ITMB
COC252
F011321

EHR System Modernization in the Republic of Moldova

by

Calin Corcimaru

Supervisor: Dr. Georgina Cosma

Department of Computer Science
Loughborough University

May 2025

Abstract

Abstract to be added

Acknowledgements

Acknowledgements to be added

Contents

Chapter 1

Introduction

1.1 Background

The Republic of Moldova is a small country in Eastern Europe that borders Romania and Ukraine, with a current population of 2.4 million people (National Bureau of Statistics of the Republic of Moldova, 2024). Since its independence in 1991, Moldova has faced a number of challenges, including political instability, corruption, and economic difficulties which have left Moldova as one of the poorest countries in Europe (BBC, 2024).

Despite these challenges, Moldova has made significant progress in its digital transformation efforts, with the government launching a number of initiatives to modernize its public services and improve the quality of life for its citizens (E-Governance Agency, n.d.[a]). An example is the Citizen's Government Portal (MCabinet), which allows citizens to access personal information such as 'valid identity documents, social contributions and benefits, own properties, information about the family doctor and the health institution where the person is registered, tax payments and other information about the citizen-government relationship' (E-Governance Agency, n.d.[b]).

To continue supporting the existing transformation initiatives, Moldova's Cabinet of Ministers has recently approved the 'Digital Transformation Strategy of the Republic of Moldova for 2023-2030', which aims to transform the country into a digital society by 2030, with the ultimate goal of having 'all public services available in a digitalized format' (United Nations Development Programme, 2023).

1.2 Problem Statement

The healthcare sector in Moldova has also seen some transformations, with the introduction of a new electronic health record system (EHR) in 15 hospitals across the country in 2017, called 'Sistemul informațional automatizat „Asistența Medicală Spitalicească" (SIA AMS)' (Ciurcă, 2021). While the system has been successful in helping doctors access patient information more efficiently such as medical history, examinations, test results, and prescriptions, the system hasn't been updated since its inception in 2017 and there are still challenges that need to be addressed in 2024.

The first challenge occurs due to a lack of a nationally-wide integrated system – each hospital and clinic have their own, siloed, information system that contains the patient information, with no communication being made between systems in different hospitals (Ciurcă, 2021).

The second challenge lies with the user experience (UX) of the existing system – the current system feels old and isn't user-friendly, with a clunky interface that is difficult to navigate, not adhering to modern accesibility standards and being only accessible on legacy version of Edge, with no support for other browsers or devices (Ciurcă, 2021).

Finally, due to the current economic situation in Moldova, the government has not allocated any funds to update the system, and the hospitals and clinics that use the system do not have the resources to update it themselves.

1.3 The Client

To address these challenges, this project, in collaboration with "Nicolae Testemiteanu" State University of Medicine and Pharmacy in Moldova (USMF), seeks to develop a prototype for a modernized EHR system. The client, USMF, is a public university in Chisinau, Moldova, that offers a range of medical programs, including medicine, dentistry and pharmacy (USMF, 2023). Many of the faculty at USMF are also practicing doctors at hospitals and clinics across Moldova, and have first-hand experience with the current EHR system in place. As such, the client has expressed a need for a new EHR system prototype that is more user-friendly, accessible on a wider range of devices and browsers, and that can be customized to meet the specific needs of different hospitals, clinics, departments and users.

Chapter 2

Literature Review

This chapter will provide a review of the existing literature, which will be used guide the student in their planning and development efforts of the project.

As such, it will be covering the following areas:

- Software development methodologies
- Requirement gathering techniques
- Accesibility considerations
- Tech stack
- EHR systems
- Machine Learning Summarization models
- Speech Recognition models

2.1 Software development methodologies

2.1.1 Software Development Life Cycle

The Software Development Life Cycle (SDLC) is a process used to guide the development of software applications or systems (Ruparelia, 2010). The SDLC consists of multiple phases, each with its own set of activities and deliverables. Yas, Alazzawi, and Rahmatullah (2023) outline the phases of the SDLC as following:

1. Requirement gathering and analysis phase - This phase involves gathering and analyzing the requirements of the software to be developed. These requirements are gathered from the project stakeholders and saved in a specific document. Based on the requirements gathered, a development plan is created and a feasibility study is conducted.

2. Design phase - This phase involves representing the previously gathered requirements in a project design written in a more technical manner, that will later guide the developers to create and implement the software.

3. Implementation phase - This phase involves the actual development of the software. Additionally, some smaller unit tests may occur during this phase as parts of the software are developed.

4. Testing phase - This phase focuses on testing the software to ensure that it meets the requirements and is free of bugs. This phase may involve multiple types of testing, such as unit testing, integration testing, and system testing. Luo (2001) describes the different types of tests as following:

    - Unit testing - This type of test is done on the lowest level of the software, testing individual units or components of the software.
    - Integration testing - This type of test is performed on two or more units combined together, usually focusing on the interfaces between these components.
    - System testing - This type of test focuses on the 'end-to-end quality of the entire system', testing it as a whole based on the system requirement specification.

5. Maintenance phase - This phase involves the deployment and maintenance of the software. Additionally, this phase may include user acceptance testing, where the software is handed over to the end-users to ensure that it meets their needs (Luo, 2001).

2.1.2 SDLC Models

The literature describes several SDLC models that have been used in the development of software applications. Ruparelia (2010) and Yas, Alazzawi, and Rahmatullah (2023) outline the most common SDLC models:

- Waterfall Model
- V Model
- Spiral Model

- Iterative Model
- Agile model

Due to the nature of the project being different to traditional software development projects, and the student's familiarity with only Waterfall and Agile models, the next sections will only focus on these two models.

Waterfall Model

The Waterfall Model is probably the most well-known SDLC model. Waterfall is a linear model, where the development process is divided into distinct, sequential phases that follow the SDLC. As such, each phase must be completed before the next phase can begin.

The Waterfall Model's strengths lie in its simplicty of use, ease of understanding and providing a structured approach to a project (Alshamrani and Bahattab, 2015). An additional stregth of the Waterfall model that the authors note is its extensive documentation and planning, which is done in the early stages of a project, but also maintened throughout the project's lifecycle. These two factors also help minimize the overhead that comes with planning and management of a project, which in the case of Waterfall is done in the early stages of the project.

However, the Waterfall model is not perfect. One of its main weaknesses, mentioned by Alshamrani and Bahattab (2015), is its lack of flexibility in regards to change of requirements. As such, once the project leaves the requirements analysis or design phase, it may be difficult to make any changes to the project deliverable. Thus, this model is not suitable for projects where the requirements are not well understood or are likely to change. Finally, the deliverable is only available at the end of the project, so the end-users are unable to see the final product until the end of the project, nor can they provide any feedback during its development (Alshamrani and Bahattab, 2015).

Agile Model

Another well-known SDLC model is the Agile model. Taking its roots from the Agile Manifesto, it describes a different way of developing software from the Waterfall model, with a focus on:

> *Individuals and interactions over processes and tools,*
> *Working software over comprehensive documentation,*
> *Customer collaboration over contract negotiation,*
> *Responding to change over following a plan* (Cunningham, 2001).

The Agile model focuses on the ideas that requirements are not always well-known or cannot be predicted, accepting that change is inevitable and emphasis should be put on being able to accommodate any changes that may arise (Sunner, 2016). Similarly, as the author mentions, the focus of this methodology is on continuous delivery of software and value to the customer. As such, it is integral for an Agile project to have a close customer involvement in the development process, to ensure that constant feedback is received.

Agile does come with its own drawbacks. One of the main issues with Agile is its lack of documentation and formal planning, especially in the early stages of the project (Ruparelia, 2010). Consequently, the Agile methodology may not be suitable for large scale projects, where extensive documentation and planning are required (Sunner, 2016; Yas, Alazzawi, and Rahmatullah, 2023). Similarly, the Agile model may not be suitable for projects where the requirements are well understood, unlikely to change or where there may be strict regulations that guide how the project should be developed. Finally, Yas, Alazzawi, and Rahmatullah (2023) also mention that unfamiliarity with Agile frameworks could also be an impeding factor in the success of Agile projects, as the staff may not be familiar with Agile and require extensive training beforehand.

Agile has multiple frameworks that can be used to guide the development process, such as Scrum, Kanban, Lean, and Extreme Programming (XP). Alqudah and Razali (2018) mention that Scrum is the most widely used Agile framework in software development. As such, the next sections will focus on Scrum and Kanban, due to their popularity and student's familiarity with these frameworks.

Scrum is an Agile framework, with its core idea being the division of the project into smaller, manageable parts called sprints. Each sprint usually lasts between 2-4 weeks and each sprint focuses on delivering value to the customer through working software features (Alqudah and Razali, 2018; Sunner, 2016). Scrum employs a set of artifacts, that are used to guide the development process, such as the product backlog, sprint backlog, and burndown charts. Additionally, Scrum also has a set of roles, such as the Scrum Master, Product Owner, and the Development Team, that are responsible for the development process (Alqudah and Razali, 2018). As such, Scrum may be more suitable for projects with at least a couple of team members, as it requires a team and their roles to work together to deliver the project and adherence to specific events within the project lifecycle.

On the other hand, Kanban is not as prescriptive as Scrum is. Kanban focuses on visualizing the workflow of the project through a visual board with columns, swimlanes, etc. Additionally, Kanban helps limit the work in progress and maximize the flow of work through the system (Sunner, 2016). Kanban does not have specific roles or artifacts, but instead focuses

on the continuous delivery of value to the customer through smaller batching and daily prioritizations (Alqudah and Razali, 2018).

Finally, there is also a possibility of combining these two frameworks into one, called Scrumban. Nowadays, the use of Scrum and Kanban together is becoming quite popular, with many teams employing both frameworks in their projects. As Alqudah and Razali (2018) mention, it can be quite beneficial as it allows the team to 'adopt the appropriate practices of both methods based on different situations to meet their needs'. The authors emphasize that the team members need to understand which elements of either framework bring value to the project and adapt them accordingly.

2.1.3 A hybrid approach to SDLC

2.1.4 Conclusion

2.2 Requirement gathering techniques

2.3 Accesibility considerations

2.4 Tech stack

2.5 EHR systems

2.6 Machine Learning Summarization models

2.7 Speech Recognition models

References

Alqudah, Mashal and Rozilawati Razali (2018). "An empirical study of Scrumban formation based on the selection of scrum and Kanban practices". In: *Int. J. Adv. Sci. Eng. Inf. Technol* 8.6, pp. 2315–2322.

Alshamrani, Adel and Abdullah Bahattab (2015). "A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model". In: *International Journal of Computer Science Issues (IJCSI)* 12.1, p. 106.

BBC (2024). *Moldova country profile*. URL: https://www.bbc.co.uk/news/world-europe-17601580. (Accessed: 22 Oct 2024).

Ciurcă, Aliona (2021). *Sistemul informațional automatizat din spitale: cum funcționează în R. Moldova și ce cred medicii că ar trebui ajustat*. URL: https://www.zdg.md/stiri/stiri-sociale/sistemul-informational-automatizat-din-spitale-cum-functioneaza-in-r-moldova-si-ce-cred-medicii-ca-ar-trebui-ajustat/. (Accessed: 22 Oct 2024).

Cunningham, Ward (2001). *Manifesto for Agile Software Development*. URL: https://agilemanifesto.org/. (Accessed: 25 Oct 2024).

E-Governance Agency (n.d.[a]). *About EGA*. URL: https://egov.md/en/about-ega. (Accessed: 22 Oct 2024).

— (n.d.[b]). *Citizen's Government Portal*. URL: https://egov.md/en/content/citizens-government-portal. (Accessed: 22 Oct 2024).

Luo, Lu (2001). "Software testing techniques". In: *Institute for software research international Carnegie mellon university Pittsburgh, PA* 15232.1-19, p. 19.

National Bureau of Statistics of the Republic of Moldova (2024). *Population*. URL: https://statistica.gov.md/en/statistic_indicator_details/25. (Accessed: 22 Oct 2024).

Ruparelia, Nayan B. (May 2010). "Software development lifecycle models". In: *SIGSOFT Softw. Eng. Notes* 35.3, pp. 8–13. ISSN: 0163-5948. DOI: 10.1145/1764810.1764814. URL: https://doi.org/10.1145/1764810.1764814.

Sunner, Daminderjit (2016). "Agile: Adapting to need of the hour: Understanding Agile methodology and Agile techniques". In: pp. 130–135. DOI: 10.1109/ICATCCT.2016.7911978.

United Nations Development Programme (2023). *A 100% digital state: The strategy for digital transformation of the Republic of Moldova for 2023-2030, approved by the Executive*. URL: https://www.undp.org/moldova/press-releases/100-digital-state-strategy-digital-transformation-republic-moldova-2023-2030-approved-executive. (Accessed: 22 Oct 2024).

USMF (2023). *Brief history*. URL: https://usmf.md/en/brief-history-usmf. (Accessed: 22 Oct 2024).

Yas, Qahtan, Abdulbasit Alazzawi, and Bahbibi Rahmatullah (Nov. 2023). "A Comprehensive Review of Software Development Life Cycle methodologies: Pros, Cons, and Future Directions". In: *Iraqi Journal For Computer Science and Mathematics* 4.4, pp. 173–190. DOI: 10.52866/ijcsm.2023.04.04.014. URL: https://journal.esj.edu.iq/index.php/IJCM/article/view/664.