Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 141 (2018) 279–286

Procedia
Computer Science

www.elsevier.com/locate/procedia

The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2018)

Parallel software architecture for the next generation of glucose monitoring

Tomas Koutny[a]*, Martin Ubl[b]

[a]*NTIS – New Technologies for Information Society, University of West Bohemia, Technicka 8, 306 14 Plzen, Czech Republic*
[b]*Department of Computer Science and Engineering, University of West Bohemia, Technicka 8, 306 14 Plzen, Czech Republic*

Abstract

Diabetes is a widespread disease. Elevated blood glucose levels continuously damage multiple organs in the long-term. In the short-term, hypo- and hyperglycemic shocks are acute risks. Diabetes patients monitor their glucose level using continuous glucose monitoring systems. Based on their measured glucose level, the patient take insulin to lower their blood glucose level. With the advances in mobile computing, an increasing number of diabetes patients engage in self-built systems. They read their glucose levels from glucose-monitoring systems and calculate their insulin dosage based on the measured levels. The self-built nature of such a system raises a number of medical and software engineering concerns. Therefore, we propose a software architecture for the next generation of glucose monitoring. The proposed architecture builds on the principles of the high-level architecture. We decompose the entire glucose monitoring system to basic elements, which are either real or simulated. This opens the proposed architecture to software engineering, simulation, and fault-tolerance research. As a proof of concept, we present an illustrative configuration of the implemented software architecture that predicts future blood glucose levels 15 minutes in advance for type-1 diabetes patients. All relative errors are in the A+B zones of Clarke and Parkes error grids, with almost 95% of errors in the safest A-zones of both grids.

Keywords: diabetes; gluocse; monitoring; software; prediction; simulation

* Corresponding author. Tel.: +420 377 632 437; fax: +420 377 632 402.
E-mail address: txkoutny@kiv.zcu.cz

1. Introduction

Diabetes is a heterogeneous group of diseases that share the common phenotype of elevated blood glucose level (BG) due to insulin insufficiency. It is a silent disease. It does not cause pain until it has developed. Type-1 diabetes (T1D) patient depends on artificial insulin dosing. Gradually, T1D may develop from the other types of diabetes.

In subcutaneous tissue, the presence of glucose triggers a chemical reaction. This reaction creates an electric current. Continuous glucose monitoring systems (CGMSs) measure this current and convert it to a glucose level (IG). Patients calibrate this conversion with BG. Patients measure BG sporadically as a result of associated pain and discomfort; this is done by drawing a drop of blood from a finger. Nevertheless, the CGMS reports IG continuously within a short period – typically five minutes. Then, an insulin pump administers insulin to subcutaneous tissue, which lowers BG.

Having the sporadically measured BG and continuously measured IG, software on the insulin pump calculates volume of the insulin dose. Traditionally, only large medical companies are capable of developing such a software – considering the research, development, testing, insurance and other associated costs. On the contrary, the growing availability and computational power of mobile smartphones has given rise to do-it-yourself projects. The Nightscout project is the first one [1, 2]. Originally, it was designed to help parents to monitor their children with T1D. The Nightscout software is attached to CGMS software to intercept data communication, and upload IG measurements to an alternative cloud. From that cloud, parents can download the levels and monitor condition of their children remotely. This project is an important lesson to medical bodies as patients can change the "rules of the game" with modern technology.

While the Nightscout project has focused on monitoring, the OpenAPS (open artificial pancreas) project goes further. It calculates the volume of the insulin dose. As insulin lowers BG, overdosing is a risk because the brain cannot synthetize nor store glucose for more than a few minutes worth of supply. The brain depends on a continuous supply of glucose from the blood [3].

Both projects raise several medical concerns [4]: outcomes are self-reported, self-built systems may differ and their users are likely more engaged in glucose controlling than average diabetic patient. In addition to the medical concerns, we must identify software engineering issues.

To the best of our knowledge, there is no publicly available research on CGMS software architecture. Therefore, we propose a new architecture that builds on the principles of distributed computer simulation systems such as the high-level architecture [5]. We break up the entire chain of glucose devices and the associated software programs to basic elements, which can be either simulated or real. We do not lose the original aims of Nightscout and OpenAPS, while considering energy consumption already at the architectural level, supporting unified and repeatable medical testing and leaving the project open to software engineering, simulation, and fault tolerance research.

Section 2 of this paper describes the proposed architecture, while the following section discusses its implementation. Fourth section gives an experimental setup of BG prediction. Fifth section concludes the paper.

2. Architecture

To optimize the delivery of healthcare, the software architecture must be identical for both physical devices and simulated devices. We need simulation to develop and evaluate new models and methods *in-silico* before conducting clinical trials. In addition, identical architectures provide identical error analysis, which is needed to compare studies for different authors fairly [6].

The high-level architecture (HLA) is a well-studied simulation paradigm [5, 7]. HLA simulation is comprised of a number of federates, which communicate using a predefined communication standard. Individual federates communicate with each other using a publish/subscribe (send/receive data) communication model. As the communication flows through HLA runtime infrastructure, i.e., an object other than a federate, a federate need not know of other federates; it only needs to know of particular data streams.

The HLA's complexity is not well suited for low power devices. Therefore, we reduced its complexity to a simplex, which is fall through architecture that retains the key benefits of the HLA architecture. As Figure 1 depicts, the devised architecture is comprised of a number of filters. A simplex pipe connects a pair of filters. A pipe exposes the two functions of send and receive, which accept a single message format. Its structure as follows:

```
struct TDevice_Event {
    NDevice_Event_Code event_code; //integral type (uint8_t) encoding particular event code
    GUID device_id;                //device that produced the event
    GUID signal_id;                //signal that the event is concerned with
    double device_time;            //fractional counter of days since January 0, 1900
    int64_t logical_time;          //Lamport clock counter
    uint64_t segment_id;           ///id of CGMS profile that the event is concerned with
    union {                        //specific information dependant on event_code
        double level;              //measured or calculated levels of glucose, insulin, etc.
        IModel_Parameter_Vector* parameters;  //pointer to reference-counted double[]
        wstr_container* information;          //pointer to reference-counted utf-16 string
    };
};
```

- event code – advertising of external events and control codes to control the devices
- logical and device time – Lamport and real-time timestamps
- device id – device that produced the event
- signal id – IG, BG, calculated BG, predicted BG, insulin, etc.
- level – measured or calculated level of a specific substance such as glucose, insulin, or carbohydrates
- parameters – advertising of current parameters of implemented models of glucose dynamics
- information – auxiliary information that is specific to a particular event code

Figure 1 depicts a sample simulation scenario. It is a sequence of six filters with the following properties:

- To control the entire filter line, the end user need not be aware of particular filters. Through the user interface, the user instructs the program to inject a specific message into the very first pipe. Then, the message flows through the entire filter line and the filters react accordingly. For example, the patient can enter addition, non-calibrating BG to improve the precision of calculated BG.
- Filter #1 receives measured IG and calibration BG from the CGMS.
- From the received IG and BG, Filter #2 determines personalized parameters for the extended diffusion model (version 2) [8] of glucose dynamics. In the depicted setup, the NewUOA method combined with the crosswalk metric is used [9]. Alternatively, it would be possible to use meta-differential evolution [10], honey bee mating optimization [11], or another method instead of the NewUOA. The solving filter determines the parameters asynchronously to the main flow. Thus, the third and subsequent filters continue to process the CGMS signal as before, until new parameters are determined. Once the parameters are generated, Filter #2 generates a respective message and sends it to the Filter #3.

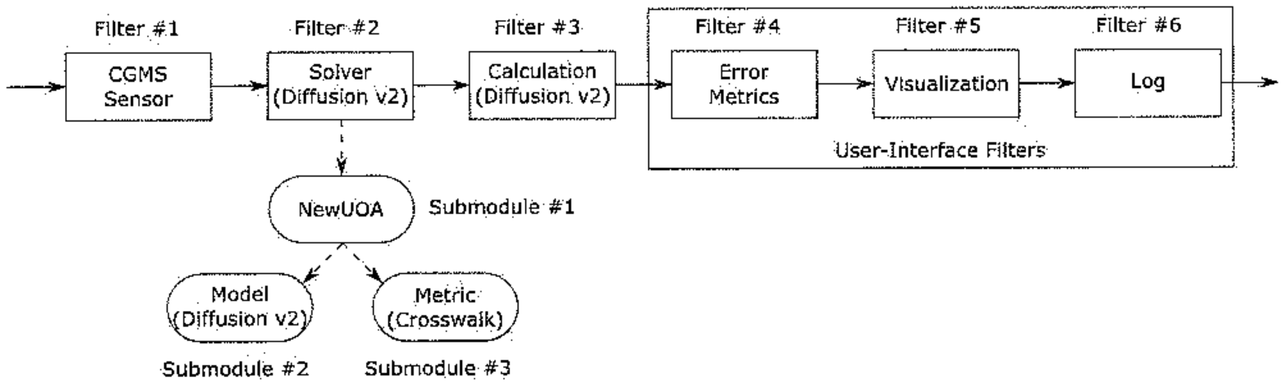

Fig. 1. Simplex fall through architecture

- Filter #3 calculates BG using the diffusion model (version 2). It updates its working model parameters on receiving new ones, typically determined by the solver filter. In addition, it is possible to store previously determined parameters and to initialize the calculation filter with them. Once loaded from non-volatile storage, such parameters are injected into the first pipe before all the filters are run. As filters forward messages, the parameters will pass through the CGMS filter to the solver filter. As NewUOA is a locally optimization method, the solver filter can use these parameters to construct an initial solution for NewUOA. Otherwise, it uses default parameters [10].
- Filters #4 – #6 are bound to the user interface to visualize relative errors of the sensor and BG and IG calculations according to the current glucose monitoring practice (medium average relative error, 95th percentile of relative error, error grid, daily plot of glucose levels, ambulatory glucose profile, etc.). Eventually, Filter #6 records the events to playback the entire simulation later.

In a more advanced scenario, it is possible to include a filter that calculates the optimal dose of insulin. Then, another filter could transmit respective commands to a specific insulin pump. With the *in-silico* trial-scenario, we would use a filter that reads glucose profile from a database instead of the CGMS filter. Alternatively, yet another filter could connect to a diabetes patient simulator.

To achieve a high degree of variability, when configuring the filter-line, each filter is a standalone module with a globally unique id (GUID [12]). The GUID identifies biomedical signals, models, solver sub-modules, and solver metrics. Combining GUID-based identification with dynamic loading of modules, third-party developers can design and deploy any GUID-identified component without any collision with other developers.

To support model composition, we implement a filter that remaps the signal GUID to another. As the message flows through the filters, this remapping filter rewrites the signal GUID in that message. Then, the output of one model of glucose dynamics is available as input for another model, which does not need to be aware of the preceding model due to the remapping.

Parallelism increases filter line throughput by exploiting the hardware concurrency of modern processors. Each filter executes in a standalone thread and the filter-connecting pipe acts as a blocking synchronization primitive. Such a design pattern prevents overgeneration of messages so that the system is stable. For the design, we utilized the finding of study [13]. This study concerned an event-based communication scheme for the parallelization of glucose level prediction.

3. Implementation

We targeted four devices:

1. EMT64 desktop/server processor for running the simulation.
2. Independent low-power device with an ARM processor, which acts as the CGMS.
3. Smartphone that reads data from the CGMS device.
4. Smartwatch that reads data from the smartphone.

3.1. EMT64 Desktop/Server Processor

The desktop/server processor runs the simulation, which benefit from time compression. Typically, it replays a number of glucose profiles. With such a processor, we perform a statistically significant number of calculations. This allows us to experiment with new models and algorithms as the processor provides enough computational power. Specifically, we used an Intel i7-7700K-based system.

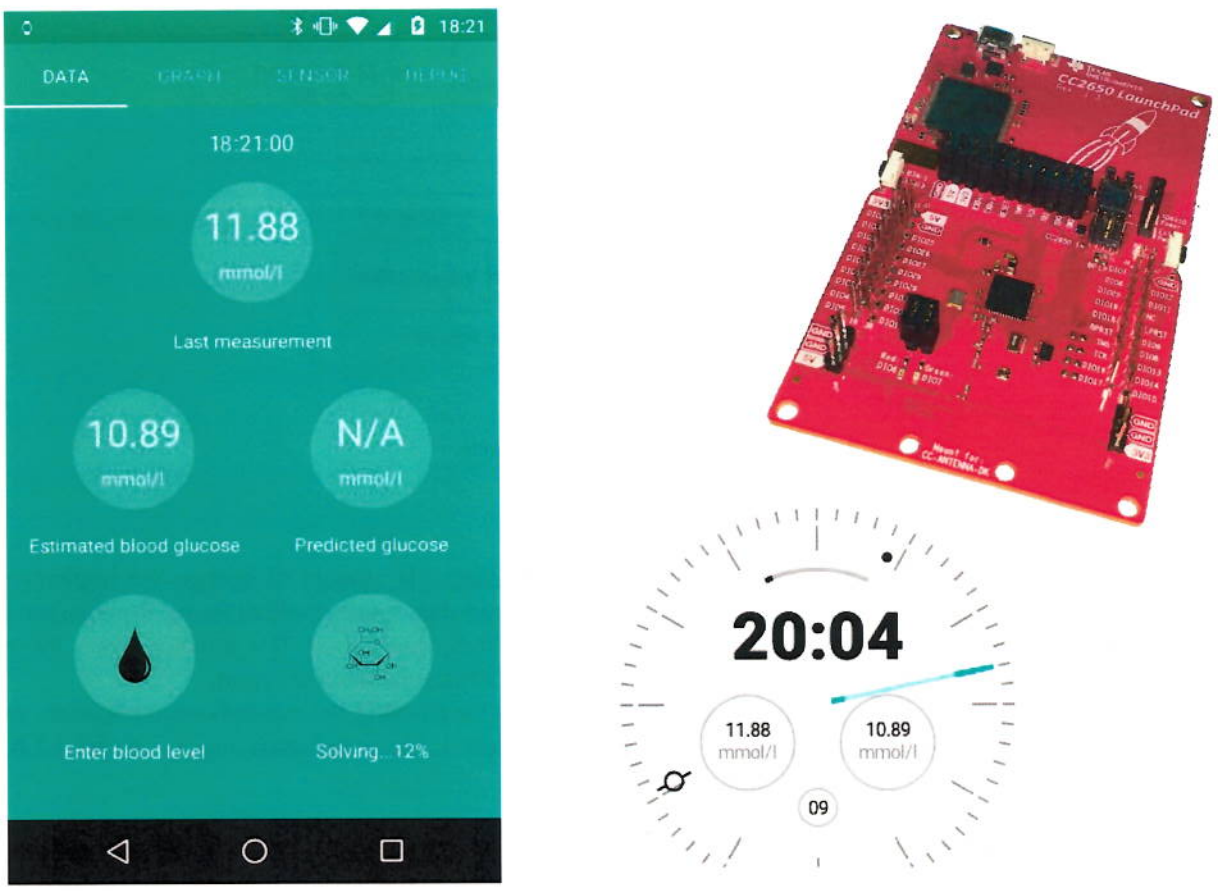

Fig. 2. Connecting the smartphone with the CGMS and connecting the smartwatch with the smartphone

To visualize glucose calculations, we used the same analytical widgets used by the diabetes.zcu.cz project [14]. The respective website offers online continuous calculation of BG from a CGMS profile. Glucose levels are visualized and analyzed using graphs, daily plots, Clarke and Parkes' error grids, ambulatory glucose profile, and the empirical cumulative distribution function of the relative error of calculated BG.

3.2. Low-power CGMS

The modern CGMSs comply with the IEEE 11073 standard. It is specialized for continuous glucose monitoring and communicates through the Bluetooth Low Energy standard. Using Texas Instruments CC2650 LaunchPad board with the TI-RTOS operating system, we implemented a hardware CGMS emulator. It replays previously recorded glucose profiles, while synchronizing them with a real-time clock.

3.3. Smartphone

The smartphone is the de facto ubiquitous mobile device of today. It connects to other devices using the Bluetooth standard. Hence, it can connect to a CGMS using the Low Power Bluetooth standard. Processors used in today's smartphones are advanced enough to run sophisticated calculations, despite being limited by battery capacity. Therefore, we targeted a smartphone as the candidate to complement and possibly replace the traditional CGMS receiver. Specifically, we used the Cortex-A53-based smartphone Motorola Moto G Gen3.

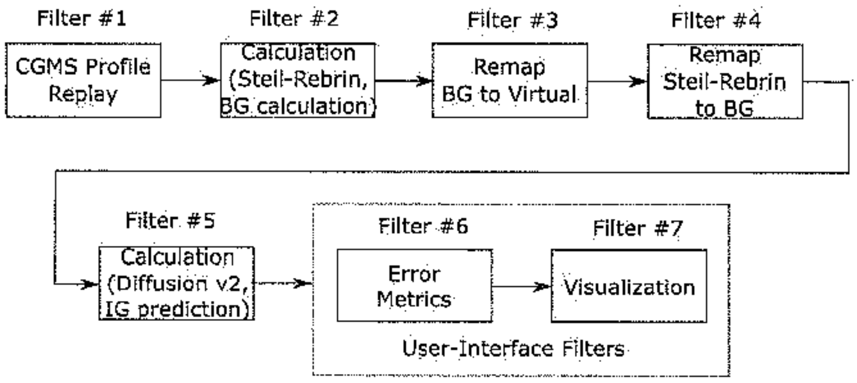

Fig. 3. Predicting glucose levels

3.4. Smartwatch

As illuminating the smartphone's display may discharge a considerable amount of energy, we implemented a glucose monitor application for a smartwatch. Such a watch displays two data streams, which the smartphone produces. The smartwatch updates the display on a change of data stream, which occurs every few minutes. Thus, the power drain is low. Specifically, we used the Cortex-A7-based smartwatch Motorola Moto360 sport.

Figure 2 depicts the board (which provides hardware emulation of the CGMS), the smartphone application, and the smartwatch screen. In accordance with the scenario depicted in Figure 1, the smartwatch displays measured IG and calculated BG.

All targeted devices share a common source code base, written in modern C++17. We chose C++ to minimize memory requirements, maximize computational performance, and reduce energy drain to target low power devices [15]. Nevertheless, we recognize users' preferences to motivate them to use the software. Therefore, although the smartphone application shares the C++ code base, it implements the user-interface of the Xamarin technology to retain the platform's look and feel.

4. Experimental Setup

To demonstrate the capabilities of the proposed software architecture, we combined two models of glucose dynamics to predict BG. BG prediction is difficult due to the number of required input signals (continuous BG and IG). Therefore, we calculate the required continuous BG signal with a filter that we connect before the prediction filter to supply all required signals.

Figure 3 depicts respective filter configuration. Filter #1 replays a glucose profile recorded from an adult T1D patient. Filter #2 calculates continuous BG signal using the Steil-Rebrin model with a sensorerror model [16]. Then, the Filter #3 remaps the measured BG as a virtual signal so that Filter #4 can remap the Steil-Rebrin calculated BG as the measured BG signal. Therefore, Filter #5 has continuous IG and BG signals on its input. As a result, it predicts future IG. Finally, Filters #6 and #7 calculate error metrics and visualize the results, respectively. To interpolate glucose levels, we used the Akima spline.

For the sake of brevity of this example, we omitted solvers in this particular experimental setup. We pre-calculated the Steil-Rebrin parameters and shifted the calculated BG 15 minutes into the future. We used the diffusion model (version 2) to realize the time shift along a simple linear regression to reduce the prediction error. Tables 1 and 2 give the parameters. While the proposed architecture supports determining the parameters on-the-fly, as depicted in Figure 1, an adaptive multi-model prediction scheme would become too complex and fall out of scope of this paper.

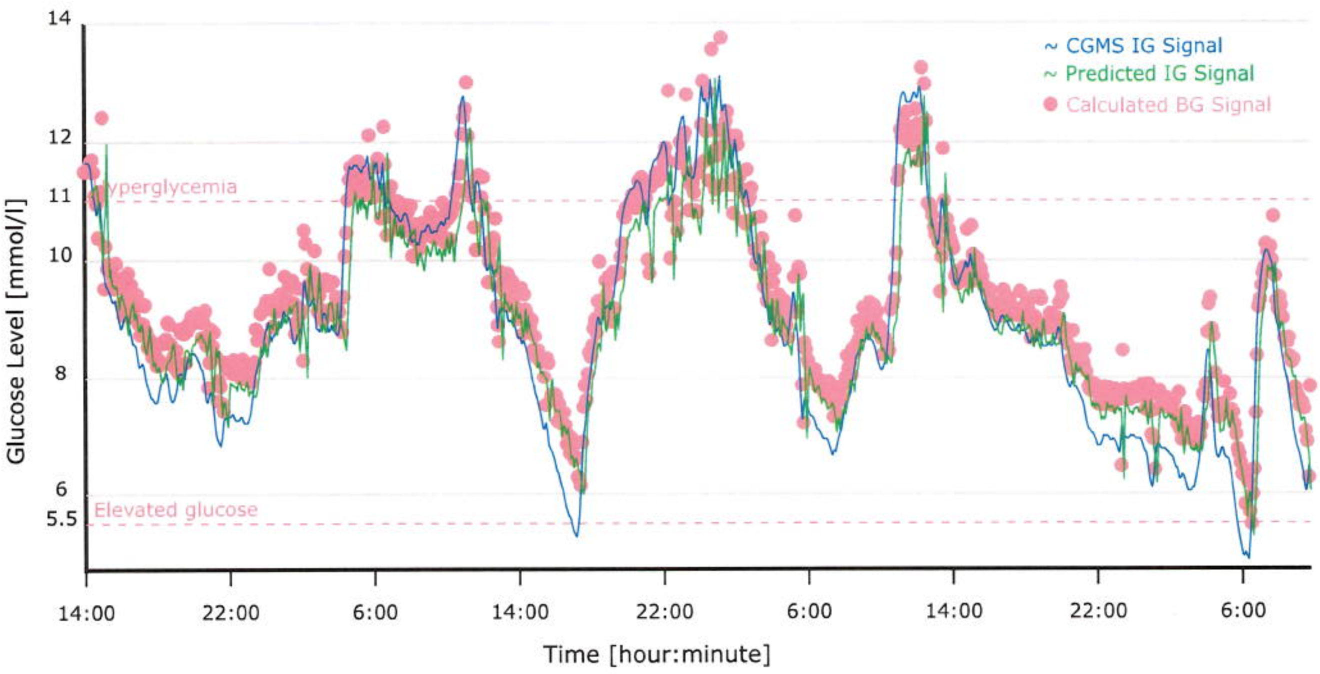

Fig. 4. Measured and calculated signals

Figure 4 depicts the measured and calculated signals. Table 3 gives an overview of the prediction error using both absolute and relative error. Relative error is the absolute difference between measured and calculated glucose level, divided by the measured level. It is the standard error measure used in diabetes treatment [5].

Table 1. The diffusion model (version 2) parameters

Parameter [unit]	p (unitless)	cg [L/mmol]	c [mmol/l]	Δt [min:sec]	k [s^2×L/mmol]	h [min:sec]
Value	0.964	0.000	-0.001	15:00	0.000	00:.00

Table 2. The Steil-Rebrin model parameters with sensor error model parameters

Parameter [unit]	τ [s×mmol/L]	g (unitless)	α (unitless)	β [mmol/l]	γ [mmol/(L×s)]
Value	-619.308	1.000	1.307	-3.376	0.000

Table 3. IG signal prediction absolute and relative errors

	Average	Std. Dev.	Minimum	1st Quartile	Median	3rd Quartile	95% Quantile	99% Quantile	Maximum
Absolute [mmol/]	0.657	0.538	<0.001	0.269	0.551	0.886	1.649	2.770	3.258
Relative [%]	7.669	6.500	0.005	2.924	6.285	10.290	21.200	31.080	38.310

In clinical practice, Clarke and Parkes error grids are the gold standard for evaluating the precision of glucometers. These error grids define zones A-E, which present increased risk to patient with respect to measurement errors. With the achieved maximum relative error, almost 95% of errors fall into the safest zone (A) of each grid. The remainder of the errors fall into the B zones of both grids. Zones A and B are both safe zones, which do not lead to inappropriate treatment. Zone B contains greater relative errors than zone A.

5. Conclusion

Using the proposed software architecture, we have demonstrated an intriguing prediction capability. It predicted BG 15 minutes ahead of time with small error and no need for additional information such as carbohydrate counting or insulin dosage. The proposed architecture allowed us to build increasingly advanced filter configurations and develop new filters to prolong the prediction window. We intend to release the software at diabetes.zcu.cz [14].

Given the architectural flexibility, it is also possible to connect an implementation of the proposed architecture with diabetic-patient simulators such as DMMS.R [17], University of Cambridge T1D Simulator [18], and Physiomodel [19]. Such a connection will require custom terminal filters to receive patient simulator data and send the calculated output back to the patient simulator.

Acknowledgements

This publication was supported by (1) project LO1506 of the Czech Ministry of Education, Youth and Sports and (2) Institutional support for long-term strategic development of research organizations.

The Diabetology center, 1st Dept. of Internal Medicine, University Hospital Pilsen, Czech Republic provided us with the anonymized T1D glucose profile.

References

[1] Omer T. Empowered citizen 'health hackers' who are not waiting. *BMC Medicine* 2106:14:118-120.
[2] Lee JM, Hirschfeld E, Wedding J. A patient do-it-yourself mobile technology system for a diabetes – Promise and challenges for a new era in medicine. *Journal of the American Medical Association* 2016:**315**:1447-1448.
[3] Poretsky L. *Principles of diabetes mellitus*, New York: Springer; 2010.
[4] Lewis D, Leibrand S, #OpenAPS community. Real-world use of open source artificial pancreas systems. *Journal of Diabetes Science and Technology* 2016:10:1411-1411.
[5] Topcu O, Durak U, Oguztuzun Hm, Yilmaz L. High level architecture. *Distributed Simulation. Simulation Foundations, Methods and Applications*. Cham:Springer 2106.
[6] Koutny T. Modelling of glucose dynamics for diabetes. 5th International work conference on bioinformatics and biomedical engineering 2017, Granada, Spain.
[7] Beaumont J, Mokhov A, Sokolov D, Yakovlev A. High-level asynchronous concepts at the interface between analog and digital worlds. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 2018:**37**:61-74.
[8] Koutny T. Blood glucose level reconstruction as a function of transcapillary glucose transport. *Computers in Biology and Medicine* 2014:**53**:171-178.
[9] Koutny T. Crosswalk – a time ordered metric. *Joint Conference of the European Medical and Biological Engineering Conference (EMBEC) and the Nordic-Baltic Conference on Biomedical Engineering and Medical Physics (NBC)* 2017, Tampere, Finland.
[10] Koutny T. Using meta-differential evolution to enhance a calculation of a continuous blood glucose level. *Computers in Biology and Medicine* 2016:**13**:45-54.
[11] Strnadek J, Koutny T, Kohout J. Introducing the effect of aging into the Honey Bee Mating Optimization to determine parameters of blood glucose level calculation. *9th International Conference on Human System Interaction* 2016 Portsmouth, United Kingdom.
[12] Leach P, Mealling M, Salz R. A universally unique identifier (UUID) URN namespace. *RFC 4122*.
[13] Koutny T. Experience with Lamport Clock ordered events with Intel Threading Building Blocks in a glucose-level prediction software. *International Work-Conference on Bioinformatics and Biomedical Engineering* 2014, Granada, Spain.
[14] Koutny T, Krcma M, Kohout J, Jezek P, Varnuskova J, Vcelak P, Strnadek J. On-line blood glucose level calculation. *Procedia Computer Science* 2016:**98**:228-235.
[15] Koutny T, Siroky D. Analyzing energy requirements of meta-differential evolution for future wearable medical devices. *World Congress on Medical Physics and Biomedical Engineering* 2018 Prague, Czech Republic.
[16] Del Favero S, Facchinetti A, Sparacino G, Cobelli C. Improving accuracy and precision of glucose sensor profiles: retrospective fitting by constrained deconvolution. *IEEE Transactions on Biomedical Engineering* 2014:**61**:1044-1053.
[17] The Epsilon Group. DMMS.R. *https://tegvirginia.com/* Last accessed on June 25, 2018.
[18] Wilinska ME, Chassin LJ, Acerini CL, Allen JM, Dunger DB, Hovorka R. Simulation environment to evaluate closed-loop insulin delivery systems in type 1 diabetes. *Journal of Diabetes Science and Technology* 2010:**4**: 132-144.
[19] Matejak M, Kofranek J. Physiomodel – an integrative physiology in Modelica. *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* 2015, Milan, Italy.