Making Sense of EveXL, a DSL for Context Awareness

Bruno Cardoso
NOVA-LINCS
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Caparica, Portugal
b.m.pinto.cardoso@gmail.com

Teresa Romão
NOVA-LINCS
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Caparica, Portugal
tir@fct.unl.pt

ABSTRACT
The development of reactive, context aware mobile applications can be a complex task. EveWorks is an engine that provides event detection capabilities to other applications in mobile platforms, interfacing with them through expressions written in EveXL, its own domain-specific language. EveXL is built around simple, everyday concepts, like intervals of time and temporal relations and simplicity has been one of the driving goals for its development. To validate if this goal has been met, we have developed a videogame that implements its reactive behavior through EveWorks and asks players to read and interpret EveXL expressions. We have tested the game with players having little to no programming experience and results were very encouraging, indicating that EveXL is easy to understand and its concepts easily grasped.

Categories and Subject Descriptors
D.3.3. Language Constructs and Features: Frameworks.

General Terms
Design, Experimentation, Standardization, Languages.

Keywords
Event detection engine, domain-specific language, evaluation.

1. INTRODUCTION
Smartphones accompany their owners for most of the time, something that grants them an undeniable potential to deliver the right interaction at the right time. This has been recognized by both the industry and the research community, and several technologies have emerged that facilitate the triggering of contextualized actions – for instance, Near Field Communication (NFC) is an illustrative example of one such technology.

The programming of reactive systems, however, may be a complex task. Beyond the difficulties of acquiring and processing context, there is also the challenge of developing a flexible enough architecture to support the detection of different events for different, application-specific interactions. Indeed, as Carlson and Lisper state [3], the separation between the mechanisms for event detection and the rest of the application's logic facilitates design and analysis of reactive systems. Acknowledging this, the research community has proposed a number of frameworks like Wissen et al.'s ContextDroid [4]. Most of these proposals, however, interface with other applications through Application Programming Interfaces (API). This means that the code implementing the application-framework communication is compiled and, therefore, the application's reactive behavior will have to be defined at compile time. This may not be desirable if one intends to build applications with the potential to change their reactive behavior after having been deployed, or if the desired reactive behavior is not known beforehand – for instance, a smart message delivery system could perform deliveries at message-specific contexts, like "*user leaves work*" or "*user enters home after work and is not in a good mood*". EveWorks has been proposed by Cardoso and Romão [2] as an option that addresses these requirements in a straightforward approach that encapsulates context composition and truly dynamic reactive behavior.

2. EVEWORKS AND EVEXL
EveWorks is an Android context awareness engine that provides event detection services to other applications, interfacing with them through expressions written in a simple, interpreted and declarative domain-specific language (DSL), the EveWorks Expression Language (EveXL). This process is analogous to the way that Relational Database Management Systems (RDBMS) rely on Structured Query Language (SQL) queries and statements to allow external applications to access and manipulate data – in our case, however, the expressions describe events of interest rather than data-related operations. The main distinctive characteristic of EveXL is that everything revolves around the concept of time interval – i.e., a period of time during which some data invariants hold. These data invariants are boolean expressions analogous to SQL predicates. Thereby, when writing EveXL expressions, developers start by writing the *declaration clauses*, declaring which intervals best describe the event of interest. The temporal relations between these intervals are then expressed in the *where clause* through expressions that use the operators of James Allen's Interval Algebra [1] articulated with the classical boolean operators. For instance, the following expression describes the event "leaving home".

```
(1) [I1] AS [location = 'home']
(2) [I2] AS [location != 'home']
(3) WHERE [I1] MEETS [I2]
```

The expression clauses mean that: (1) "I1" is an interval of time during which the sensor "location" reads "home", (2) "I2" is an interval of time where the sensor "location" reads a value different

MOBIQUITOUS 2015, July 22-24, Coimbra, Portugal

DOI 10.4108/eai.22-7-2015.2260303

than "home" and (3) the interval I1 ends when the interval I2 starts. Upon parsing the expression above, EveWorks will start to periodically read the involved sensors (in this case, just the "location" sensor) and trigger the event whenever it occurs.

3. A GAMIFIED EVALUATION OF EVEXL

EveXL has been designed for simplicity. To assess if this objective has been met we have developed a browser-based videogame that requires players to read and execute events written in EveXL, the Dungeon of Colors (DoC) – our basic assumption being that the correct execution of a described event implies that the player has understood it. The game's architecture includes three components: (1) the DoC app, a mobile application running on the smartphone that interfaces with EveWorks; (2) a browser rendering the game and (3) a server brokering the communication between the browser and the DoC app (server push).

The gameplay is truly simple: a character must progress through a sequence of locked rooms, whose doors can only be opened by using a smartphone to perform specific events. Each time the character arrives at a new room, the browser will show the player the EveXL expression for the event that grants access to the next room while simultaneously sending this expression to the server. The server will then push the expression to the smartphone's DoC app, which, in turn, will forward it to EveWorks for parsing and processing. When EveWorks detects that the event has occurred, it will notify the DoC app, which will notify the server which, in turn, will advance the game character to the next room.

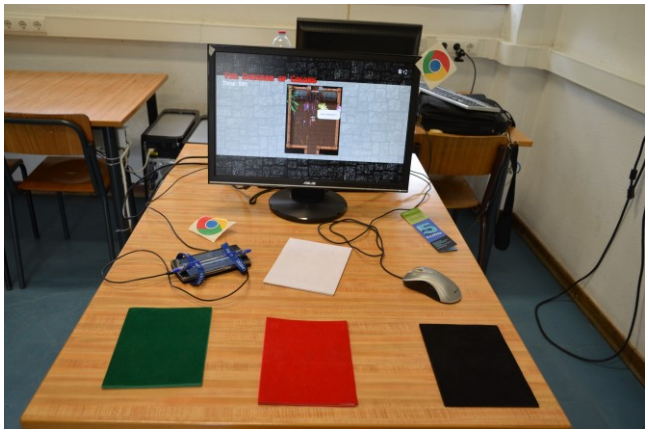

Figure 1. Dungeon of Colors' setting. A browser showing the gameplay action, a mobile phone running EveWorks (left of white pad) and four colored pads with embedded NFC tags.

For the sake of simplicity we have only used the smartphone's NFC sensor and recalled it "color". This design becomes meaningful because we have used four colored pads, each embedding an NFC tag transmitting a "white", "green", "red" or "black" value, in accordance with the color of its pad. For instance, an interval of time with the *color='green'* data invariant corresponds to a period of time during which the smartphone is laid on top of the green *pad* (that is, the smartphone's NFC sensor is reading the "green" data value for a period of time). In order to grant access to the rooms, participants were asked to execute the declared events, by placing the smartphone on top of the colored pads in the correct sequence (further details in the next section).

We have had 37 participants (27% female) with ages ranging from 14 to 60, averaging 22. Programming experience ranged from 0 to 30 years, averaging 2.

4. EVEXL SNIPPETS

Two illustrative expression examples that have been used to unlock rooms in the DoC videogame are described below.

```
(1) [X] AS [color='red' OR color='green']
(2) [Y] AS [color='black']
(3) WHERE [X] IS AFTER OR IS MET BY [Y]
```

(1) X is an interval of time where sensor "color" reads "red" or "green" values; (2) Y is an interval where it reads "black" and (3) Y occurs before X. This event will succeed if the smartphone is laid on top of the red or green *pad* for a time and, after a while (IS AFTER) or immediately after (IS MET BY) it is laid on top of the black *pad*.

```
(1) [I1] AS [color='red']
(2) [I2] AS [color='green']
(3) [I3] AS [color='black']
    WHERE
(4) [I1] MEETS OR IS BEFORE [I2] AND
(5) [I2] MEETS OR IS BEFORE [I3] AND
(6) DURATION OF [I3] > 6s
```

(1) I1 is an interval of time where sensor "color" reads "red"; (2) I2 is an interval during which it reads "green" and (3) I3 is an interval where "black" is read; (4) interval I1 meets or takes place before I2, (5) I2 meets or is before interval I3 and (6) interval I3 takes longer than six seconds. This event occurs when the smartphone is sequentially laid on top of the red, green and black *pads* having been left over the last one for more than six seconds.

As an alternative to any room default event, the game allows players to write their own events in EveXL, thereby doubling as a learning ground for advanced players wishing to further explore EveXL. When these custom events are detected, the character will also progress through the room.

5. CONCLUSIONS

Our tests indicate that EveXL is straightforward and easy to understand, since no participants have manifested particular difficulties. This is illustrated by the low mean number of errors (< 0.5 mean errors per participant) and the average time of just 12 seconds taken to execute events (placing the smartphone on the pads in the correct sequence) among players with virtually no programming experience. As such, we can extrapolate that the fundamental concepts of EveWorks – time intervals and temporal relations – are easily mastered by more experienced programmers.

6. REFERENCES

[1] Allen, J. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM 26*, 11, 832-843.

[2] Cardoso, B. and Romão, T. 2014. Presenting EveWorks, a framework for daily life event detection. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (EICS '14). ACM, New York, NY, USA, 289-294.

[3] Carlson, J., Lisper, B. 2004. An Event Detection Algebra for Reactive Systems. *Proc. of the 4th ACM international conference on embedded software (EMSOFT 04).* ACM, New York, USA, 147-154.

[4] Wissen, B., Palmer, N., Kemp, R., Kielmann, T. and Bal, H. 2010. ContextDroid: an expression-based context framework for Android. *In Proceedings of PhoneSense 2010*.