A CSP BASED ONTOLOGY FOR A SMART HOME

José Ignacio Rendo Fernández, Iain Phillips, Adrian Lawrence
Department of Computer Science,
Loughborough University, Loughborough, LE11 3TU, England
J.I.Rendo@lboro.ac.uk

ABSTRACT

This paper addresses the problem of establishing collaboration between devices in a home environment, even when no previous knowledge is known or stored on the devices. It focuses on two basic questions: how a collaboration is established between devices and how it is described. A true collaboration needs semantic information about the services a device offers and how these services can be combined. After discussing current technologies for this aim, we propose a network of embedded processes in home appliances. These processes are modelled with a process algebra called CSP (Communicating Sequential Process) and represented with an ontology. Finally, the authors describe the advantages of this new approach and briefly suggest possible future work in this topic.

KEYWORDS

pervasive, ontology, semantics, CSP.

1. INTRODUCTION

In the future, a possible home environment would allow all home devices to automatically collaborate offering services to their users. As an example, consider an inhabitant watching one of their favourite DVD movies in a darkened lounge. Suddenly, the phone rings and subsequently, the system indicates this to the user through a remote control and provides a choice as to whether to answer the call or not. If the user chooses to answer the phone, when the call is taken, the system pauses the DVD and increases the intensity of lights. When the call is finished, the system restarts the movie and reinstates the original darkness level.

Each of the devices (the lights, the DVD, the phone and the user interface device) have been separately installed and never previously used in collaboration. Our aim is for the system to offer this service to the user without being pre-programmed, totally spontaneously. Issues raised include: how does the phone know that the DVD needs to be paused and the lighting level raised? There needs to be some intelligence (such as an agent) working on the user's behalf to coordinate the actions of devices (Davies, 2002).

Many protocols, such as Jini (http://www.jini.org), UpnP (http://www.upnp.org), HAVi (http://www.havi.org), exist to accomplish both device discovery and service invocation. However, the ability to establish spontaneous collaborations and to describe them remains challenging. All of the previous efforts describe devices and services as set of methods and attributes. However, none of these efforts has enough semantics for spontaneity, i.e. enough meaning for extracting the useful information necessary for device collaboration. Furthermore they do not support a rich description of services allowing for cooperation.

One possible way to resolve these two questions entail the use of ontologies to enrich the description of the devices. Examples of them are the one proposed by FIPA (Foundation for Intelligent Physical Agents) (http://www.fipa.org), for cooperation of agents, and CC/PP (Composite Capability / Preference Profiles) (http://www.w3c.org/2001/di), for adapting web content to any kind of device. However, both of them do not provide enough information for composing services from devices. For example, with the former is not possible to specify if a device has an AV input/output. The latter is simply intended to describe the rendering capabilities of devices for presenting web content. The University of Maryland has developed a tool for building collaborations among devices in an office, TEC (Task Computing Environment) (Masuoka, 2003). It complements UPnP with the technologies for the Semantic Web (RDFS and DALM-OIL). This approach allows enriching the description of devices, giving information of what the devices do and how to use them.

With an ontology it is possible to describe devices. However, the issue of describing a collaboration remains. There are several methods for describing collaborations among software entities, but in this work

we concentrate of formal models of concurrency. The nature of the network, with a great number of devices collaborating, suggests a formal method that supports concurrency as these characteristics allow the designers to use the method's primitives, which nature is to describe programs collaborating among them, to model and prove the correctness of their prototypes when they are developing them. Methods with these characteristics are SDL (Specification and Description Language), CSS, or CSP (Communicating Sequential Processes). The aim is to complete the ontology that describes devices with all the semantics of one of these methods to give the possibility for describing the collaboration correctly.

In this proposal, we employ CSP, developed by Hoare (Hoare, 1985). Devices are modelled as a set of communicating processes engaging in events, instead of a set of objects. CSP is a simple-to-use process algebra excellent for developing systems connecting building blocks that can be reusable. The rest of the paper will present the ontology for this network.

2. CSP ONTOLOGY

2.1 Brief introduction to CSP

CSP describes systems as a set of interacting components engaging in events. Fundamentally, components can be considered as processes that communicate with each other through channels, a channel being a set of events. Each process represents the behaviour of an object in terms of a set of events, i.e. actions of interest. For example, the interrupted cinema session (from section 1) can be modelled with the following processes, one for each device and another one for their collaboration.

Figure 1: Process diagram representations

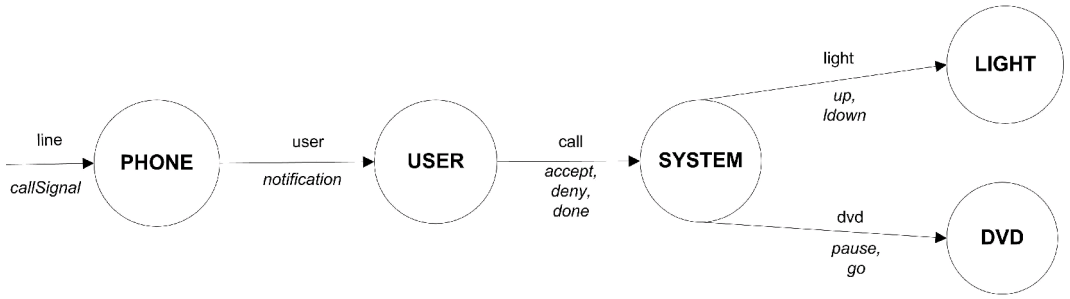


Figure 1 is an informal representation of the following CSP equations for devices.

$$PHONE = line?callSignal \rightarrow user!notification \rightarrow PHONE$$
$$USER(waiting) = user?notification \rightarrow ((call!accept \rightarrow USER(talking)) \ \Pi \ (call!deny \rightarrow USER(waiting)))$$
$$USER(talking) = call!done \rightarrow USER(waiting)$$
$$DVD = dvd?go \rightarrow DVD \mid dvd?pause \rightarrow DVD$$
$$LIGHT = light?up \rightarrow LIGHT \mid light?down \rightarrow LIGHT$$
$$SYSTEM = call?accept \rightarrow (dvd!pause \rightarrow SYSTEM \ ||| \ light!up \rightarrow SYSTEM) \mid$$
$$call?done \rightarrow (dvd!go \rightarrow SYSTEM \ ||| \ light!down \rightarrow SYSTEM)$$

The words in capital letters represent processes while the other ones are events. There are two special events, which correspond to the actions of reading and writing on the channel. Both are described by the name of the channel, the character "!" for writing or "?" for reading, and the variable transported on it. In the example, the phone (process *PHONE*), every time that it receives a call (reading message *callSignal* on channel *line*) it sends a notification to the user (sending the message *notification* on channel *user* to process *USER*). If the user wants to answer, (sending the message *accept* on channel *call* to process *SYSTEM*) the controlling system will send the orders to pause the DVD (sending the message *pause* on channel *dvd* to

process *DVD*) and to increase the light intensity (sending the message *down* on channel *light* to process *LIGHT*). Once the call is finished (message *done* on channel *call*) the user continues watching his movie in a dark room. Depending on the first event engaged, processes will behave in one way or another. For example, the USER offers two possible behaviours, either answer or not the phone. All the collaboration is modelled with the special process *COLLABORATION*.

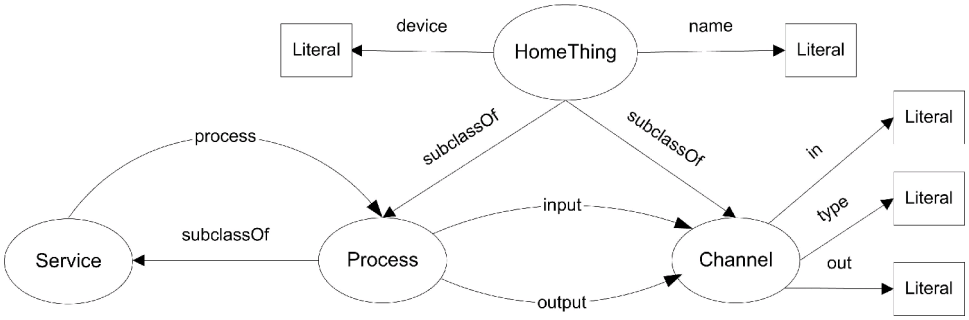
$$COLLABORATION = PHONE \underset{user}{\|} USER(waiting) \underset{call}{\|} SYSTEM \underset{light \cup dvd}{\|} (LIGHT \,|||\, DVD)$$

Which means that the *COLLABORATION* consists of a set of parallel processes (PHONE, USER, SYSTEM, DVD and LIGHT).

2.2 Ontology description

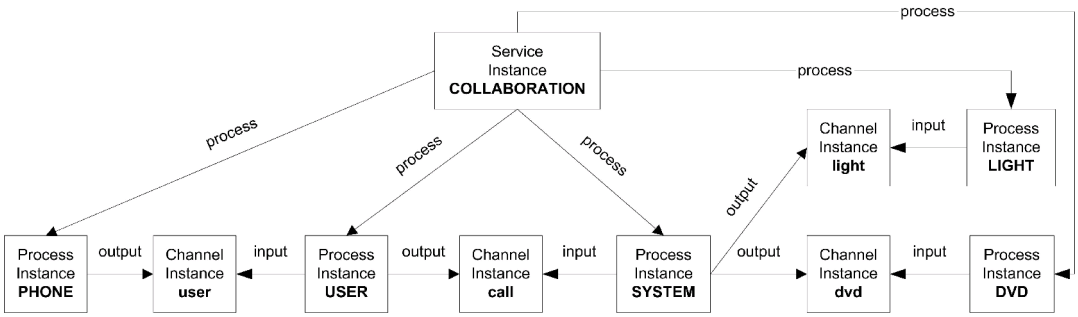
Here we discompose devices as a set of processes that can interact through channels with understood and agreed protocols. These entities and their relations are modelled with CSP equations and described with an ontology. The ontology presents three main classes, one for processes, another for channels, and the remaining one for services, a process composed of other processes.

Figure 2: Ontology Specification



In Figure 2, every process is characterized by its input and output channels, with channels defined by type and direction, i.e. input, output or both. This ontology has to be completed with instances of the three main classes, so every time a new device joins the network, it has to add its own instance to the ontology. The more entities are declared, the more complex relations can be inferred from the ontology. Figure 3 is a representation the collaboration between the phone, the DVD, the light system and the controlling system.

Figure 3: Ontology Instance of the Collaboration between devices



2.3 Inference process. Generating collaborations

In the Inference process new statements are extracted from the ontology, being a statement a relation among elements of the ontology. For example, in the example of Figure 3, the process *LIGHT* and the channel *light* are related by the property input, so from this relation it is possible to declare the statement

"*The process* LIGHT *has an input channel called* light". At this point, it is possible to declare the ontology as a set of statements. The responsible agent for inferring statements is the "reasoner", a black box that receives as inputs statements and rules and outputs new statements, i.e., new collaborations. Rules are the criteria follow by the reasoner to inference the new collaborations.

The service *COLLABORATION* presented in section 2.1 can be derived from the statements of each device and it channels. For the controlling system, "*a process* SYSTEM *has an output channel of type command*", for the light system "*a process* LIGHT *has an input channel of type command*" and for the DVD, "*a process* DVD *has an input channel of type command*" (command is a type of channels in which is possible to transport orders or commands to devices). If the reasoner is fed with these three statements and the rule "connect process A with input channel of type T with process B with output channel of type T", the inference process will try to connect the processes *SYSTEM*, LIGHT and DVD through two channels of type command. The statements generated can be added to the ontology and feed back into the reasoner to allow discovery of further new statements. The more complex the ontology is and the more number of rules are there, the more number of new statements can be derived.

3. CONCLUSION

Our approach allows an easy way of representation of devices and their collaborations. The use of a formal method allows a classical engineering approach, in which the system is built of components (processes and channels) related by a well known predefined methods. Another advantage of CSP is that is possible to ensure and to predict the correct operation of the network preventing problems such as deadlocks and starvation, typical of parallel programming. Also, as the nature of the network is dynamic it is necessary to avoid dangerous or useless collaborations between devices, referred as the feature interaction problem in (Kolberg, 2003).

At this time of developing, the principal limitation is that the only rule for creating services is the channel matching. The next step consist to enrich the ontology with new elements for representing contextual user information (Prekop, 2003), i.e., what and where the user task is, what the user knows, and what the system capabilities are. Another line of future work is to give enough semantics to generate customizable graphical user interfaces (Quantrill, 2003).

The use of the CSP based ontology makes the system independent of the platform and programming language. However, for the implementation, Java is the chosen language, with JCSP (Java CSP) API (http://www.cs.kent.ac.uk/projects/ofa/jcsp/), for programming in CSP style and the Jena API (http://jena.sourceforge.net) for dealing with ontologies.

The applications of this approach are beyond the smart home scenario. Due to the abstraction given by CSP, it can be applied to any kind of pervasive computing purpose. It is only necessary to create the corresponding instances and rules to fit in the problem to solve. The use of CSP combined with ontological modelling is a powerful method for describing any pervasive computing scenario.

REFERENCES

Hoare, C.A.R. 1985. *Communicating Sequential Processes.* Prentice Hall International, London, UK.

Davies, N. et al, 2002, Beyond Prototypes: Challenges in Deploying Ubiquitous Systems. *IEEE Pervasive Computing*. Vol. 1, No 1, pp 26-35.

Masuoka, R. et al, 2003, Ontology-Enabled Pervasive Computing Applications. *IEEE Intelligent Systems.* Vol. 18, No. 5, pp 68-72.

Kolberg, M. et al, 2003. Compatibility Issues between Services Supporting Networked Appliances. *IEEE Communications Magazine*. Vol. 41, No 11, pp 136-147.

Prekop, P. et al, 2003. Activities, Context and Ubiquitous Computing. *Computer communications.* Vol 26, No. 11, pp 1168-1176.

Quantrill, M. et al, 2003. A Web Services Framework for Managing Complexity in a Pervasive Home Environment. *IASTED International Conference on Communications, Internet and Information Technology*, Scottsdale, AZ, USA, pp., ISBN.