Smart usage of Mobile Phones Sensors within an Event Calculus Engine

Student experiences inside AI courses

Valerio Mazza, Michele Solimando

Dipartimento di Informatica – Scienza e Ingegneria
Università di Bologna
Viale Risorgimento, 2, 40136 Bologna, Italy

valerio.mazza@studio.unibo.it

michele.solimando@studio.unibo.it

Abstract. In the following paper we show how we define and integrate an Event Calculus rule system, written by us with xtext Dsl on an android application to work with the mobile phone sensors, in order to smartly use them.

Keywords: Event Calculus, Xtext, DSL Language, Drools, Android.

1 Introduction and Motivation

Nowadays computer systems are so complex, in a way that even expert programmers, have difficulties to verify whether they operate correctly. Moreover it is also very difficult to introduce new features without weaken its internal logic.

In literature there are a few proposal of logical formalisms which can be used to mitigate these drawbacks. One of them, for instance, is the Event Calculus (EC). A possible approach, in fact, consists in using a EC machinery that operates in the deductive mode to monitor the application and verify that it is compliant with what it is expected. Unfortunately, these frameworks are difficult to master either to build advanced applications.

The goal of this Paper, is to present a framework based on EC with a twofold purpose. On one side we have developed in Xtext[1] an Integrated Development Environment (IDE) to allow a domain expert to express a problem in terms of EC concepts. We chose Xtext/Xtend because is a statically typed functional and object-orient language. It automatically compiles to readable Java code. We will describe this process properly in the next chapters. On the other side we have defined a procedure that automatically converts the given problem into an Android application[2] which uses a Drools[3] implementation of Event Calculus.

Consequently thanks to such a integrated system, a user with any level of understanding, can leverage the complexity of defining an EC problem and automatically build an Android application around it. In particular we show a case study in medical rehabilitation that demonstrates the potential of this approach.

This paper is organized as follows: in the follow section we present the framework with the tools and the components of interest that we used. The third section shows the potential of the system developed, taking as example a medical study case. The last section illustrates the objective we have reached, our system limits and future developments.

2 Framework

We briefly describe the tools and components of our framework in order to understand them.

2.1 Event Calculus

It is a logical formalism introduced by Kowalski and Sergot in 1986. There are many different version with varying degrees of expressiveness and efficiency. In this work we use the multivalued reactive variant (most efficient), Drools based, already presented in [1]. This particular version allows you to use any real value, instead of only Boolean value, to represent the magnitudes of the domain.

This formalism owes his fortune to the logical correctness and simplicity. In fact, it is based on two concepts only: the Event that is the notification of an action occurred in the considered domain in a specific time instant; and the Fluent that is single measurable aspects of the domain that can change over time. The EC formalism is completed by other axioms to report the relation between Event and Fluent. For more details see [1].

[1] `http://www.eclipse.org/Xtext`
[2] `http://developer.android.com/index.html`
[3] `http://www.jboss.org/drools`

2.2 Drools and ECA-rules

Production rules are one the most common way to represent knowledge in many areas of Artificial Intelligence (AI). Drool is a Production Rule System that use the Modus Ponens (MP) as rule of inference. In this type of reasoning formally if one proposition P implies a second proposition Q and P is true, it concludes that Q is also true. Rules in PRSs are called instead productions and they primarily express some behaviour that transforms the available information about the domain in new information.

Drools consists of many components: a Production Memory that contains all the productions; a Working Memory that contains information about the current state of the system, the knowledge and the assertions; an Agenda that contains all the activations of all productions. The first action taken by PRS is the pattern matching, that is to build a network of constraints. The algorithm used is RETE algorithm. This algorithm filters the domain knowledge to identify the set of data that satisfies the preconditions of some productions. The active productions are passed to the Agenda. Last task of the PRS is the execution of the actions.

The RETE algorithm can handle the ECA-rules (Event Condition Action) [2]. The ECA-rules consists of three parts: an Event that triggers the invocation of the rule and, if the Condition is satisfied, provoke the Action, a set of updates to the system.

2.3 Domain Specific Language and Xtext

A General Purpose Languages (GPLs) are designed to address a wide variety of application domains, while Domain Specific Languages (DSLs) do not aim to solve every problem in every domain, but focus on a specific restricted one in order to solve problems inside it in a better way than a GPL language would do.

Our approach exploits Xtext capabilities to define a DSL language for EC which can be used by Xtext itself to guide the automatic conversion of any problem into a working application built with a GPL, namely Drools. In addition, the Xtext framework conveniently provides a mechanism to build an IDE for the given DSL language with no additional efforts. Therefore the resulting tool allows to simplify the expression of EC concepts while taking care of the translation of them toward the chosen DSL.

Our language is essentially intuitive because it is similar to the way in which the humans represent knowledge and is readable because the syntax of the rules appears like the spoken language understandable by all.

2.4 Android

Android is a Linux-based operating system designed for mobile devices such as smartphones and tablet computers.

While creating a standard android application, building up a logic to use the phone sensors (and not just them) can be tricky, and not easily maintainable. When you start having several different patterns of behavior to observe, the complexity of the software increase sensibly. With our Event based language it takes less to create a fully working logic, and it separates the two part of the application, making it cleaner and less complex.

3 Case Study

The designed system shows its potential in our case study. We used our DSL to assist doctors in tasks that involved physical effort, such as rehabilitation in physiotherapy. Our goal is to present how different type of users can take advantage of our development environment to simplify their everyday jobs, both doctors and patients.

As we all know, in the last decades, life expectation is getting longer, causing bad outcomes such as medical complications in the older age segment. Specifically, the problems due to reduced joint mobility are particularly insidious because they seriously affect the quality of elderly life style.

We have formulated an EC problem to provide suggestions and incitements to the patient, that are based on the performance during the exercises assigned from a physiotherapist. The suggestions are notified through a simple Android application. The innovation is, thanks to the powerful DSL implemented and the intrinsic logic of the EC, that the physiotherapist will be able to sketch the important characteristic of the particular exercise. He will also have the possibility to create a new one, or modify a given one, without the action of an IT engineer.

Let us see how the tools we presented before can help us to reach our goal.

3.1 Workflow

In the figure 1 we show the entire workflow of the system. In the figure there are two types of users: the physical therapist, expert user of the system, and the patient, passive user.

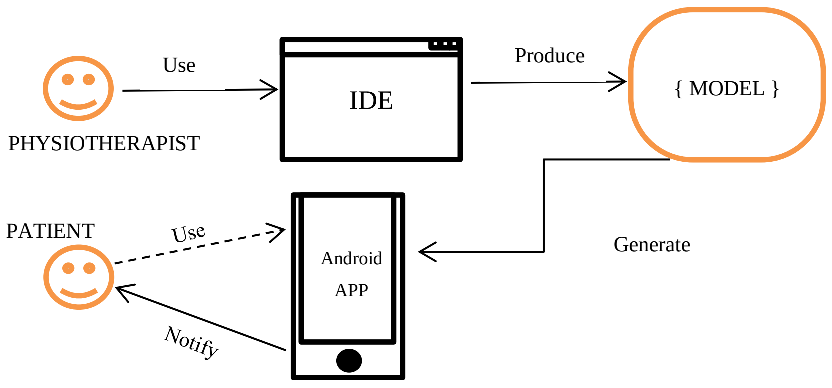
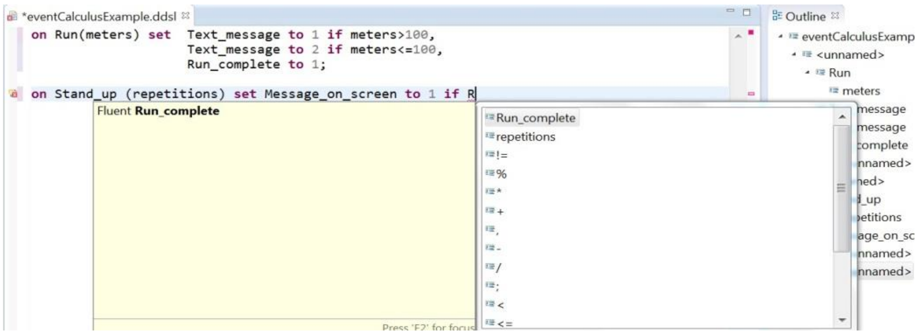


Fig. 1. Workflow

In the first place the Physiotherapist has to express an EC problem throughout the editor built in the Xtext generated IDE as shown in figure 2. In this case the user of our system is the physiotherapist, who will have in hand a series of Events which map the smartphone sensor to the EC world.



```
on Run(meters) set  Text_message to 1 if meters>100,
                    Text_message to 2 if meters<=100,
                    Run_complete to 1;

on Stand_up (repetitions) set Message_on_screen to 1 if R
```

Fig. 2. IDE Editor where the Physiotherapist inserts his Event Calculus rules.

Consequently the physiotherapist will know exactly when a given Event that represents, for istance the accelerometer behavior, is thrown, the Event will map a specific pattern of movement, rather than standing or sitting. The system establishes a correspondence between fluents and interface elements, giving the possibility to the physiotherapist to decide how the application shows its notification of the thrown events.

When the Physiotherapist is satisfied with the model (an example is presented in next paragraph), he may request its conversion into an Android application. The conversion is performed through additional buttons introduced in the IDE, as shown in figure 3.

The application will monitor the behavior of the patient acting accordingly as indicated by the logic defined by the physiotherapist.

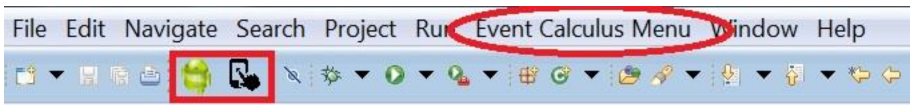


Fig. 3. Custom IDE menus and buttons

In the end, the Physiotherapist, by connecting the patient smartphone to his PC, will install the App directly into it.

3.2 Example

In the following listing we present a basic EC problem consisting of simply movement exercises.

```
on Run(meters) set
      Text_message to “SUPER” if meters>100,
      Text_message to “GO-GO” if meters<=100,
      Run_complete to 1;

on Stand_up(repetitions) set
      Message_on_screen to “Exercise Completed” if
            ((Run_complete == 1) AND (repetitions>10)),
      Message_on_screen to “Restart Exercise” if
            Run_complete == 0;

on SitDown(time) set
      Message_on_screen in (now+1000) to “Are you OK?” if
            time > 10000,
      Button_on_screen1 to “YES, I’m OK” if
            Message_on_screen == “Are you OK?”,
      Button_on_screen2 to “NO, I'm sick” if
            Message_on_screen == “Are you OK?”;

on Press_button_on_screen2 set Call_Doctor to “URGENT”;
```

Listing 1. Rehabilitation exercises synthesized into a EC problem.

With these few rules, we want to show that it is possible to provide information to the EC mechanism by passing parameters along with the event notification. Also note that, for each fluent, is possible to explicit an expression that, when evaluated, provides the value to be attributed to the fluent.

The notifications are represented by strings that may have a numeric identifier. For ease of exposition in this example we used the strings instead of numeric value.

4 Conclusion and future development

The goal of realize an easy to approach language was met. The Editor as well is simple but efficient in the implementation of the rules created.

Our intention is to make something useful for the case of study we have. A limitation of the application is surely the narrowed freedom to use the smartphone features.

In this regard, it could extend the mapping between events and sensors to make the monitoring of patient activity more complete.

Furthermore after careful field testing, followed closely by medical professionals, we could build databases that inform us on unwanted behaviors that lead to disagreeable consequences, for a timely prevention.

From the theoretical point of view, the powerful logic EC engine can be expanded with the use of the ECE-rules (Event Condition Expectation). The ECE-rules are evolution of the well-known ECA-rules and they can express exceptions and anomalies of the system.

Another aspect that would increase the potential of our application, is the possibility of having a fuzzy logic rather than exact, in order to express also approximate reasoning (rather than exact), also very useful in our case study.

References

[1] S. Bragaglia, Monitoring Complex Processes to verify System Conformance, 2013.

[2] Schmidt, K.-U. a. Stuhmer, R. a. Stojanovic and Ljil-jana, Blending complex event processing with the RETE algorithm, 2008.

[3] L. Bettini, Implementing Domain Specific Languages with Xtext and Xtend, Packt Publishing, 2013.

[4] S. Russel and N. P., Intelligenza Artificiale: un approccio moderno, Prentice Hall, 2010.