COMPUTER SCIENCE

DEVELOPMENT OF TELEMEDICINE PROCESS SUPPORT VISUAL TOOLKIT

Gagik Kirakossian, *Prof. Dr., Head of Department "Computer Systems and Networks"*
National Polytechnic University of Armenia, Yerevan, Armenia
Antranig Momjian, *PhD student at National Polytechnic University of Armenia, and Senior Software Engineer at Ogma Applications, Yerevan, Armenia*

DOI: https://doi.org/10.31435/rsglobal_conf/30032021/7479

Abstract. *The diagnosis and treatment processes of diseases, the effectiveness of different programming approaches in the automation of the treatment process were studied.*
A telemedicine process support visual toolkit has been suggested, which will allow the doctors to build the treatment schemes using simple blocks, depending on the patient's input data.
The visual blocks that are planned to have in the IDE can be divided to the following groups: dataFlow, arithmetic and logical operations blocks, Artificial intelligence blocks that will be implemented on the basis of various machine learning models, Blocks intended for read data and configure the devices of the IOT network, and custom blocks.
The toolkit will translate the represented treatment visual schemes to python code, which will call the corresponding functions of represented treatment regimen's blocks, which will be developed by us.
Keywords: *IoT, Artificial Intelligence, specialized visualization toolkit, accuracy of the results, IDE.*

Introduction. Chronic diseases have always been a burden for patients which require regular measurements, recording measurement data, visiting doctor regularly, showing the doctor the recorded measurement data, receiving updated treatment plan, and following instructions.

The main problem for this diseases are that they require to have visit the doctor and get care plan updates frequently.

The goal of the research is to create an automated diagnosis and treatment system that is able to monitor the patient in real time and modify/adjust the patient's care plan automatically based on their health state that can be detected by analyzing the data collected from the patient via IoT network, to ensure the best treatment quality. The generated care plan will be available to the patient trough mobile application. [1]

The automated diagnosis and treatment system has the following basic requirements:

- The designed system should have the ability to be easily updated with the latest medical innovations that can improve the quality of patients' treatment.
- The system must have high accurate.

The Problem. A system should be developed which will solve the above-mentioned problems as much as possible.

There are two approaches to software development: the traditional algorithm description approach and the artificial intelligence approach.

In the first case, an accurate software can be developed, but in order to keep the system up to date with the latest scientific innovations, constantly adding new software code will be necessary, which requires programming knowledge that doctors do not have; it's not feasible for this project.

With the help of AI methods a system can be developed that learns medical innovations in real time, but it is impossible to get 100% accuracy. In other words, its answers will be approximate, because the artificial intelligence is based on statistics and probability theory.

Moreover, by these methods not the algorithm of the patient's treatment are taught to the system, but clear cases of the disease with the appropriate care plans are taught to the system. Then, during patient's treatment process, the method of artificial intelligence matches the patient's case to one of the cases it has learned and gives the appropriate treatment plan for the case.

Solution. The best solution to this problem would be the creation of a specialized visualization toolkit for the medical field that would allow doctors to describe treatment schemes through comprehensible visual blocks and to update them in the future.

The structure of the visualization toolkit's blocks should be designed to be convenient, clear and comprehensible to doctors, with preserving the flexibility of language to express the most complex medical algorithms at the same time.

The visual scripts should be compiled then to Web API application that must run on application server to serve the web requests.

For simplicity and benefit from the existing libraries and frameworks, we will translate the visual script to an existing programming language. Thus we won't need to write compilers to compile the scripts to assembly or to write interpreters from scratch. Nor we will need to write web frameworks, database adapters or machine learning libraries.

The language to which the visual scripts will be translated should be easy to deploy new versions during the application runtime, without the need of "blue green deployment" approach. Because the system should be upgraded to newer version every time a doctor makes change to any visual script, and the hard code reload feature will be a must for the language chosen.

As we know hot code reload only works for the scripting languages, thus the language chosen will be scripting language.

From the wide range of the scripting languages, we should choose a one that has a reliable security and good web framework, having good machine learning libraries would be a plus.

It's clear that python would be the best for this case, with its various machine learning libraries including: pandas, Tensorflow, NumPy, etc. And god web development frameworks including Flask, Django, etc.

Each visual script block should have corresponding implementation in python. The toolkit will translate the represented treatment visual schemes to python code, which will call the corresponding functions of represented treatment regimen's blocks, which will be developed by us.

Figure 1 presents an example of diabetes monitoring and care plan adjustment generalized scheme. The real system schemes should be very detailed and complex.

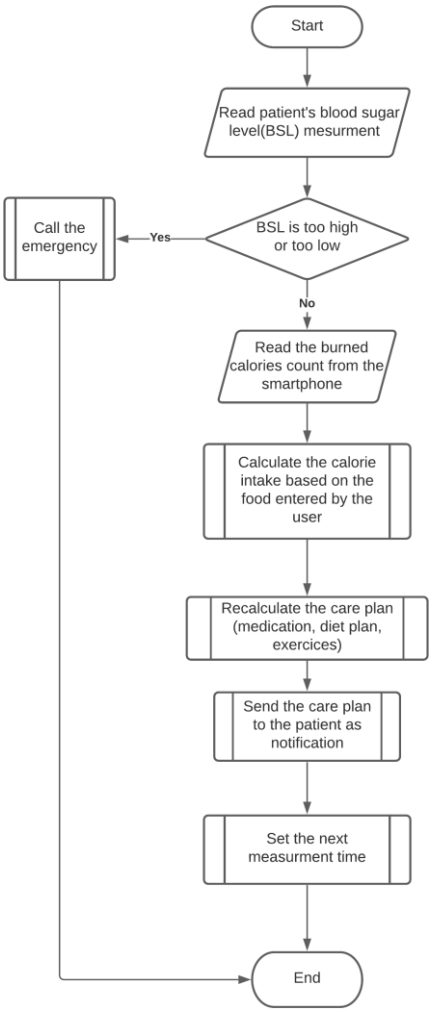

Fig. 1. Diabetes Generalized scheme

We can have two possible policies for visual schemes update permissions

1. Have a team of trusted doctors with that permission.
2. Allow different organizations to have their personal server instance, and each organization will have the right to update only its own instance's visual schemes.

The visual blocks that are planned to have in the IDE can be divided to the following groups:

1. DataFlow, that will include conditional commands, cycles and etc.
2. Arithmetic and logical operations blocks
3. Artificial intelligence blocks, that will be implemented on the basis of various machine learning models
4. Blocks intended for read data and configure the devices of the IOT network
5. Custom blocks.

DataFlow blocks:

1. If
2. If-else
3. For
4. While
5. ForEach

Arithmetic and logic blocks:

1. Add
2. Subtract
3. Multiply
4. Divide
5. And
6. Or
7. Not
8. XOR

Artificial intelligence blocks:

1. Blocks that will perform predefined tasks using pre-selected and trained models.
2. Blocks that will represent various machine learning models, that would be able to be trained through the visual toolkit, by passing the training set and parameters as input.

IOT blocks. Foreseen for receiving data from and configuring the devices that exist in the IoT network.

This blocks are the ones that will interact with the IoT devices that exist in the system. We will split them to two groups: devices' configuration blocks and data fetching blocks.

Devices configuration blocks are responsible to send configuration to the devices that will determine the frequency of the measurements that the device should perform, or the frequency of uploading the collected data to the server. Data fetching blocks are responsible for reading the measured data.

We have two type of devices in our system that require different workflows to collect the data from them. The first is the devices that upload their data to the device's manufacturer's server, for example Dexcom [2] that uploads the user data to its own servers and provides APIs to configure the frequency and the timing of the measurements [3]. In this case the devices' configuration blocks will configure the frequency and the timing of the measurements trough the API. And the data fetching blocks will get the measurements data from the Dexcom server trough the API.

The second type of the devices are those who sync their measurements with health applications: 'Samsung Health' for Android devices, and 'Health' for iOS, which fortunately have corresponding SDKs.

Because we want to write our application with the Xamarin framework we will use Healthkit for Xamarin.iOS [4] and MKM-HealthDataSDK [5, 6] for Xamarin.Android to get the health data from health applications.

For this type of devices the devices' configuration blocks will send to our application the frequency and the timing when it should read the health data through the SDKs and upload them to our server. And the data fetching blocks will fetch the data that is already uploaded to our server from the database.

Custom blocks. It will allow users to build new blocks, the functionality of which will be possible to describe by one of the methods mentioned below:

1. Visual toolkit scheme
2. Python code

The programs represented through visualizations toolkit represent structured data, which we will use for training our machine learning based visual toolkit intellisense. GPT-2 or TabNine models could be used as a model.

The model will help doctors to write effective scripts by hinting the blocks that could be used, cases to consider, etc.

Conclusions. The existing technologies were studied and appropriate programming languages, technologies, strategies, and network systems were selected to develop the telemedicine process support visual toolkit. The toolkit will include various types of already existing blocks and the ability to create new blocks, through which appropriate treatment schemes can be built. Depending on the patient's input data, treatment schemes can give different results. In other words, a toolkit with a simple interface has been developed, in which a logical sequence of blocks can be which will work in case of all possible input data. When running the schemes, each block will be converted to code: and they work as a complete software code.

REFERENCES

1. Momjyan A.J., Andreasyan L.K., Kirakossian G.T. The architecture design of type 2 diabetes prevention and treatment decision support smart system in the telecommunication network // Proceedings of Engineering Academy of Armenia. -2018. –Vol. 15, № 2. – P. 291-294
2. Dexcom // Continuous Glucose Monitoring // https://www.dexcom.com/
3. Dexcom // Dexcom Developer Portal // https://developer.dexcom.com/
4. Microsoft // HealthKit in Xamarin.iOS // https://docs.microsoft.com/en-us/xamarin/ios/platform/healthkit
5. Kirakossian G. T., Mayilyan A. K., Momjian A. J. // MKM-HealthDataSDK // https://www.nuget.org/packages/MKM-HealthDataSDK/
6. Kirakossian G. T., Mayilyan A. K., Momjian A. J. // MKM-Health Data software development kit for mobile applications // Journal of Environmental Science, Computer Science and Engineering & Technology. September 2019- November 2019. Vol. 8, № 4. P. 290-300.