare service provided. With this expected interest of the healthcare providers, the market of medical software applications has been quickly growing, with a wide variety of software applications that are now available.

Medical software applications are used by different types of users, like administrators, physicians and patients, each one of them having different needs and requirements that need to be satisfied. However, considering the graphical visualization of information, we can say that the majority of medical software applications are more focused in the needs of one type of user: the hospital administrator. The medical software applications support a wide range of administrative functionalities in order to address the needs of administrators: billing modules with graphical analysis tools and financial performance indicators such as payment velocity. Regarding the healthcare service, these applications present generic medical information, offering tools for patient management, appointments management and e-prescribing.

Considering the needs of physicians, we can say that there are three main requirements for a medical software application, in order to enhance his/her performance: i) the application gathers all the appointment data and stores it in a digital format; ii) must allow a graphical visualization of the evolution of several medical parameters regarding a patient; iii) must allow analysing the data and showing the result of the analysis through graphics. Most of the existing medical software applications are not flexible regarding the graphics availability, limiting the access to graphics to a specific and isolated module or area of the application, which usually is a generic dashboard. The generic dashboard allows the creation of predefined graphics, such as blood pressure, over the patients' population. However, limiting the access to graphical data to only a specific area of the application is not practical as the physician needs to change between pages in order to access the information he/she desires to see. Regarding the analysis of medical information, it is required by the physicians, that an application provides tools to analyse the stored medical information and present those results through easy to understand graphics. Most of the existing medical software applications, such as Meditouch EHR[1] and Kareo[2], only offer tools to graphically analyse administrative data. There are software tools that enable the visualization of medical data through graphics, however these do not comply with the needs of the physicians, as it is only possible to create graphics over a population of patients and the graphics available to create are predefined and very limited.

The purpose of this thesis is to develop a medical data visualization module that properly assists Urology physicians when performing a diagnosis. This module will be focused on requirement ii) of the physicians requirements, described in the previous paragraph, however the developed functionalities should also be possible to apply for requirement iii). In order to answer requirement ii), this module must present the patient's data relevant for a diagnosis, providing a dashboard containing some predefined graphics considered relevant as well as the option to create and save new graphics by the physician according to his/her needs. However, if the physician intends to have a graphical look at the evolution of one or more parameters within a specific patient's exam, this should also be possible. In order for this data to be quickly and efficiently accessed

*Produces the permission block, and copyright information
†The full version of this thesis is available as `ist173837dissertation.pdf` document

[1] https://www.healthfusion.com/ehr-software/
[2] http://www.kareo.com/

by the physician, this module should present a user-friendly interface. Our solution will be integrated in the Umedicine application, described in Section 1.1.
This document is organized into five chapters. Section 2 analyses other medical software applications according to the requirements described in Section 1.2. In this section, we also analyse some data visualization tools concluding with the choice of the one to use for the development of our solution. Section 3 starts by presenting the developed solution, naming the technologies used, the code developed and the functionalities of the solution. Section 4 describes the validation that we performed for the new medical data visualization module of Umedicine. Finally, Section 5 concludes this thesis.

1.1 Umedicine: Tool to Support Urology Appointments

Umedicine is a medical sofware application, focused in the needs and requirements of the physicians, that offers functionalities to store, update and access appointment data and then to manage and analyse that data in a way that facilitates the work of the physicians as well as brings benefits to their patients, through a better quality of service offered. This software application is being developed according to the requirements specified by a physician of this specialty. Umedicine can be accessed by any device through a web browser, requiring internet connection. Umedicine, currently lacks a module that offers a graphical visualization of the data collected and stored.

1.2 Requirements

Considering the problem identified in Section **??** we identified the following requirements for a medical data visualization module:

- *Visualization of medical information*: It should support visualizing medical information through graphics, in particular:
  Patient's health condition evolution: It should allow tracking the evolution of a patient's health condition or parameter (PSA, for example) through one or more dimensions.
  Graphic Availability: The access to the graphical information should not be limited to a single module or area (dashboard) of the application. It should be also possible the access to graphical information in every page that contains relevant information.
  Customizable: It must be possible for the physician to modify or create graphics by choosing which patient data he/she wants to see.
  It should support several types of graphics to show the medical data: bar graphics, linear graphics, etc.

2 RELATED WORK

In this section we perform an analysis of the existent medical software tools according to the requirements described in section 1.2. Following this analysis, we conduct another analysis on the data visualization tools by comparing their functionalities.

2.1 Medical software applications

In this section we present the summary of the analysis of the medical software applications regarding the requirements listed in Section 1.2.

Table 1 summarizes the software applications described in this chapter using, as comparison criteria, the requirements identified in Section 1.2.

		Visualization of medical information		
		Patient health condition evolution	Graphic Availability	Customizable
My MedicineOne		N/D	N/D	N/D
eClinical Works	EMR/PM	✔	✗	✗
	CCMR	✔	✗	✗
AllScripts Professional EHR		N/D	N/D	N/D
Kareo		N/D	N/D	N/D
MediTouch EHR		N/D	N/D	N/D
NextGen Healthcare		✔	✗	✔

Table 1: Requirements fulfilled by the medical software applications.

As we can observe in Table 1, the majority of the analysed medical software applications do not show medical information through graphics, being therefore not defined (N/D) for this requirement. Regarding the medical software applications that present medical information through graphics, we can see that all of them enable monitoring a patient's health condition evolution. Considering the graphic availability, in all of the applications, these graphics are only accessible through a single area, the generic dashboard. In addition, only NextGen Healthcare enables the physician to create customizable graphics, being possible to choose the type of graphic, for example. Therefore, we conclude that none of the analysed medical software applications fulfills all of the requirements.

2.2 Data visualization tools

In this section we present an analysis of the data visualization tools based on comparisons between each of them.

Table 2 presents a comparison of the several data visualization tools according to some important parameters for this thesis.

As we can observe in Table 2, all of the data visualization tools enable chart customization when creating a chart. All of these tools allow data extraction from databases which is crucial for this thesis, however, Chartblocks only allows it on a paid version. So, now considering D3.js, Highcharts and Tableau Public, from these three only D3.js offers non-limited chart types. Highcharts and Tableau Public only offer some predefined chart types to choose from, being therefore more limited than D3.js. Concluding, D3.js is the best choice of a data visualization tool to use in this thesis.

	Type	User	Chart Types	Customization	Data source
D3.js	JavaScript Lib	Developer	Custom	Yes	objects, files, database
Highcharts	JavaScript Lib	Developer	Fixed	Yes	objects, files, database
Tableau Public	free software	Non-developer	Fixed	Yes	files, database
Chartblocks	online chart builder	Non-developer	Fixed	Yes	files, database

Table 2: Comparison of the data visualization tools.

3 SOLUTION

This section details the development and the functionalities of the medical data visualization module.

3.1 Developed Solution

This section contains all the information about what was developed regarding the purpose of this thesis, presenting the final solution with all the features that are currently available to use.

3.1.1 Code development. This section explains the essential steps which were taken when developing the code for the solution. The programming languages, libraries and frameworks applied during the development of our solution, were: HTML, CSS, JavaScript, jQuery, D3.js and the Spring framework.

Entity creation

In order to develop our solution, first of all it was needed to create an entity that would store the data gathered from the database when extracting it through SQL queries.We developed an entity called *Parametros* that contains as attributes the most important parameters we needed to retrieve from the database, which were the following:

- *nprocesso* : pacient's process number;
- *idExame* : ID of the exam;
- *data*: date of the gathered value for the parameter;
- *valor* : value of the parameter;
- *nomeP* : name of the parameter to be retrieved.

DAO functions

We needed to create DAO functions in order to actually retrieve the data from the database. Therefore, we created *ParametrosPacienteDAO* and *ParametrosPacienteDAOImpl* classes. The *ParametrosPacienteDAO* class is an interface containing all the methods required. The *ParametrosPacienteDAOImpl* class implements the previously mentioned interface.
There were several functions created in order to fetch the data for each parameter and also to fetch data for the graphics' legends. Essentially these functions execute SQL queries over the database and then that information is stored in a *ResultSet*. This *ResultSet* is a list in which each element contains one record of the data retrieved by the query. We then search this list and create a list of Parametros which contains in each element the data needed to construct the graphics. Finally, we return this list of *Parametros*.

Services

We created a service class, named *ParametrosPacienteService*, which essentially has several methods, each of which calls a specific method within the already developed DAO class. For this purpose, this service class has an instance of the *ParametrosPacienteDAO* interface.

Web Service

We created a Web Service class named *DashboardPacienteWebServices*. This class is the one responsible for triggering all the process of data gathering from the database. For this purpose, it contains an instance of the *ParametrosPacienteService*. Then, in each of its functions, it calls a specific service (depending on the data we need to obtain), stores the result in a list of *Parametros* and returns it.
Spring is essential in this web service class, as with it, it is possible to make every function produce JSON results. This is essential for the creation of graphics, as with the D3.js library, we need the data to be structured in JSON. Spring also allows us to define a value (url) for each function, which is also very important, as it is needed in order to correctly call D3.js methods.

Using D3.js for the creation of graphics

We created and modified two javascript pages, one for the dashboard and the other one, which was already created, for historic. Within each of these pages we then started the process of creating graphics with the D3.js library. The process of creating a graphic with D3.js, essentialy complies three steps: SVG creation; Using D3.json function; Adding a legend.
Regarding the first step - SVG creation - this is the step that defines the area which the graphic will occupy. SVG is the rectangle where the graphic with all its components will fit. The second step - Using *D3.json* function - is the most essential one for creating a graphic. The *D3.json* function requires as arguments an URL and an option callback function.
With the*D3.json* function we define all the characteristics of a graphic. The URL that the function requires as argument is the value we specified for the each function of the Web Service and in the callback function we put a function with the array we wish to store in the data from the database.
For the third step - Adding a legend - we needed to have previously defined some colors, stored in variables, so that we could use them when adding each rectangle of the legend to the graphic. These

colors were specified in RGB. Finally we also had to add some functions to resize the graphics correctly when, for example, the user resizes the window.

Refactoring

The methods for creating an svg, creating the corresponding graphic and resizing the graphics were refactored. Of course this, as expected, brought more difficulties such as for example, graphics with more parameters woud need more data from the database and therefore we needed to add some extra verifications.
Although, from all the refactored methods, the one considered most challenging, was refactoring the creation of legends. The method of creating legends seemed very strict and it required a long analysis process in order to define the best way to refactor this process. The fact that graphics can have different number of parameters, is of course, a challenge when refactoring this function. We came to the conclusion that the parameters should be stored in an array and also the colors to be applied in each rectangle of the legend. The most complex step was applying the arrays correctly within the process of creating the legend. There were some more steps taken in order for this to work correctly and after all of the analysis and testing was done, the function ended up by working as intended.

3.2 Functionalities

This section presents all the working functionalities of our solution. This section is divided in two subsections: the Dashboard and the Historic.

3.2.1 Dashboard. The Dashboard is the core functionality of the solution. It is available in the patient's page of Umedicine, which presents that patient's medical information.
By default, the Dashboard page presents five predefined tabs. The first four tabs correspond to predefined graphics that were confirmed relevant by the physician collaborating on this project. The fifth tab allows the user to create another graphic showing the variation of one or two parameters of the patient's data. The user-specified graphics can be persistently added to the dashboard or they can be seen only once.

Figure 1: Questionnaire graphic within the dashboard page.

The four main predefined graphics are: Qmax, PSAs, Prostate Volume and Questionnaire(seen in figure 1). Each of these graphics is presented over the time dimension.
When accessing the fifth tab, named *Criacao de grafico*, the physician is presented some dropdown menus with predefined options for the x and y axis. It is also possible to monitor the variation of two parameters over time in the same graphic by adding an option from the dropdown menu to both y-axis. Once the parameters are selected, a name for the graph is automatically generated, which the physican can modify if he thinks it's not suitable.
After this process, the physician is then allowed to create the graphic, by selecting the corresponding button, and also to save that graphic for posterior uses/consults. After a graphic is saved, a new tab appears with the name of the graphic which was created and it is possible to delete that tab if the physician considers that the graphic is no longer important or relevant to keep.For as long as the physician does not erase the tab, it will appear in the posterior accesses to the patient's dashboard.

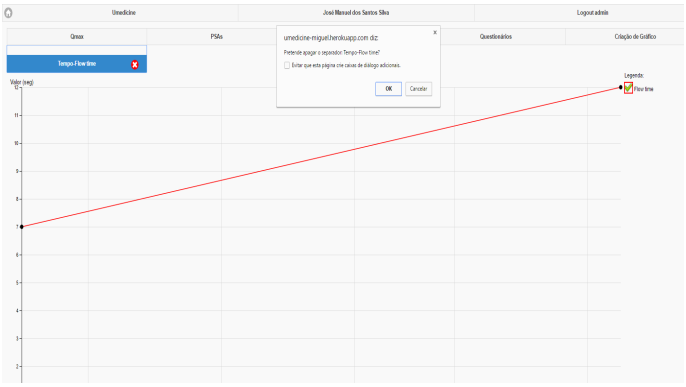

Figure 2: Example of the process of the deletion of a created graphic.

Figure 2 presents a graphic that was created by the physician to monitor the evolution of the Flow-time against Time. As can be seen, after clicking on the "x" button to delete the tab, a dialog box appears in order to confirm the physician desire to erase that graphic from the dashboard.

3.2.2 Historic. The Historic functionality consisted on allowing the physician to consult the variation of values for the parameters of a specific exam. Each parameter is associated to an urologic subcategory and their values were only shown by means of a table.
To this functionality that already existed, it was added to the possibility of checking the variation of one or more parameters through graphics. This graphical functionality is divided in two main features. The first feature allows monitoring the variation of the parameters belonging to a specific urologic subcategory and the second feature allows consulting the variation of one or more selected parameters.
In order to develop this functionality it was needed to gather the values presented in the table. For this, we had to create several arrays where, when creating the table, we would store those values

in the arrays. Then, for the first feature, we had to create buttons, which generate a pop-up, for each urologic subcategory with HTML and CSS, and verify, with JavaScript, if any of those buttons was being clicked and which button it was. Finally, considering the chosen subcategory, it is called a specific function that then gathers the desired parameters values (based on the subcategory) from the arrays and creates the graphics, taking into consideration the range of values for the determination of the x-axis and y-axis, lines of values, points and legends.
For the second feature, we had to associate checkboxes to every parameter and a general one that would select all of the others. Additionally, we also had to create a create button that would generate a pop-up which would show the variation of the selected parameters.This was done with HTML and CSS languages. Then, it was created an array that would contain the names of the selected parameters. Finally, it was associated a function to the create button that, essentially, verifies which parameters are in the array and then gathers the values for those parameters. After that, the function creates the graphic's elements.
The access to this feature is done through the patient's page, under the "Exames Auxiliares de Diagnostico" section, by clicking on the desired exam and then selecting the "Historico" button in the page that follows, to go to the feature's page.

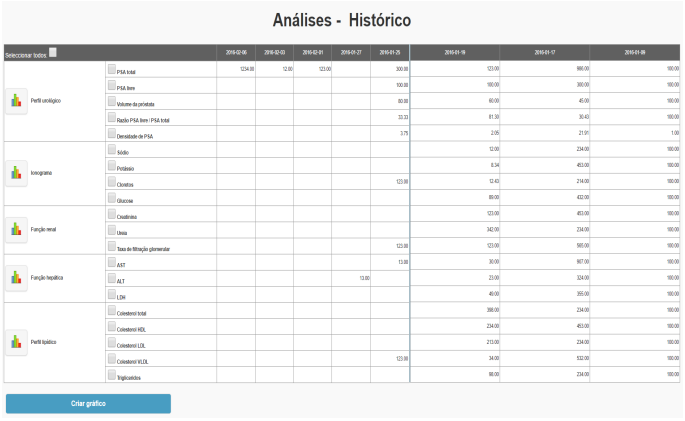

Figure 3: Example of the screen for the Historic functionality for a specific selected exam.

Figure 3, presents the Historic screen. Within this figure it is possible to verify the two, previously mentioned, new features: monitorization of the variation of the parameters belonging to a specific urologic subcategory; display of the variation of one or more selected parameters.
For both features, the graphics shown in the generated pop-up are grouped by unity of measurement. This means that, for example, all the parameters that have the unity "ng/mL" will appear in the same graphic. Additionally, each page of the pop-up may contain up to four graphics and, in case there are more than four graphics, it is generated another page.

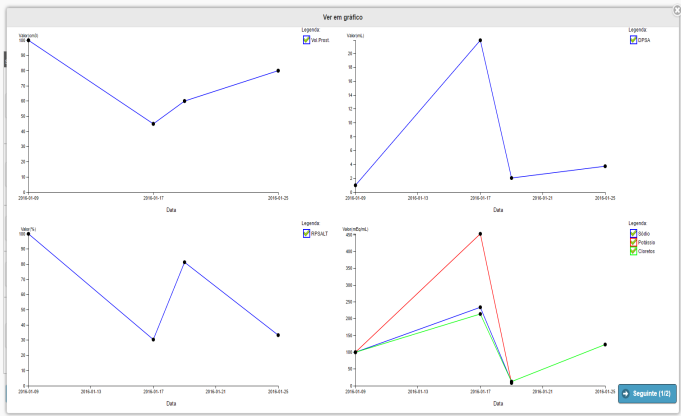

Figure 4: Example of the pop-up window displaying the evolution of the historic for all the parameters.

Figure 4, presents the pop-up window which is generated after selecting all the parameters and clicking the button to create the graphic. It is also possible to verify what was already mentioned before, which is the grouping of the parameters by their unity. In this case, the pop-up window contains more than one page as each page only presents four graphics.

4 VALIDATION

This chapter refers to the evaluation and validation of the developed solution. It starts by describing the the type of evaluation conducted and also what was evaluated. Finally, it describes the results of the evaluation, concluding on these results.

4.1 Validation of the user interface

To validate and evaluate this module's user interface, we performed tests with users.
The user tests evaluation was based in the methodology presented in *"Task Centered User-Interface Design: A pratical introduction"* by Clayton Lewis and John Rieman.
This evaluation was performed by ten users with ages between 20-50 years old, without any medical background. Initially, it was provided to the users a brief explanation of the purpose of the evaluation and also the purpose of the application and of what was developed. Then, these users were given tasks to execute within the Medical Data Visualization Module, where, for each task, we used metrics such as: the time needed to complete it; number of help requests; number of mouse clicks; and the number of errors commited. Before executing the tasks, none of the users had had any previous contact with application.
After each task was completed the users were asked to fulfill a questionnaire where for that task they would evaluate the difficulty on understanding the purpose of the task and the difficulty on completing it.
In the end, the users filled a satisfaction questionnaire where they manifested the utility degree of the user interface.

4.2 User tasks

This section presents the tasks that were performed by the users during the evaluation of the user interface of our solution. Every user started each task from the patient page. After the conclusion of each task, the user went back to the patient page.
The performed tasks were the following:

- *Task 1: Check the evolution of the patient's Qmax;*
- *Task 2: Check the evolution of "volume da prostata" and Questionario IIEF and IPSS;*
- *Task 3: Create and save two graphics. The first one to analyse the variation of Flow time across time and the second one to analyse the relation between volume da prostata and PSA total;*
- *Task 4: Check the graphic containing the evolution of the flow time across time;*
- *Task 5: Erase the tab corresponding to the graphic containing the variation of flow time across time;*
- *Task 6: Check the graphic of Perfil urologico's historic from the patient's analysis;*
- *Task 7: Check the graphic of PSA total, PSA livre, Sodio, Creatinina, AST, LDH and Colesterol LDL's historic from the patient's analysis;*
- *Task 8: Check the graphic of all parameters' historic from the patient's analysis.*

4.3 Questionnaires

This section explains the purposes of the questionnaires, already mentioned in Section4.1, used during the user tests.
The first questionnaire[3], is a very simple one and was created with the purpose of evaluating the difficulty on understanding the purpose of each task and the difficulty on completing it. To measure the difficulty on completing the tasks it was used the SEQ (Simple Ease Question)[4] which defines a 7-point rating scale to better measure this matter. This questionnaire has a question per task. Whenever the user finishes a task, he has to answer the question concerning the task.
The second questionnaire[5] was created with purpose of evaluating the usability of the user interface and satisfaction of the user with the system, and was based on the already defined CSUQ (Computer System Usability Questionnaire)[6], with some minor changes in order to better adapt to the context of this thesis.

4.4 Results

This section contains the result of the user tests.
This section is organised into two parts. The first one analyses the metric results collected during the execution of each task. The second part presents an analysis over the results of the questionnaire regarding the usability of the user interface.

[3]https://docs.google.com/forms/d/1PuyPVUL4_-7wZ7yk6jWyna3aSfL3Jt2bjTktV0ZU8EA/viewform?edit_requested=true
[4]https://measuringu.com/seq10/
[5]https://docs.google.com/forms/d/1WZ32wHfU0dJUpdeosS7TmdwqHQ8DGRsI8ecmjkysyQc/viewform?edit_requested=true
[6]http://garyperlman.com/quest/quest.cgi?

4.4.1 Metrics results analysis. In this section we analyse the metrics results gathered from the users, when performing the user tasks.

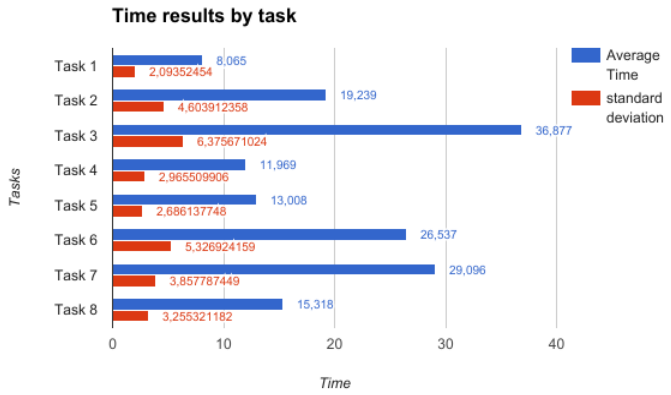

Figure 5: Table displaying the gathered results for the metric: time to complete the task.

Figure 5 presents the average time needed for the users to complete each task as well as the standard deviation of the users' times over the average time.

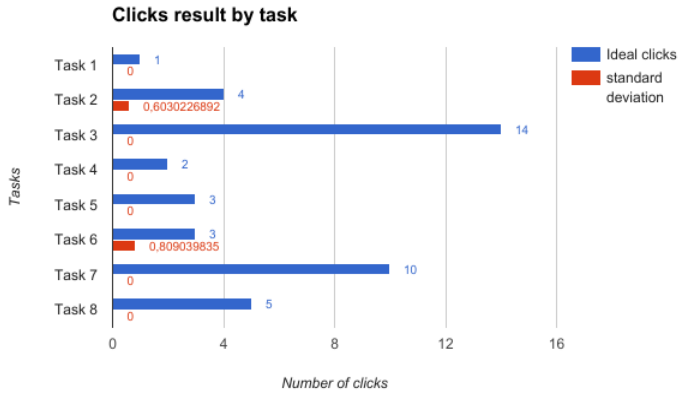

Figure 6: Table displaying the gathered results for the metric: number of clicks needed to complete the task.

Figure 6 presents the ideal clicks needed for the users to complete each task as well as the standard deviation of the users' number of clicks over the ideal number.

These results will now be used to conduct an analysis of the metrics for each group of tasks: grouped according to the difficulty of the tasks. There are two groups of tasks: very simple, and simple tasks.

Very simple tasks

The tasks that are part of the very simple tasks group are:

- **Task 1** - Check the evolution of a patient's Qmax.
- **Task 4** - Check the graphic containing the evolution of the flow time across time.
- **Task 5** - Erase the tab corresponding to the graphic containing the variation of flow time across time.
- **Task 7** - Check the graphic of PSA total, PSA livre, Sodio, Creatinina, AST, LDH and Colesterol LDL's historic from the patient's analysis.

During the realization of any of these four tasks, no user commited any error or asked for any help. these tasks were concluded very quickly, being the maximum average time 29 seconds corresponding to task 7 and the standard deviation around 3 seconds. Additionally, there was no standard deviation when comparing the number of clicks needed and the ideal clicks for any of these tasks, which affirms that all users completed the tasks using the ideal number of clicks.

Simple tasks

The tasks that are part of the simple tasks group are:

- **Task 2** - Check the evolution of volume da prostata and Questionario IIEF and IPSS.
- **Task 3** - Create and save two graphics. The first one to analyse the variation of Flow time across time and the second one to analyse the relation between volume da prostata and PSA total.
- **Task 6** - Check the graphic of Perfil urologico's historic from the patient's analysis.
- **Task 8** - Check the graphic of all parameters' historic from the patient's analysis.

During the realization of any of these four tasks, no user asked for any help. However, when analysing the number of errors commited when performing these four tasks, we obtained 3 users which commited errors. One of the users commited one error during the realization of task 2. This user's error was the fact that he clicked on another section in the patient page (the diagnostic exams section), however the user quickly understood, due to the information presented, that that page was not the correct one, and remembered that the Dashboard contained in one tab that specific graphic. The other two users commited errors when performing task 6. One of these two errors was due to lack of concentration, as the user clicked by mistake in the wrong exam and once, he accessed the wrong page, he understood that he had clicked on the wrong exam and then clicked on the correct one. The other error was the fact that the user clicked on the Dashboard tab instead of the analysis exam in the diagnostic exams section, the user then re-read the task and understood quickly what he had to do.
The errors committed lead to more time in completing the task, which increased the average time taken, staying at 37 seconds. These errors also lead to a slight standard deviation when comparing the number of clicks needed and the ideal clicks for any of these tasks.

4.4.2 Usability of the user interface and user satisfaction questionnaire. This section contains a summary of the results obtained with the Usability of the user interface and user satisfaction questionnaire.

Analysis of the results

This section starts by providing a summary of the results obtained during the user tests evaluation on which will be conducted a brief analysis.

Questions

This section describes all the questions within the questionnaire which was based on the already defined and known CSUQ (Computer System Usability Questionnaire) with some adaptations. This questionnaire is composed by 13 questions which are the following:

- *1. Overall, I am satisfied with how easy it is to use this system.*
- *2. It was simple to use this system*
- *3. I can efficiently complete my work using this system.*
- *4. I am able to complete my work quickly.*
- *5. I feel comfortable using this system.*
- *6. It was easy to learn to use this system.*
- *7. The information provided with this system is clear.*
- *8. It is easy to find the information I needed.*
- *9. The information is effective in helping me complete the tasks.*
- *10. The organization of information on the system screens is clear.*
- *11. The interface of this system is pleasant.*
- *12. I like using the interface of this system .*
- *13. Overall, I am satisfied with this system .*

Overall analysis

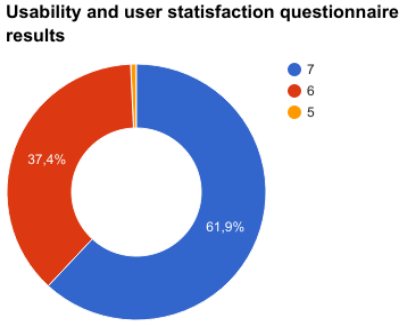

Figure 7: Summary of the results of the questionnaire regarding the usability of the user interface and the user satisfaction.

Figure 7, presents a summary of the results of the questionnaire. This figure was generated by calculating the average percentage (%) of each classification grade for all the 13 questions, presenting therefore the most frequent classifications throughout the entire questionnaire.

Through the analysis of the results displayed in figure 7 it can be concluded that the user tests were very satisfying as the best two possible grades, which correspond to grade 6 and 7, are the ones with the most percentage throughout all the questions. Grade 7, which is the best grade, clearly dominates the results of the questionnaire with 61,9%, which is almost the double of the average percentage of grade 6 (37,4%).

The questions that largely contributed to the dominating percentage of grade 7 were:

- *6. It was easy to learn to use this system.*
- *12. I like using the interface of this system.*
- *13. Overall, I am satisfied with this system.*

By analysing the results for each of these three questions, we can affirm that:

- Our solution is easy to learn;
- Our solution offers a user-friendly interface;
- Overall, the users were satisfied with the solution.

Some of the questions that contributed to the percentages assigned to grade 6 (37,4%) and grade 5 (0,7%) ,when analysing the results of the questionnaire were the following:

- *1. Overall, I am satisfied with how easy it is to use this system.*
- *2. It was simple to use this system*
- *8. It is easy to find the information I needed.*

By analysing the results for each of these three questions, we can affirm that:

- Our solution is easy and simple to use;
- Our solution presents clear and intuitive user-interface.

5 CONCLUSIONS

In this thesis we proposed a Medical data visualization module for the Umedicine application. Our solution has as purpose presenting a patient's medical information through easy to understand graphics. These graphics enable the physicians to analyse the evolution of the desired parameters during a patient's appointment, supporting them in the decision of the best treatment to be applied.

The main objective of this thesis was to address the conclusions of the analysis from the State of the art which stated that: the majority of the medical software applications provide graphics focused on the administrative aspects and when considering the medical data, they are very limited. In order for our solution to deliver a good graphical analysis over the medical data, we used as data visualization tool, the D3.js library, which is a powerful library and complies to the requirements described in Section 1.2, being highly customizable.

The solution developed in this thesis was the following:

- Medical data visualization module with following features:

  Dashboard: the Dashboard page presents five predefined tabs. The first four tabs correspond to predefined graphics confirmed relevant by the physician collaborating on this project. The fifth tab allows the user to create a graphic showing the variation of one or two parameters of the patient's data. The user-specified graphics can be persistently added to the dashboard of the user or they can be seen only once.

  Historic: To the historic functionality that already existed, which showed the variation of values for the parameters of a specific exam through a table, we added the possibility to check the variation of one or more parameters through graphics. This graphical functionality is divided in two main features. The first feature allows monitoring the variation of the parameters belonging to a specific urologic subcategory and the second feature allows consulting the variation of one or more selected parameters.

Finally, in order to validate the solution developed and its functionalities, we conducted a user tests evaluation covering each functionality within this new module, in order to measure the usability of the user-interface developed through each user's performance during these tests. As future work, we propose more graphics' customization and the possibility to export these graphics to JPG or PNG.

REFERENCES

[1] Riccardo Bellazzi and Blaz Zupan. Predictive data mining in clinical medicine: current issues and guidelines. *International Journal of Medical Informatics*, 77(2):81–97, 2008.

[2] Ashish K Jha, Catherine M DesRoches, Eric G Campbell, Karen Donelan, Sowmya R Rao, Timothy G Ferris, Alexandra Shields, Sara Rosenbaum, and David Blumenthal. Use of electronic health records in us hospitals. *New England Journal of Medicine*, 360(16):1628–1638, 2009.

[3] Jeffrey A Linder, Jun Ma, David W Bates, Blackford Middleton, and Randall S Stafford. Electronic health record use and the quality of ambulatory care in the united states. *Archives of internal medicine*, 167(13):1400–1405, 2007.

[4] Malcolm Maclean. *Data Driven Documents D3.JS Tips and Tricks.*

[5] Jonathan Makanjuola and Zakarian Artaches. The emerging use of smartphones apps in urology. *The Journal of the American Medical Association*, 12(7):78–80, 2012.

[6] Jakob Nielsen. Usability inspection methods. In *Conference companion on Human factors in computing systems*, pages 413–414. ACM, 1994.

[7] Mahadev Satyanarayanan. Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

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