# Safe Implementation of Embedded Software for a Portable Device Supporting Drug Administration

Alena Simalatsar[1], Romain Bornet[2], Wenqi You[3], Yann Thoma[2], and Giovanni De Micheli[1]
Email: *alena.simalatsar@epfl.ch*, *romain.bornet@heig-vd.ch*, *you.wenqi@gmail.com*,
*yann.thoma@heig-vd.ch* and *giovanni.demicheli@epfl.ch*
[1] Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[2] Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (HEIG-VD), Yverdon-les-Bains, Switzerland
[3] Sophia Genetics, SA, Lausanne, Switzerland

***Abstract*—Poor adherence to medical regimen causes approximately 33% to 69% of medication-related hospitalizations and accounts for $100 billion in annual health care costs. In this paper we address the problem of unintentional non adherence, when patient fails to take a medication due to forgetfulness or carelessness. We present the safe approach to software implementation of a portable reminder device with enabled personalization of medical regimen. The presented prototype is designed for *imatinib* administration, a drug used to treat Chronic Myeloid Leukemia (CML). However, thanks to the component-based structure of the software, the method can be applied to other cases by replacing implementation of certain components.**

## I. Introduction

Statistically, over 300 million people worldwide suffer from diabetes, and thousands have diagnosed cancer of various types. Those are examples of chronic diseases that may last for years. The treatment of such diseases requires the adherence to the medication regimen, with proper dosages and administration scheduling, to maintain the plasma drug concentration within specific therapeutic ranges, which for many drugs, such as anticancer ones, are very narrow. According to [1] poor adherence causes approximately 33% to 69% of medication-related hospitalizations and accounts for $100 billion in annual health care costs. Therefore, often medical treatments are performed under close control of medical personnel that also performs occasional tests needed to adjust the treatment in personalized manner to achieve better results. Usually, the a priori personalization of medication is achieved by using Pharmacokinetic (PK) models describing the patient plasma drug concentration changes in time based on patient specific parameters and administered dosages. Later, in the *a posteriori* phase, the PK model parameters can be personalized [2] even more by performing occasional in-plasma drug concentration measurements called Therapeutic Drug Monitoring (TDM). The requirement to the adherence to the medication regimen with a posteriori personalization of the treatment demands that patients spend a significant amount of time in hospitals, which essentially lowers their life quality.

The introduction of a portable device, which could remind the patient when to take a drug dosage, perform the occasional TDM measurements, and adjust the treatment based on patient adherence to the medication regimen and the TDM measurements would be beneficial to many chronic patients. It would add more freedom to their lives, and reduce the treatment cost by only reducing the number of hours they spend in the hospitals. The requirements to such devices, predominantly controlled by software, are high. Even though they can run the same PK algorithms used by medical doctors to prescribe the dose, they must play a role of a doctor in validating the decision. Automatization of such behaviours may reduce the number of human factor errors, however, often with the price of introducing new types of software design errors, that may cause system failures or provide unpredictable behaviours. For example, the device may fail to provide an on-time reminder, due to being busy with processing other tasks. In embedded system design, such systems are known to be safety-critical. One of the approaches to the development of safety-critical software is to use formal methods, theorem proving, verification and/or correct-by-construction software design methodologies, which allow a high level representation of the software architecture that can be proven operating correctly against predefined properties. Some of the techniques provide code synthesis with the guaranty that the code preserves the properties of the high level model.

In this paper we present the implementation of a portable device prototype addressing the problem of unintentional non adherence to the medication regimen of *imatinib* [3] drug used to treat Chronic Myeloid Leukemia (CML). The prototype is implemented as a control software running on the ultra-low power processor *Icycom* [4]. At the abstract level the software is modeled with Timed Automata extended with Tasks (TAT) model implemented in TIMES tool [5]. The software is developed as a closed-loop control flow activating the tasks of TDM measurements, recording of the patient adherence to medication, PK modeling personalization augmented with the decision support system for medication regimen adjustment and validation of the regimen against medical guidelines (GLs) for drug administration, and, if necessary, generation of alarms sent to the patient and/or to the hospital. It is clear that the final device should be able to perform TDM measurements to adjust the medication regimen, which requires a real embedded biosensor. However, by now such sensors are not yet embeddable and no human experiments have been conducted. Instead, in order to validate the software, we used a data generator able to mimic the drug concentration evolution of a real human. The scheduling real-time properties of the implemented control flow where verified within the TIMES tool. We have also developed a new code synthesizer that allows the automatic generation of the final code to be deployed on the *Icycom* platform.

The paper is organized as follows. In Section II we present the heterogeneous space of the related and our prior work showing how various methodologies and algorithms linked to address the problem. Section III presents the Parametrized SVM algorithm, as the core algorithm for medication regimen adjustment. The control flow of the software implemented on our prototype device as well as the newly developed code synthesized are presented in Section IV. The actual prototype implementation is described in Section V. In Section VI we draw the conclusion over the presented work.

## II. Related and essential prior work

Adherence to the medication regimen is usually computed in relative percentage of times when the drug was taken within the prescribed period vs the overall drug administration events. The poor adherence causes a very high percentage of medication-related hospitalizations [1]. Nonadherence can be classified as unintentional, when patient fails to take a medication due to forgetfulness or carelessness, and intentional, when the medication is avoided by some rational decision making process based on perceptions, feelings and beliefs. Dealing with intentional nonadherence has more of a social flavour. In contrast, various technologies may help improving medication adherence if it was not intended.

In [6] authors give a comprehensive overview of medication adherence methods focusing mostly on smartphone applications. They have evaluated 160 smartphone apps all including a large list of options, with the main purpose of reminding the patient to take a certain drug at specified periods. Among the options they list: *online data entry*, *complex medication instructions*, *database of medications*, *tracking missed and taken doses* and many others. However, generally these applications, in one way or another, send reminders following the predefined static medication plans. Only one of the applications is reported to have a feedback loop with a bio degradable sensor swallowed together with each pill, which can automatically give a precise information about when and whether the pill was taken [7]. The rest of the applications rely on patient responsibility to provide such feedback. Nevertheless, the option of tracking missed or delayed doses is used only for computing the level of medication adherence. None of the reported applications is able to provide a personalized drug administration plan based on the misses or delays of drug administration. Moreover, none of them is able to adjust the medication plan based on occasional blood tests.

With our device we are targeting the unintentional nonadherence. The reminders are given based on the personalized medication plan, where the plan is adjusted by the algorithms. The adjustment can take into account both the misses or delays of drug administration as well as seldom TDM measurements. The safety-assured implementation of the software for such drug administration devices requires the use of specific software design techniques as well as several algorithms. Therefore, the related and essential prior works are covering a heterogeneous space of existing methodologies and algorithms, including: (A) design methodologies for safety-critical systems; (B) closed-loop drug administration; (C) modeling of drug concentration curves; (D) decision-support systems for dose adjustment; and (E) formal representation of medical Guidelines (GLs). Figure 1 depicts the abstract representation of the control flow of the prototype device. We use this figure to show the interrelation of the above mentioned technique as parts of heterogeneous space of the related works.

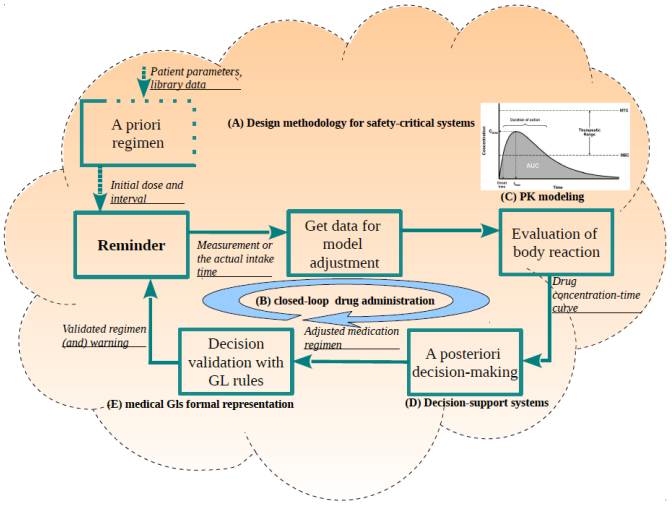


Figure 1: Interrelation of the techniques

### A. Design methodologies

There exist several system-level design methodologies helping to structure, plan, and control the process of developing an information system [8]–[10]. In this section we will focus on a frameworks allowing formal analysis of system properties essential for the design of safety-critical systems.

BIP (Behaviour - Interaction - Priority) [9] is a framework that relies on correct-by-construction methodology that allows modelling of systems as composition of atomic components. The behaviour of each atomic component is described as 1-safe priority Petri-net, while components are interacting by means of connectors [11]. Priorities are used to eliminate conflicts between interactions and thus restrict non-determinism. BIP provides verification means [12] and code synthesis techniques for distributed architectures [13]. However, the use of BIP for our case would be excessive, since the whole controlling software can be represented with one BIP component, while the code synthesis needs to be done for a specific platform.

*Timed Automaton* (TA) [14] is a formal model of computation used in embedded system design domain to describe a system behaviour and its progress in time. TA is an extension of the classical Automaton that is a finite state graph composed of the finite set of locations $Loc$ and transition relations (edges) $\hookrightarrow$. TA extends the classical Automaton with the finite set of clocks, later in the text also called timers, $C$ and a set of constraints over clocks $ClockCons(C)$, where constraints are conjunctions, disjunctions and negations of atomic expressions over clocks in the form $x \bowtie n, x \in C, n \in \mathbb{N}_0$ for $\bowtie \in \{<, \leq, >, \geq, =\}$. Each location is characterized by an *invariant* ($I$) that specifies a constraint on a clock under which TA can stay in this location and/or enforce a transition to another location. An edge of TA $e = (l, g, a, r, l') \subseteq \hookrightarrow$ represents a transition from $l$ to $l'$ ($l, l' \subseteq Loc$), where $g$ is a guard of $e$, which indicates when the transition can be executed, $r$ is the set of clocks that

is reset when the edge is taken, and $a$ is the action of $e$, $a \subseteq Act$. TAT [5] is an extension of TA with tasks that represent pieces of code associated with locations of the model. The execution semantics of TAT is the one of TA extended with a task queue. Any time the task is triggered by a transition it is added to the task queue, after which it will be executed upon a chosen scheduling policy. The timed model checker UPPAAL [15], implements TA extended with variables. A set of cooperating TA is called a network of TA. The cooperation mechanism may either make use of shared (global) variables or be realized as joint execution of dedicated transitions, denoted as rendezvous synchronization. TIMES tool [16] is the successor of UPPAAL implementing TAT model.

For our purposes we employ a TAT model. The main advantage of TAT is that it is supported by TIMES model checking environments that allow automatic schedulability analysis of the models it implements. In our prior works we also employed TAT model for formal representation of medical GLs (see Section II-E). Therefore, when using TAT, we can model the control flow activating required computational tasks, combine this control flow with the formal model of the GLs and verify that all the essential deadlines will be met. The original version of TIMES also provides platform independent code synthesis from high level TAT models. In this paper we present our extension of the TIMES code synthesiser (see Section IV-B) for automatic code generation for the *Icycom* platform (see Section V).

### B. Closed-loop drug administration

The continuous medication regimen adjustment requires the introduction of a feedback loop to provide the parameters needed to calibrate the system. Those can be measurements of specific biomarkers, the actual drug concentration and/or adherence to the medication regimen. An example of drug delivery control based on measuring specific biomarkers is presented in [17], where the authors introduce a safety-assured approach to the development of a Generic Patient Controlled Analgesic (GPCA) infusion pump using TA model. The feedback loop is based on measuring such indirect marker as $SpO_2$ concentration in patient's blood to evaluate oxygenation. The low level of $SpO_2$ indicates respiratory insufficiency and thus the drug delivery must be stopped. However, usually for many drugs used to treat chronic cases, including the case of Chronic Myeloid Leukemia (CML), it is not required to perform continues measurement of vital parameters.

This way, in the absence of real measurements of biomarkers or drug concentration the dose adjustment can be based on offline models (see Section II-C) able to compute the drug concentration-to-time (DCT) curve while accounting for how neatly the patient was following the schedule. For instance, if the patient has taken the drug with a delay, the model will foresee the effect of such mistaken action into the change in drug concentration. Nevertheless, occasional measurements of plasma drug concentrations is essential for personalized dose adjustment. Therefore, we would also like to be able to calibrate such models once a valuable real TDM measurement is available.

### C. Drug concentration modelling

There have been several models developed in support of Pharmacokinetic (PK) studies that are able to predict the drug concentration in the blood [18] and account for new measurements [2], [19]. Several personalized drug concentration prediction method based on *Support Vector Machine* (SVM) algorithm where presented in our prior works [20]–[22]. The initial method was only able to perform a point-wise drug concentration prediction, therefore, it is impossible to calibrate in personalized manner the prediction every time when a new measured concentration value is available for the patient under treatment. The Drug Concentration over Time (DCT) curve prediction approach, Parameterized SVM [23], used in our implementation, combines the SVM and analytical models. This method allows to introduce a mixed approach, in which we unify an offline SVM-based drug concentration prediction model, adherence to the medication regimen and occasional direct TDM measurements of the real drug concentration in the blood. The details of the algorithm are presented in Section III.

### D. Decision-support Systems in Drug Administration

For the closed loop drug delivery we need to solve the problem that is inverse to the PK modeling, e.g. providing the recommendation of the dose and delivery rate based on the drug concentration. In our prior work [21] we have introduced a Drug Administration Decision Support System (DADSS) to help clinicians/patients with the initial dose computing. The system is based on a Support Vector Machine (SVM) algorithm (see Section II-C) for estimation of the potential drug concentration in the blood of a patient, from which a best combination of dose and dose interval is selected at the level of a DSS. In the current implementation we are employing the above mentioned approach to solve the reverse problem of computing the right dose and administration interval to reach the target concentration.

### E. Formal representation of medical GLs

Once the dose is computed we need to make sure that it is effective and not harmful. This can be done by validating the output of the DADSS system with a process mimicking the decision of a medical doctor, which follows a medical GL. Medical GLs for drug administration usually contain the medication regimen, dosages and scheduling, for classes of patients and rules, based on which the medication regimen may be changed. Therefore these GLs can be seen as the specifications of an independent medical system able to take decisions regarding the changes of medical regimen. The automatization of such decision-making requires the computer-interpretable GLs representation. In our prior works [24], [25] we have presented an example of medical GL representation applying the TAT formal model. As was defined in Section II-A, TAT allows modeling of a system by composing it out of several TAT models combined into a network of interoperating automata. This allows a step-by-step detalization of the system control flow. For simplicity reason, in the control flow presented later we only use the rules limiting the dosage and delivery interval decided based on drug concentration models (see Section IV).

## III. Adjustment of the Medication Regimen

The adjustment of the medication regimen of our prototype device is performed based on the Parameterized SVM (ParaSVM) model able to predict the plasma drug concentration in the next cycle also accounting for the adherence to the medication regimen, defined as the time of drug intake $t_{intake}$, and occasional TDM measurements. In cycles when no TDM value is available, the time when the medication was taken is given to the model to compute the drug concentration value at that time. The output is the residual drug concentration that needs to be accounted in the next cycle, while the time of the next cycle trough value needs to be computed accordingly. This way, both adherence to the medication and TDM measurements are represented as plasma drug concentration value over time, used to adjust the model at each administration cycle.

As the input, the ParaSVM model requires a population library of the plasma drug concentration samples, where each sample is the concentration value in correspondence with the time of measurements and patient parameters (e.g. weight, age, gender, etc.) for whom this measurement was performed. To build the analytical representation of the DCT curve, ParaSVM uses the common basis functions $\beta^j = \{t^{-2}, \log(t), 1-e^{-t}\}$, respecting the shape of DCT curve obtained from the PK method [2], where $t$ stands for time [22]. Therefore, the target is to obtain the parameters $y$ for the weights of $\beta$:

$$f_{concentration} = y \cdot \beta = \begin{bmatrix} y^1 & y^2 & y^3 \end{bmatrix} \begin{bmatrix} \beta^1 \\ \beta^2 \\ \beta^3 \end{bmatrix}. \tag{1}$$

Therefore, parameters $\{y^1, y^2, y^3\}$ together with patients features form the Parameter Library being used as the training data. Unlike in [20] and [26] where the SVM algorithm was used to predict the drug concentration values based on patient parameters, here the SVM algorithm is applied to learn the mathematical relationship between the parameters of the basis functions and then to predict the parameter values of the DCT curve for a new patient in the testing dataset. With one given measured concentration value, in case of *a posteriori* adaptation, or a value of the residual concentration after the previous dose intake, the curve parameters can be adjusted, this way allowing to build the personalized DCT curve for each next cycle.

In the general case of modelling $N$ patient samples, the form of patient samples becomes $(x_i, y_i^1, \cdots, y_i^j, \cdots, y_i^{NP})$, where $i$ is the ID of a sample $i \in \{1, 2 \cdots, N\}$, $y_i^j$ denotes the $j$-th parameter value of this patient, and $NP$ is the number of parameters, which in our case is equal to three. The goal is to find $NP$ linear functions $f^j(x) = w^j \cdot \phi^j(x) + b_j$ to describe the relationship between the dataset points and estimate the parameter value $y$ according to a new input dataset. For that we need to minimize the following modified objective function:

$$\min_{w,b} \frac{1}{2}||w||^2 + \underbrace{C_0 \sum_{j=1}^{N_P} \sum_{i=1}^{N} [y_i^j - w^j \cdot \phi^j(x_i) - b^j]^2}_{H}, \tag{2}$$

where $H$ takes into account the combined difference of all three predicted values plus the ones in the parameter library. Note that this objective function has Root of Sum of Square (RSS) fitting error and a regularization term, which is also a standard procedure for the training of Multi Layer Perceptrons (MLP) and is related to ridge regression [27], [28]. Applying Lagrangian analysis to solve the optimization problem of objective function, we obtain $w$ as:

$$w^j = \sum_{i=1}^{N} \alpha_i^j \phi^j(x_i). \tag{3}$$

Combining Equ. (2) and (3), we can obtain a linear system:

$$\begin{bmatrix} \mathbf{K}^j + \frac{1}{C_0} I & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} \alpha^j \\ b^j \end{bmatrix} = \begin{bmatrix} y^j \\ 0 \end{bmatrix}, \tag{4}$$

where each entry of the kernel matrix $\mathbf{K}^j$ is defined to be $K_{ab}^j = \phi^j(x_a)^T \phi^j(x_b)$. A Gaussian Kernel is applied in a similar way as in [20]. Therefore, the prediction function becomes: $f^j(x) = \sum_{i=1}^{N} \alpha_i \mathbf{K}^j(x_i, x) + b^j$.

For each next drug administration cycle we refine the DCT curve computed by ParaSVM with either the real measured concentration value or the residual concentration value from the previous cycle computed with the same model taking into account the time of the drug intake $t_{intake}$. For model adaptation we use the following constraints:

- The modified DCT curve has to pass through the given measured concentration value;
- After a dose administration, the concentration value should start monotonically growing: $\frac{\partial g_{concentration}}{\partial t}|_{t=T_{bp}} > 0$, where $T_{bp}$ is any time point before the peak value and after $t_{intake}$.
- After several hours, it reaches the peak value and starts to decrease: $\frac{\partial g_{concentration}}{\partial t}|_{t=T_{ap}} < 0$, where $T_{ap}$ is any time point after the peak value and before $t_{p'}$ is the distance from $t_{intake}$ until the end of the fixed medication regimen cycle, e.g. $p_{regimen} = 24h$. Therefore, $t_{p'} = p_{regimen} - (t_{intake} - t_p)$, where $t_p$ is the similarly computed end time of the previous cycle.
- Taking into consideration the residual concentration value, the difference between the $t_{intake}$ and $t_{p'}$ should be within a certain threshold, i.e. $< 50 mcg/L$. $|g_{concentration}^{t_{intake}} - g_{concentration}^{t_{p'}}| < \mathrm{TH}$.
- The concentration curve whose shape is the most similar compared with the one predicted from ParaSVM will be chosen: $\min_{g_r} \sum_{j=0,\cdots,N_s} (g_r^{t=j} - g^{t=j})^2$, where $g^{t=j}$ stands for the concentration value at time $j$ estimated using the predicted parameters and $g_r^{t=j}$ denotes the one in the refined curve. The set of parameters $y$ corresponding to the best $g_r$ are selected.

This way, using the above described constraints we can adjust the DCT curve for each next drug administration cycle taking into account not only the real concentration values but also adherence to the medication regimen.

## IV. Safety-critical software development

In order to address the safety-criticality issue of the software we first represent the control flow using TAT model that was formally introduced in Section II-A. In this section we introduce the control flow with the reminder for drug administration of *imatinib*, the drug used to treat Chronic Myeloid Leukemia (CML). Then we present the code synthesizer for the *Icycom* platform that we have developed as an extension of the TIMES tool.

### A. Control Flow

The control flow that can be ultimately executed on an embedded electronic device is presented in Figure 2. For simplicity we have eliminated most of the details of the model.

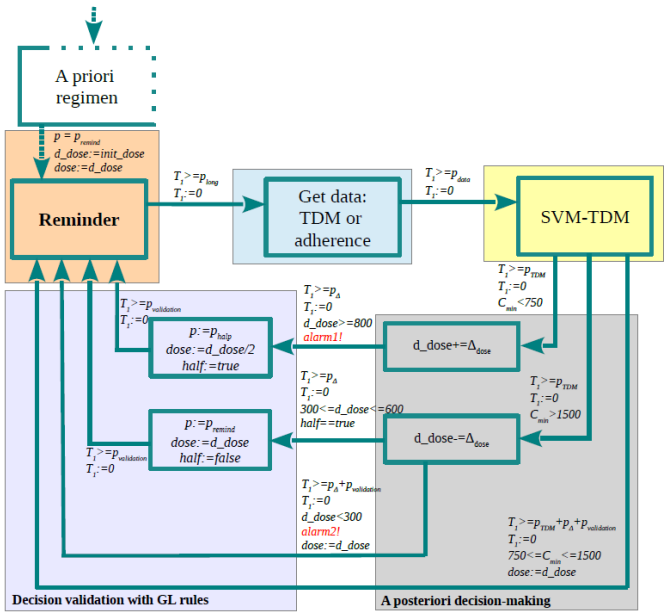


Figure 2: Synthesized control flow

It essentially follows the general feedback loop control flow of Figure 1. It is divided into *a priori regime*, *Reminder*, *data aquisition*, *evaluation of body reaction*, *a posteriori decision making* and *decision validation with GL rules* blocks. The variable $T_1$ of the model is a clock (timer). At each transition, the value of the clock is compared to a reference intervals, e.g. $p_{long}$, $p_{data}$, $p_{TDM}$, $p_{\Delta}$, or $p_{validation}$, representing transitions time guards. Once the guard condition is satisfied, the transition is taken and the clock is reset to zero. When arriving to a new location, the task associated with this location will be added to the scheduling queue to be further activated with a chosen scheduling policy. Some transitions have guards over the values of the drug concentration $C_{min}$ or the computed dosage *d_dose*. For example, the transition from the *SVM-TDM* to *Reminder* location only occurs when $T_1 >= p_{TDM} + p_{\Delta} + p_{validation}$ and $750 <= C_{min} <= 1500$. That means that the model will spend minimum $p_{TDM} + p_{\Delta} + p_{validation}$ of time in the *SVM-TDM* location. When transition is taken, the clock $T_1$ will be set to zero while the drug dosage will be set to the dosage value of the previous cycle: $dose := d\_dose$. We fix the values for the big remind intervals: one day $p_{remind}$ or half a day $p_{half}$; while the dose can be changed with a finer grain than in currently applied *imatinib* GL values defined by the parameter $\Delta_{dose}$ (e.g., 50 mg).

The flow starts with the initialization of the peripheral of an embedded device. Here the daily dose is also initialized with a personalized value (*d_dose:=init_dose*) after being computed externally using the algorithm in Section III. The period $p$ is first set to one day ($p:=p_{remind}$) in case the initial dose value is less than 800 mg, while the actual dose to be administrated is set to the daily dose (*dose:=d_dose*).

The next location of the closed loop flow is *Reminder*. The only outgoing transition from this location leads to the *Get data* location and can be taken only when $T_1 >= p_{long}$. Here $p_{long}$ is set to be such that the rest of the computation for medication regimen adjustment is computed right before the next reminder: $p_{long} = p_{remind} - (p_{data} + p_{TDM} + p_{\Delta} + p_{validation})$. While the other reference intervals correspond to the worst case execution time of the corresponding task, e.g. $p_{TDM}$ is the maximum time required for the drug concentration prediction task added to the queue in the *SVM-TDM* location. The drug concentration prediction must be computed for each drug administration cycle taking into account the adherence to the medical regimen or the real measurement of the drug concentration used to calibrate the SVM-TDM algorithm.

In the *decision-making* stage the decision about increasing, decreasing or keeping the dose is taken. In the present example we assume that the trough concentration value $C_{min}$ (at 24 or 12 hours after the last intake) should lay within the 750:1500 $\mu g/l$ range ($750\ \mu g/l <= C_{min} <= 1500\ \mu g/l$) [29]. If $C_{min}<750\ \mu g/l$ the daily dose will be increased ($d\_dose\ += \Delta_{dose}$) and decreased in case $C_{min}>1500\ \mu g/l$.

In the next stage we check if the value of the daily dose and intake interval are conformed with the rules derived from the formal representation of the *imatinib* GL presented in [25]. The rules are as follows:

1) $dose >= 300$ - the minimum dose assigned should be not less than 300 mg;
2) $dose <= 800$ - the maximal dose assigned should be not more than 800 mg;
3) $dose >= 800 \rightarrow p:=p/2, dose:=dose/2$ - when the dose is equal (or greater) than 800 mg it should be administrated in two shots.

For example, when $d\_dose>=800\ mg$ the period must be set to $p_{half}$ and the dose divided by 2 for each drug administration. The alarm (*alarm1!*) will also be generated since the maximal dose defined by *imatinib* GL is exceeded. The period will be set back to one day next time only after the *d_dose* falls down to 600 mg. An alarm (*alarm2!*) will be generated when we reach the minimal dose value defined in *imatinib* GL (300 mg). The generated *alarm1!* and *alarm2!* will be captured by other model, not presented in this section, that generated alarms packets to be sent to the PC.

### B. Code Synthesizer

We have developed a code generator out of the TAT models in TIMES toolbox for the *Icycom* platform. This way, the more detailed version of the control flow presented above

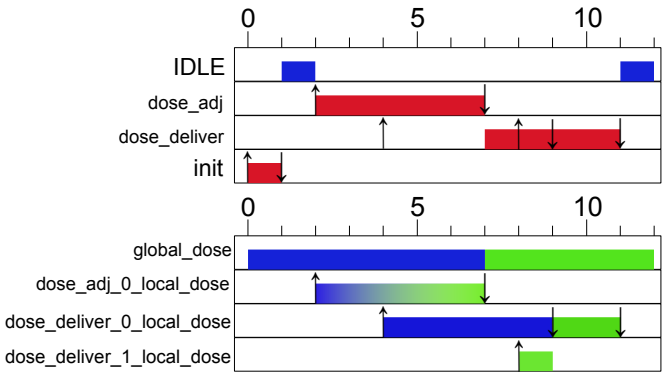


Figure 3: An example for the global variable handling

can be directly synthesized into the executable code. In the synthesized model tasks such as *Reminder*, *Get data*, and *SVM-TDM* represent the class of taks, the platform independent C code for which must be developed separately. The control flow modeled using Graphical User Interface (GUI) of TIMES plays a role of a coordinator of the tasks (drivers) execution. We have extended the existing platform independent TIMES code synthesizer. The *Icycom* code generator synthesizes the final code directly compilable for the *Icycom* platform. This code incorporates the drivers C code as well as the platform dependent coordination of task execution that invokes the tasks based on a chosen scheduling policy. Further we describe some details of the new code generator implementation.

Each embedded platform needs to be initialized. Therefore, at the model level we have added an obligatory *Init* location with an associated initialization task that can be modified and adapted for any target platform independently from the control flow. The *Init* function is present in our control flow depicted on Figure 2, *a priori regimen*. It has to run before all other tasks and be executed only once.

We have also reworked the implementation of the variables passing through the network of automata. The global variables defined in TAT can be evaluated in the guards and set in assignments of the automata network. Their values can also be modified in the task body, which may cause problem to other tasks using the same global variable. Therefore, when the task is added to the scheduling queue by the control flow, the value of the global variable is copied to a local variable, this way, variable modification within task execution does not affect the value of the global one until the task is completed. If the maximum number of the identical tasks in the queue is greater than 1, then we implement a FIFO of local variables. The task body first added to the scheduling queue will take the first local variable. If the second one is added to the queue before the first task of the same type is finished, the current, not yet modified, value of the global variable will be stored at the second position of local variables FIFO. The third task will store the current value of the global variable at the third position and so on. When the first task finishes, all the values will be moved by one position up.

An example of time diagram for global variable passing is presented on Figure 3. Let us assume that we have a model with two tasks (except *init*) and a global variable *dose*. One task named *dose_adj* adjusts the value of the global variable, and the other task named *dose_deliver* delivers the drug based on the value of the global variable. The maximum number of the *dose_deliver* task in the task queue is 2 and the maximum number of the *dose_adj* task is 1. The behavior of tasks is shown in the upper graph while the changes of the global and local variables are shown in the lower one in Figure 3. The *dose_adj* task is added to the queue at cycle 2 and keeps running for five cycles, until cycle 7. When it is added to the task queue, the value of the *global_dose* is copied to the local variable, *dose_adj_0_local_dose*. The local variable changes during the *dose_adj* task execution. After the *dose_adj* task finishes, the value of *dose_adj_0_local_dose* is copied to the global one, so the *global_dose* is changed at the 7th cycle. The *dose_deliver* task is added to the queue in the 4th cycle. Therefore, the global variable copied to the *dose_deliver_0_local_dose* variable still has the unchanged initial one. Thus, the first *dose_deliver* task running during the 8th and 9th cycle will use the initial value of *dose*. This way of handling global variables passing is valid only for the First Come First Serve scheduling policy. For other scheduling policy, since the order of tasks execution might change dynamically, the scheme will be more complicated. It is essential to be aware of the global variables passing mechanism when building new models.

## V. Prototype Implementation

The target device for this drug monitoring and medication adherence is implemented on a platform (Figure 4) based on the *Icycom* processor [4]. This ultra-low power processor, developed by the CSEM SA [30], runs at only 3.2MHz and owns 96KB of on-chip RAM shared for code and data. An RF radio interface is also embedded on-chip, allowing us to exchange data with other *Icycoms*. The motivations behind the use of this processor were mainly its very-low power consumption and integrated radio communication. It makes it an excellent candidate for future systems that could be embedded in a Body Area Network (BAN). The obvious drawback of such processors is the low amount of memory available. This required specific optimization of the code generator, in order to fit within the 96KB of RAM.

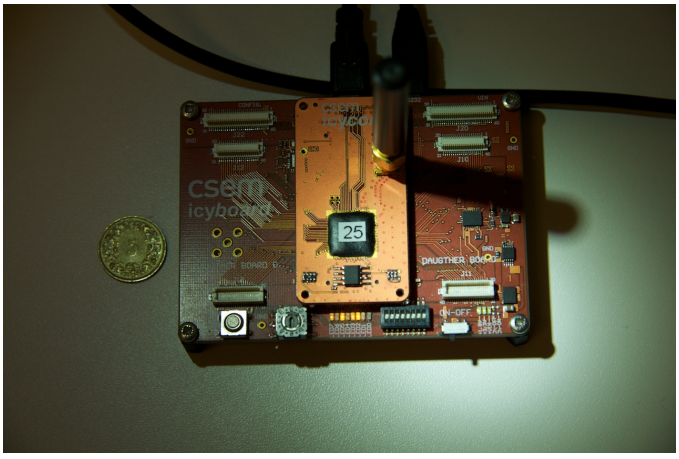


Figure 4: Icycom (under number 25), on an Icyboard

It is clear that the setup used for occasional drug concentration measurement would require real embedded bio-sensors.

Therefore, the platform also contains an interface for such sensors. However, the sensors targeted by this study are not yet embeddable and no human experiment has been conducted. Instead, in order to validate the software, we used a data generator able to mimic the drug concentration evolution of a real human.

The full hardware setup is depicted in Figure 5. It consists of two *Icycom* boards communicating wirelessly, where one of them is connected to a PC through the RS232 port. The

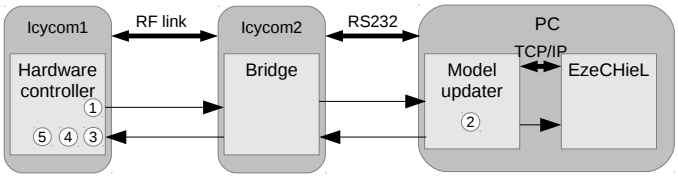


Figure 5: Hardware setup. The numbers denote the tasks

first *Icycom* on the left, "Icycom1", is the portable drug administration reminding device that communicates with the second *Icycom*, "Icycom2". The second *Icycom* just acts as a bridge or radio gateway to enable the communication between the first *Icycom* and a standard PC.

The abstract representation of the software running on the first *Icycom* is the one presented in Section IV. The implemented software was first modelled using TAT model activating tasks execution at specific moments. We have used basic profiling data to estimate the worst case execution time of each task of the model. The periods of the model presented on Figure 2 where calculated such that none of the tasks will take longer than the specified period as described in Section IV. The final code for the *Icycom* was generated using the code synthesizer presented in Section IV-B. The actual implementation of the tasks and its communication with the PC is as follows:

1. Get the data: gets the drug concentration, its value and time of the measurement, and/or the time when the last dose was actually taken. For the actual representation these data must be received by either reading the data from the bio-sensor or some input device (e.g. button) communicating the actual drug intake time. In our prototype, these data are replaced with the simulated values. Since the device may not always be connected to the PC, the data are accumulated during the offline period.
2. SVM-TDM: once there is the connection, the data are sent to the PC to perform the computationally intensive algorithms (see Section III) based on the new data. In the current prototype we outsource the computational algorithms to the PC due to the limitations of the RAM memory on *Icycom*. The updated drug concentration values are sent back to the *Icycom* for further processing.
3. Adjust the next drug dosage based on the predicted concentration computed in step 2.
4. The computed dosage is validated with the GL rules.
5. The reminder to take the drug or alarm signals are produced. The reminder makes one of the diodes blink while the alarm is sent back to the PC to be displayed.

It is obvious that in the offline mode the device will be producing reminders based on the medication regimen defined in the previous online mode while the computation of tasks 2-4 will be skipped. In case the complete algorithm is implemented on the *Icycom* the medication regimen may be updated at each cycle, even in the offline mode.

The software running on the second *Icycom* just forwards what it receives on one of its interfaces (RS232 or radio) to the other one in both directions. This allows seamless communication between the PC and any remote *Icycom*.

In parallel with step 2, the dosage adjustments and new measures are automatically sent to EzeCHieL [31] tool running on the PC. This software allows data interpretation for Therapeutic Drug Monitoring. It offers a graphical interface to display a concentration prediction, based on patient covariates and drug concentration measurements (see Figure 6). It is exploited by the medical doctor and will store the new data in a local database, in order to record any activity related to the patient (dosages and measures).

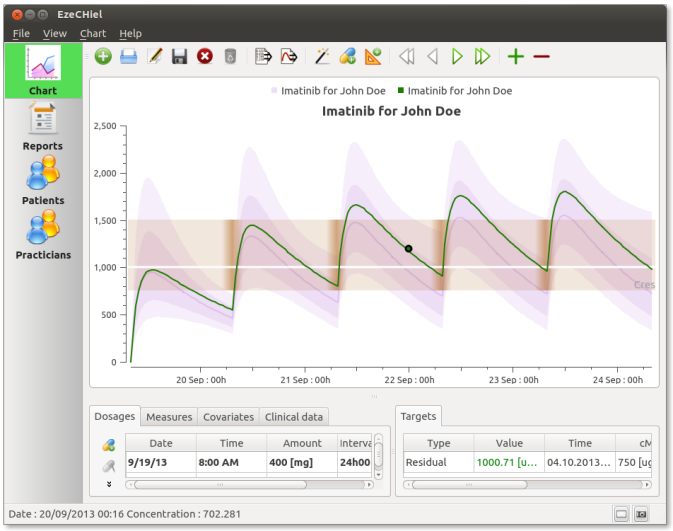


Figure 6: EzeCHieL GUI: Interpreting drug concentration

## VI. Conclusion

TIMES tool is a model checking environment that allows the automatic verification of real-time properties of the control flows with associated tasks modelled with TAT. The synthesizer that generates the code that preserves timing properties of the verified model is an important step towards a safety-assured development of the medical control-flow software. With our work we present a safe approach to software implementation of a medication reminder with enabled personalization of the medication regimen. We show case it with the *imatinib* case study. However, thanks to the component based structure of the software, the method can be applied to other cases by replacing implementation of certain components. For example, for implementing the reminder for the drug with another medical GL, first of all, the administration periods will have to be changed according the GL. Second, the technology of the

bio-chip has to be chosen to measure this specific drug. Third, the initial library for the ParaSVM algorithm should contain samples of the corresponding drugs, while the algorithm will stay unchanged. Fourth, the therapeutic range values of the drug used to compare the $C_{min}$ need to be chosen for this specific drug. And finally, the new GL rules must be extracted from the drug administration GL of the new drug. All described changes correspond mostly to parameters setting, while the synthesis able control-flow structure remains the same. In the future we would like to study the application of our approach to the administration of drugs with faster pharmacokinetics, e.g. analgesic drugs.

## ACKNOWLEDGMENT

The authors would like to thank T. Buclin, N. Widmer and V. Gotta from CHUV Hospital of Lausanne for the precious suggestions on clinical data modelling and provision with sufficient data. The research work presented in this paper was funded by the ISyPeM Project "Intelligent Integrated Systems for Personalized Medicine", with a grant from the Swiss NanoTera.ch initiative, evaluated by the Swiss National Science Foundation.

## REFERENCES

[1] L. Osterberg and T. Blaschke, "Adherence to medication," *New England Journal of Medicine*, vol. 353, no. 5, pp. 487–497, 2005, pMID: 16079372. [Online]. Available: http://www.nejm.org/doi/full/10.1056/NEJMra050100

[2] N. Widmer, L. A. Decosterd, C. Csajka, S. Leyvraz, M. A. Duchosal, A. Rosselet, B. Rochat, C. B. Eap, H. Henry, J. Biollaz, and et al., "Population pharmacokinetics of imatinib and the role of alpha-acid glycoprotein." *British Journal of Clinical Pharmacology*, vol. 62, no. 1, pp. 97–112, 2006.

[3] Imatinib. http://www.glivec.com/files/Glivec-400mg-coated.pdf.

[4] E. Le Roux, N. Scolari, B. Banerjee, C. Arm, P. Volet, D. Sigg, P. Heim, J.-F. Perotto, F. Kaess, N. Raemy, A. Vouilloz, D. Ruffieux, M. Contaldo, F. Giroud, D. Severac, M. Morgan, S. Gyger, C. Monneron, T.-C. Le, C. Henzelin, and V. Peiris, "A 1v rf soc with an 863-to-928mhz 400kb/s radio and a 32b dual-mac dsp core for wireless sensor and body networks," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, Feb 2010, pp. 464–465.

[5] T. Amnell, E. Fersman, P. Pettersson, H. Sun, and W. Yi, "Code synthesis for timed automata," *Nord. J. Comput.*, vol. 9, no. 4, pp. 269–300, 2002.

[6] L. Dayer, S. H. Heldenbrand, P. Anderson, P. O. Gubbins, and B. C. Martin, "Smartphone medication adherence apps: Potential benefits to patients and providers," *Journal of the American Pharmacists Association*, vol. 53, no. 2, pp. 172–81, 2013.

[7] Proteus. http://www.proteus.com/technology/digital-health-feedback-system.

[8] A. Davare, D. Densmore, L. Guo, R. Passerone, A. L. Sangiovanni-Vincentelli, A. Simalatsar, and Q. Zhu, "Metro ii: A design environment for cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 1s, p. 49, 2013.

[9] A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T.-H. Nguyen, J. Sifakis *et al.*, "Rigorous component-based system design using the BIP framework," *IEEE Software*, vol. 28, no. 3, pp. 41–48, 2011.

[10] A. Simalatsar, Y. Ramadian, K. Lampka, S. Perathoner, R. Passerone, and L. Thiele, "Enabling parametric feasibility analysis in real-time calculus driven performance evaluation," in *Proceedings of the 14th international conference on Compilers, architectures and synthesis for embedded systems*, ser. CASES '11. New York, NY, USA: ACM, 2011, pp. 155–164.

[11] S. Bliudze and J. Sifakis, "The algebra of connectors: Structuring interaction in BIP," in *Proceedings of the 7th ACM &Amp; IEEE International Conference on Embedded Software*, ser. EMSOFT '07. New York, NY, USA: ACM, 2007, pp. 11–20. [Online]. Available: http://doi.acm.org/10.1145/1289927.1289935

[12] S. Bensalem, M. Bozga, J. Sifakis, and T.-H. Nguyen, "Compositional verification for component-based systems and application," in *Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis*, ser. ATVA '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 64–79. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88387-6_7

[13] B. Bonakdarpour, M. Bozga, M. Jaber, J. Quilbeuf, and J. Sifakis, "From high-level component-based models to distributed implementations," in *Proceedings of the Tenth ACM International Conference on Embedded Software*, ser. EMSOFT '10. New York, NY, USA: ACM, 2010, pp. 209–218. [Online]. Available: http://doi.acm.org/10.1145/1879021.1879049

[14] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[15] G. Behrmann, R. David, and K. G. Larsen, "A tutorial on Uppaal," in *Formal Methods for the Design of Real-Time Systems: 4th Intl. School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*. Springer, 2004, pp. 200–236.

[16] TIMES. http://www.timestool.com/l.

[17] B. Kim, A. Ayoub, O. Sokolsky, I. Lee, P. Jones, Y. Zhang, and R. Jetley, "Safety-assured development of the gpca infusion pump software," in *Proceedings of the ninth ACM international conference on Embedded software*, ser. EMSOFT '11. New York, NY, USA: ACM, 2011, pp. 155–164.

[18] R. F. Brown, "Compartmental system analysis: State of the art," *IEEE Transactions on Biomedical Engineering*, vol. 27, no. 1, pp. 1–11, 1980.

[19] G. Blau and S. Orcun, "A bayesian pharmacometric approach for personalized medicine - a proof of concept study with simulation data," in *Proceedings of the 2009 Winter Simulation Conference*, 2009, pp. 1969–76.

[20] W. You, N. Widmer, and G. De Micheli, "Example-based support vector machine for drug concentration analysis," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, 30 2011-sept. 3 2011, pp. 153 –157.

[21] W. You, A. Simalatsar, N. Widmer, and G. D. Micheli, "A drug administration decision support system,," in *Bioinformatics and Biomedicine Workshops, 2012 IEEE International Conference*, Oct. 2012, pp. 122–9.

[22] W. You, A. Simalatsar, and G. De Micheli, "RANSAC-based enhancement in drug concentration prediction using support vector machine," in *Proceedings of the International Workshop on Innovative Simulation for Healthcare (IWISH)*, Sept. 19-21 2012, pp. 21–5.

[23] W. You, A. Simalatsar, and G. D. Micheli, "Parameterized svm for personalized drug concentration prediction," in *Proceedings of the 35th annual international conference of the IEEE Engineering in Medicine and Biology Society (EMBC'13)*, 2013.

[24] A. Simalatsar and G. De Micheli, "TAT-based formal representation of medical guidelines: Imatinib case-study," in *Engineering in Medicine and Biology Society, EMBC, 2012, Annual International Conference of the IEEE*, Aug. 28 -Sept. 1 2012, pp. 5078– 5081.

[25] A. Simalatsar, W. You, V. Gotta, N. Widmer, and G. De Micheli, "Representation of medical guidelines with a computer interpretable model," *International Journal on Artificial Intelligence Tools*, vol. 23, no. 03, p. 1460003, 2014. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218213014600033

[26] W. You, N. Widmer, and G. De Micheli, "Personalized modeling for drug concentration prediction using support vector machine," in *Biomedical Engineering and Informatics (BMEI), 2011 4th International Conference on*, vol. 3, oct. 2011, pp. 1505 –1509.

[27] G. Golub and C. F. Van Loan, *Matrix Computations*. John Hopkins University Press, 1989.

[28] J. D. Brabanter, "Ls-svm regression modelling and its applications," *PhD thesis*, pp. 25–28, 2004.

[29] V. Gotta, N. Widmer, M. Montemurro, S. Leyvraz, A. Haouala, L. Decosterd, C. Csajka, and T. Buclin, "Therapeutic drug monitoring of imatinib: Bayesian and alternative methods to predict trough levels," vol. 51, no. 3, March 2012, pp. 187–201.

[30] CSEM. http://www.csem.ch/site/.

[31] EzeCHieL website. [Online]. Available: http://www.ezechiel.ch