
XSED: notations to describe status-event ubiquitous computing systems

Jair C Leite

Universidade Federal do Rio Grande do Norte
Av. Sen. Salgado Filho, 3000
Natal, RN 59072-970 Brazil
jair@dimap.ufrn.br

Antonio Cosme

Universidade Federal do Rio Grande do Norte
Av. Sen. Salgado Filho, 3000
Natal, RN 59072-970 Brazil
antonioscosme@hotmail.com

Abstract

Ubiquitous systems require a flexible and scalable developing environment. Since they are spread in the environment, they should deal with both status and event phenomena. The XML Status-Event Description (XSED) is a notation to describe configurable systems, which can be developed by linking status-event components. We have developed a software tool to generate a configuration of SE components from XSED. The configuration is itself a SE component wrapped as a Java Bean that can be plugged in a middleware to ubiquitous system.

Keywords

Status-event, XML, software languages, ubiquitous computing

ACM Classification Keywords

D3.2. Language Classification: Very-high level languages.

Introduction

Computing systems are become embedded into the environment. They provide a natural and transparent interaction where sensors provide input data by monitoring user's activities and other environmental variables. These data also can be used to control

actuators in the environment. The environment can be a local site or a wide geographical area.

These ubiquitous systems need to be flexible and scalable. They should be flexible to allow easy modification of components, configuration, and control and temporal information. They should be scalable to allow new input and output components to be plugged or unplugged.

In ubiquitous or pervasive computing environments, sensors could monitor different kinds of phenomena. Status phenomena as temperature, pressure, sound are those that are more permanent in time. Event phenomena such as people entering in a place or an alarm ringing are those that happen at a particular moment. Of course, at computing level, these sensor values are translated into discrete data samples at particular moments. However, at design time, it is more appropriate to deal with those important conceptual differences.

We are motivated by the idea that to deal with both status and event information, ubiquitous systems should be implemented as a collection of status-event (SE) components. The SE components are linked together following a configuration. It is possible to have different configuration to the same collection of SE components defining different systems. New components can be added to a configuration. The properties of a link between two components are described using annotations. Annotations are descriptions about the rate and the initiative of data sample.

The objective of our project is to develop ubiquitous systems based on configuration of SE components. In order to achieve that goal, we have developed a collection of notations using XML. The notations are collectively called XSED (*pron. exceed*) – XML Status–Event Description.

The goal of this paper is to present briefly the XSED notations and a tool that we use to develop prototypes using them. The tool parses the XSED files and generates Java implementation of a configuration of linked SE components. The implementation is by itself a SE component wrapped as a Java Bean.

The XSED notations

A simple scenario illustrates an application of status-event systems. Consider a typical office or room where temperature sensors provide information to control the air-conditioner. The room has also sensors that inform if someone enters or leaves the room. If the ambient temperature is above a set value and there is someone in the room, the system should turn the air-conditioner on. The system recognizes registered persons when the sensor triggers events. The system can also poll the temperature sensor to get its status.

Using XSED, we describe the system using a configuration of components. A configuration is a collection of named component slots with typed input/output nodes and linkage between them. In the XSED configuration file, each component lists its inputs and outputs each with a status/event tag and type (see figure 1 for an example). A <links> section specifies the connections between the components.

A small fragment of this configuration is described using XSED in figure 1. The component named 'world' is the connection of the system with the environment. Its input and output are those ones that link the system with the environment.

The "person" component monitors if someone enter or leaves the room. There should be a component for each person. It receives the events "person enters" and "person leaves" that allows it to inform the status of the person. The component also has output events to fire the "movement" event to another component whenever there is someone entering or leaving.

There are more two SE components in our example, but they are not shown in figure 1. The "room-status" component receives the status of all person components and output the status informing if there is somebody in the room. The "turn-on" component is triggered by movement event, which is the output event of the person component. When an event happens, this component checks the room-status and the room-temperature. If there is someone in the room and the temperature is above the set temperature this component fires an event to turn the air-conditioner on.

Each component is described in a particular file using XSED notation for components. A single SE component has initial input and output declarations each of which may be a status or event. This may be followed by default status-status mappings giving output status in terms of input status. The states also contain status-status mappings. Finally the transitions describe the effects of events in each state. Each transition has a single input event that triggers the transition and a condition.

The components are dynamically linked. For instance, when a component requests an input status, it asks dynamically ask to the configuration component to its linked component. A detailed description is in [2].

```
<config>
  <export name="world" />
  <components>
    <comp name="world">
      <input>
        <event name="person_enters" type="void" />
        <event name="person_leaves" type="void" />
        <status name="room-temp" type="float" />
        <status name="set-temp" type="float" />
      </input>
      <output>
        <event name="turn-on" type="int" />
      </output>
    </comp>
    <comp name="person">
      <input>
        <event name="person_enters" type="void" />
        <event name="person_leaves" type="void" />
      </input>
      <output>
        <event name="movement" type="void" />
        <status name="person_in" type="bool" />
      </output>
    </comp>
    ... more components ...
  </components>
  <links>
    <status id="3562">
      <from name="person" port="person_in" />
      <to name="room-status" port="person_in" />
    </status>
    ... more links ...
    <event id="32">
      <from name="person" port="movement" />
      <to name="turn-on" port="movement" />
    </event>
    ... more links ...
  </links>
</config>
```

figure 1. XSED description of a configuration.

In the annotations description (not shown in this paper due to limited space), unique link identifiers refer to links in a configuration file and specify properties of those links that can then be used to optimize the runtime behavior of the system. The annotations include: *initiative*, which defines whether status links should be demand driven or data driven; *time*, which defines the timeliness of a link; *last-or-all*, when a sequence of events are fired whether all of them are important or just the last; *synchronization*, which informs the timeliness annotations can mean that events and status change are not passed on in the same temporal order as they are produced.

It is very easy to add new components. For example, we can add new “person” components to monitor different people. The room-status component can be easily modified to inform the number of people in the room (another status output).

Generating SE Components to ECT Platform

The Equator Component Toolkit [3] (ECT) is a collection of JavaBeans ‘components’ that provides a library of tools for constructing interactive ubiquitous computing applications. We have developed a software tool that generates a Java Bean version of the SE Component to run in the ECT platform [2]. SE components are transformed into Beans using a wrapper class generated from the schema. It will enable us to experiment the prototype with sensors, actuators and other components that have existing drivers/wrappers for ECT. For each input and output, both status and event, a Java slot is provided, but these are expected to be used differently depending on the type of node.

Conclusion

XSED allows descriptions of systems that include both status and event phenomena to be included naturally and without having to prematurely transform the status into discrete events. The notation separates components, configuration, linking and annotation allowing reuse, scalability and flexibility. It also allows global analysis (by hand as now, or in future automated) to feed into optimization of the execution over the infrastructure. We have been working on a software tool that generates a core component to process both status and events in ubiquitous systems.

Acknowledgements

The authors would like to thanks to Alan Dix and Adrian Friday to previous support on XSED definition. Jair would like to thanks CAPES Brazilian Foundation for financial support.

References

- [1] Dix, A.: Status and events: static and dynamic properties of interactive systems. in Proc. of the Eurographics Seminar: Formal Methods in Computer Graphics. Marina di Carrara, Italy. (1991).
- [2] Dix, J. Leite, and A. Friday (2007). XSED – XML-based Description of Status-Event Components and Systems. In Proceedings of Engineering Interactive Systems 2007, IFIP WG2.7/13.4 Conference, Salamanca, March 22-24, 2007, LNCS (to appear).
- [3] Greenhalgh, C., Izadi, S., Mathrick, J., Humble, J. and Taylor, I. A Toolkit to Support Rapid Construction of UbiComp Environments. Proc. of UbiSys 2004. Nottingham, UK. Available online at <http://ubisys.cs.uiuc.edu/2004/program.html>.