Ubiquitous Computing Gaurav Kumar
University of Illinois at Urbana Champaign Software Research Group
Department of Computer Science October 2002
Request For Comments: 004

## 1.0 Status of this Memo

The memo is in the Request for Comments state. This memo will undergo changes as and when required.

## 2.0 Introduction

In the world of Ubiquitous Computing, one of the challenges is to present to the end-user, an intuitive and easy way to program and configure the "Smart Spaces" environment.

In the present infrastructure that we have for Gaia, our ubiquitous operating system, we have a lot of services like Discovery/Context and active devices/ applications that we can use to configure an "Active Space" for an activity, say a Gaia Meeting. Context –Aware Scheduler aims to work as a Work Flow Management System, and aims to make it easy for users to use the present infrastructure to develop different and sophisticated everyday scenarios that may involve a lot of users and devices.

The word "Context" in Context-Aware Scheduler may signify a person's entry/leaving the room, weather/temperature of the outside world, location of a person in the room, or change in the stock price of Microsoft.

Scheduler collates information from all the Event Channels available in Gaia, process the events, and based on the events trigger actions for active devices/applications.

## 3.0 Requirements

### Configure Preferences for an Activity

Scheduler should provide an easy and transparent way to configure the Active Space according to an individual's preferences, through a GUI.

### Monitor for Events

Scheduler serves as a listener to all the Event Channels present in Gaia.

### Schedule Actions on Event Based Triggers

Scheduler should trigger an action (say start) to active devices/applications, based on a certain event, which has been predetermined. Scheduler should be able to process the information encoded in an event structure.

### Resource Conflicts

Scheduler should provide a way for handling resource conflicts related to Two persons requesting resource at the same time or dispatching conflicting actions for a given resource at the same time.

### Optimizing User Satisfaction

In face of different individuals having different preferences for the room, the Scheduler should aim to schedule actions in such a way so as to optimize satisfaction for all the users of an Active Space.

## 4. Description

To model the Context Aware Scheduler, we use a State Machine as a basic component for our architecture. The State Machine has been used before for modeling scenarios, and Scheduler tries to use most of features provided by State Machine and related models like State Charts.

A State Machine is a more powerful mechanism for encoding actions in comparison to certain Events than a table lookup as it can encode a sequence of steps to be performed before a certain event results in a certain predetermined action.
(So a State Machine has History information, which a table lookup cannot have).

**Basic Component**

A State Machine consists of states and transitions between states are made using event triggers, and with each event trigger you can associate an action to be performed.

In a classical State Machine model , to make a transition between two states A and B you have an Input/Output tuple, here the Input is the Event and Output is an Action/set of Actions associated with that Event.

## StateCharts

If we follow the normal State Machine approach it will very soon lead to explosion both of States as well as transitions between States. To counter this problem in 1973, D. Harel came up with the idea of State Charts.
Scheduler uses two important features of the State Chart approach in its design

a) **Hierarchy:** This concept entails each state to consist of *substates*. For example, A person may be sitting in a room and he may be listening to music, playing chess or watching a movie on a DVD.

   So here his being "alone in the room" is a *superstate* and the *substates* are :
   a) Listening to Music
   b) Playing Chess
   c) Watching a movie on a DVD

b) **Orthogonality:** By orthogonality, we mean that two *substates* may be active at the same time. For example, in the above case a person may be listening to music and watching a DVD at the same time.

Using StateCharts solves two problems: -

a) *Reduces the number of states*
By using the concept of orthogonality, we have more than one state active at a time. By having two states active at the same time, say a person listening to music (let us denote it by L) and playing chess (denoted by P), we can model four states possible from the combinations of these two say

i) L & P
ii) !L & P
iii) !L & !P
iv) L & !P

b) *Reduces number of transitions*

By using the concept of Hierarchy of states one can reduce the transitions between states. Now if the State Machine switches from one *superstate* to another *superstate*, one need not have transitions from every *substate* to the next substate, but only one transition is needed between *superstates*.

*Substates :-* There can be transitions encoded between any two given *substates*, associated with event/action just like between *superstates*. When a *Superstate* receives an event it tries find a transition associated with that event, if it finds one, it follows the transition to the next Superstate. If not found, then the event is passed on to each of the existent substates(both active and non-active), and their associated transitions are collected and recorded to be dispatched to an active device/application.

## What does a State Machine represent ?

*A State Machine represents an activity, which may be a group activity or an individual one . It can represent a meeting of a certain group of individuals ,or a person sitting alone in his home watching baseball on TV. State Machines make sense for even everyday activities such as guests visiting your house.*

*Let us take a sophisticated example*

Suppose a family is having a Thanksgiving break and enjoying in their home, which is an Active Space. Their house is in a "Owner" mode say, the Scheduler is managing their Active Multimedia devices etc , say the lights in the house are at their preferred lighting , the MP3 player is shuffling their predetermined sequence of MP3's etc.

Based on different triggers, which may be temporal, their Active TV is changing the channels etc. Now suppose some guests arrive in the house, the Scheduler will get these events from the Discovery Service and change the Active Home to the "Guest" mode. Now the Scheduler can play a welcome song for the guests , increase the brightness of the lights to the maximum and instruct the MP3 Player and the TV to stop using the preferences of the owner. Now suppose the Guests who have some kids with them , then WFMS shall make sure that all the Multimedia components in the Active Home will have their R-rated files locked. The Scheduler will continually monitoring the Context Service and suppose the Context Service reports (by contacting the Yahoo Weather ) that it is going to snow very soon , the Scheduler will announce or display on the TV that it is going to snow , so that the guests can return to their houses soon . Say the Context Service reports a Breaking News ( by contacting the cnn.com) , the WFMS will change the present TV channel or the present video being played to the CNN channel . Suppose the guests who have arrived are a bit funky and want to use all the possible devices , and suppose the Active Home does not have resources to support their requests , the Scheduler will notice this , and deny such requests that may clog the Active Home System . When the guests leave, the Active Home reverts back to the "Owner" mode.

**More than one State Machine Required?**

Suppose you want to program an Active Space for two different people with different preferences. In that case these two different individuals have two different State Machines with their individual preferences encoded in them. One of the major problems faced in the design of the Scheduler was how to make two different State Machines work together concurrently. That could have involved communication and synchronization between State Machines, which has been a hard problem to solve. Scheduler takes a neat approach to solve this problem by using a Resource Controller for each resource. Now two different State Machines operating concurrently in an Active Spaces environment can be agnostic of each other. They can dispatch whatever actions they want to a Resource, and everything is handled by the Resource Controllers.

**What is a Resource?**

The word Resource as used here denotes an Active Device in the Gaia infrastructure such as the X10 enabled lights, or an Active Application in Gaia such as a MP3 player or Calendar.

**Resource Controller: How does it Handle Resource Conflicts?**

The Resource Controller is an intelligent entity which acts as an interface to a resource, provides a transparent way of dispatching actions to a resource, and takes decisions regarding which actions to process in case of conflicting actions from two different State Machines, or two different substates of the same State Machine.

Here the problem comes of asynchronous activity, say an event is processed and corresponding actions are dispatched to a resource controller by separate State Machines or different substates and they arrive at Resource Controller different actions at different points of time. This is not desired, and it is solved by collecting all possible actions (which may be a no-op) for a particular event, and dispatching it to the Resource Controller.

A Resource Controller can have different policies for handling conflicting actions by different State Machines/ different substates like First Come First Served (FCFS) or some Priority based mechanism. Also it can have certain mechanisms so that it prevents starvation, by allowing only say n number of requests of a person to be rejected.

## 5. Interfaces

To Do

## 6. Conclusion

To Do