Sensor Fusion-Based Middleware for Smart Homes

L. Coyle, S. Neely, G. Stevenson, M. Sullivan, S. Dobson, P. Nixon and G. Rey*

* Systems Research Group, School of Computer Science & Informatics, University College Dublin, Ireland
(Tel : 353-1-716-2485; Fax : 353-1- 269-7262; E-mail: lorcan.coyle@ucd.ie)

Abstract

Systems for home automation are a necessary component for intelligent smart homes applications. Existing systems suffer both from competing and often closed standards bases and from a message-based architecture that can complicate the development of flexible applications requiring information from disparate sources. We describe a knowledge-based pervasive computing middleware and show how it can be used to provide semantically rich unification over a range of home- and web-based automation systems.

1. Introduction

Whilst a number of automation systems for smart homes are available commercially, their use of competing (and often closed) standards and protocols can impede the creation of "smart homes" in which large numbers of heterogeneous devices communicate freely. Such systems also typically use simple broadcast or hub-and-spoke communications architectures which lead to excessive centralisation and can impede the free flow of information around a system, especially as new sensors are added to existing installations. Moreover, there is often a false dichotomy between sensor and other data such as web-based information sources and information stored digitally on PDAs or cellphones. These asymmetries are unhelpful for application developers.

A common approach to addressing such issues is through the use of middleware. Traditional solutions such as CORBA or JINI are generally regarded as being too heavyweight for smart environments, where devices' processing and communications capabilities are limited by cost and power considerations.

However, alternative forms of middleware remain an attractive solution to integrated home automation. In particular, we believe that it is vital for developers to be able to abstract away from the detailed topology, protocols, data formats and control of sensors, actuators and other information devices, and instead focus on the *fusion* of information from disparate sources. This provides great architectural flexibility and has the potential to improve responses to the noisy and uncertain information typically encountered in sensor-rich environments.

In this paper we introduce a sensor-fusion-based middleware for smart homes. The system – Construct – treats *all* devices as either sensors or actuators, allowing arbitrary home automation equipment to be controlled. All data are given a uniform representation, and their level of abstraction is raised through the use of knowledge-based data-fusion techniques. This aids developers in the process of building applications for the smart home, providing them with information that is semantically closer to their needs than the raw data provided by individual sensors. Furthermore Construct can collect data uniformly from other sources of interest, such as the web. This provides developers with a rich body of information with which to drive the automated home. The system is fully decentralised and decoupled from individual information sources, allowing flexible and robust use of an evolving device population.

To demonstrate the efficacy of Construct in the smart-home and assisted living domains, we describe a smart home-heating system that is under development. Construct collects the inputs from a diverse set of virtual and physical sensors and fuses them into a distributed data store using Semantic Web technology. Applications query this data directly rather than maintaining point-to-point connections with sensors. This decoupling of consumers of data from producers enables application developers to work with a single data format rather than requiring them to master a number of proprietary formats. This allows us to incorporate inferencing abilities into Construct that would not be possible in a system with fragmented data representation.

The rest of this paper is organised as follows: Section 2 discusses existing middleware solutions and home automation standards. Section 3 describes the Construct architecture and shows how it provides a semantic hub for home automation systems. Section 4 demonstrates

Construct's application to smart homes through a case study in controlling home environments using a variety of sensors based on radically different technologies. We show that the system provides developers with an integrated and homogeneous view of information that is not possible in more message-based systems. Section 5 concludes with some observations on middleware for smart-homes and some possible directions for future work.

2. Home automation

The complexity involved in constructing multi-vendor distributed systems can quickly become unmanageable without support from a tailored infrastructure. The standard approach is to build abstraction layers with common services made available to developers. Traditional middleware platforms provide these facilities; middleware in this context can be broadly defined as a layer between application and system software. Middleware systems support developers by automating much of the integration between various products and platforms whilst maintaining the integrity of the overall solution in terms of robustness and reliability.

Middleware for general distributed systems has evolved around a number of distinct categories. Object based systems such as Java EJB and CORBA provide a platform on which to build loosely-coupled object-based systems, complete with operations for registering objects, discovering new services, transaction handling, security and facilitating object message passing. Message-oriented middleware decouples client-server communications using the exchange of small messages. More recently, peer-to-peer (P2P) systems such as Pastry [12] and Chord [14] have become an area of significant research interest. By removing the need for infrastructural support, P2P systems can potentially support wireless (and other) *ad hoc* networks extremely well while distributing the load and costs of service provision over the node population – at a cost of more complex resource location, and no guarantees that particular services be, or remain, available. However, the removal of a central point of failure results in P2P systems having excellent fault tolerance properties that can be exploited.

The building and automation industries have a number of standard systems for use in the deployment of smart spaces. The LonWorks platform from Echelon Corporation is one such example. LonWorks was designed to address the issues of installation, performance, reliability and maintenance of control applications. It is built on a low-bandwidth communications protocol, LonTalk, facilitating the networking of devices over twisted-pair, power cables, fibre optics and radio. LonWorks has an affiliated IP tunneling standard (EIA-852) that can connect devices deployed on LonWorks-based networks to IP-aware applications and remote network management tools. This is an important step towards fully opening controls systems to Internet-based applications and services, but requires careful design, installation and management.

BACnet [4] is another example of a data communication protocol designed to support development of building automation and control networks. BACnet was developed in an attempt to create a standardised model for of representing devices and interactions between them with control applications. More recently the oBIX (Open Building Information Xchange) standard [8] has emerged: oBIX is an effort aiming to create standard XML and Web Services to facilitate the exchange of information between intelligent buildings and applications. A technical committee is working towards defining a standard web services protocol for exchange of information with the mechanical and electrical systems in commercial buildings.

Despite these efforts, there are an increasing number of sensor systems and modules becoming available that adhere to different (or indeed no) standards but which provide essential information for many applications. Some sensor vendors have focused on IP-enabled platforms using WiFi or Bluetooth for communications. Other systems provide proprietary, often research-based interfaces: Smart-ITs [2], Wavenis [1] and i-Bean [11] provide decentralised networks with simple data flows.

The technological landscape may be summarised as follows. Middleware systems assume that component nodes have significant processing and communications capabilities – assumptions that are typically not respected in pervasive computing systems, and perhaps especially in systems intended for use in existing buildings such as are typically encountered in home automation. Sensor systems often have only simple control and data interfaces, and do not provide an attractive programming platform for complex applications. It is hard to build decentralised applications and hard to handle the inevitable addition of (or failure of) devices in a way that does not require significant application customisation.

Moreover the need for self-management, self-description, self-configuration, self-optimisation and

other so-called "self-*" properties points towards the need for more autonomic and decentralised approaches to middleware targeting pervasive systems [9].

3. Construct

We have designed the Construct platform as a system for integrating noisy data sources in clean, dynamic, flexible and semantically well-founded manner. Construct provides applications with a uniform view of information regardless of how that information is derived, and supports extensive inferencing and sensor fusion within the platform to be shared between applications.

Construct has a fully decentralised architecture; devices within a smart space each run an instance. Each instance manages the local data provided by sensors (physical or logical) connected to that device – a "local star" topology in which dumb sensors are connected to more computationally capable hubs which then exchange information between themselves. Collectively, the devices maintain a global model of the data within a smart space. All data are modelled using the Resource Description Framework (RDF) [16], which provides a standardised way in which to model contextual information and properties. Construct stores and manipulates this data using the Jena Framework [10]. Data entering Construct are described using ontologies and Construct provides a query service that allows applications to query against the ontology rather than the data [5]. Data exchange between instances of Construct is performed using gossiping [15], which provides robustness and fine control over network utilisation.

Applications connect to a running instance of Construct, and use its services to obtain required information. Components with domain-specific knowledge may request and aggregate data from multiple sources in order to contribute new, or refined information. All data stored within the system are associated with metadata, which describe data lifespan and security restrictions.

Construct provides a core set of components (shown in Figure 1) that provide population management, data management, querying, and data propagation functionality [13]. Each deployment contains a Data Store Manager that is responsible for maintaining the local data model, and for removing old data as they expire. The Query Service allows applications to request the data they require to adapt their behaviour. The Gossiping layer select peers with which to share data, whilst components responsible for summarising information select the view of local data to be gossiped and integrated with remote deployments of Construct. Finally the Network layer abstracts the underlying communications medium.

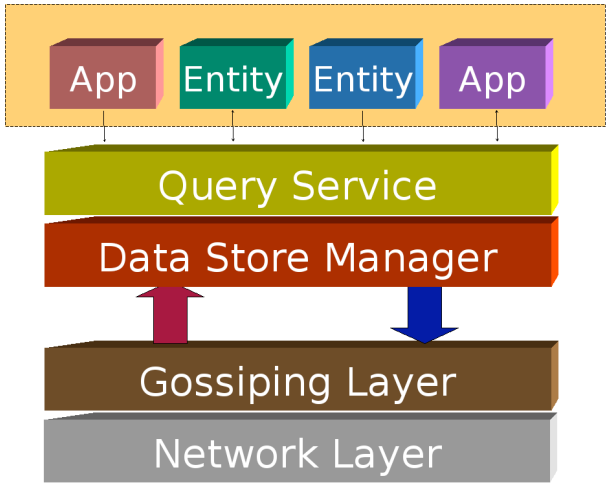

Figure 1: Construct architecture diagram

3.1. Mapping sensor data

Construct views *all* information sources as sensors that generate RDF triples. This uniformity means that to support a particular middleware standard or sensor protocol, a developer must define a mapping from that system's internal representation to RDF. (This has the incidental advantage of making the underlying system's information more easily accessible to open services and applications that can interpret RDF or XML information.)

The oBIX standard provides a way to model sensory devices in XML. We use the example of a simple thermostat from the oBIX specification document [8] to show how the XML information from oBIX is transformed *via* XSLT to RDF. This process is illustrated in Figure 2. The oBIX XML snippet models the current state of the thermostat; its temperature; the target temperature; and the actuator that controls the furnace. An XML transform is used to convert this XML to RDF. These RDF values are inserted into Construct's data store.

3.2. Virtual Sensors

Although home automation systems focus "inwards" on information sensed within the home, a properly pervasive approach to assisted living should also look "outwards" to information available on the web and through other digital channels. In Construct these information sources are simply additional sensors – with the advantage that the emerging consensus on XML and web services makes much of the information

available in formats that are close to (if not already) RDF.

We believe that this integration of traditional home automation with open access to web information is vitally important for smart-home applications, granting application developers access to a world of TV listings, weather reports, stock quotes, supermarket prices, etc. While these sources of data are somewhat useful for traditional smart home applications, they become far more valuable for applications tailored for inhabitants dealing with impairment. One may easily envisage scenarios in which an individual's medical or other requirements are downloaded securely from remote servers, and summaries of information derived from home-based sensors are uploaded to provide medical telemetry. Construct provides these capabilities uniformly.

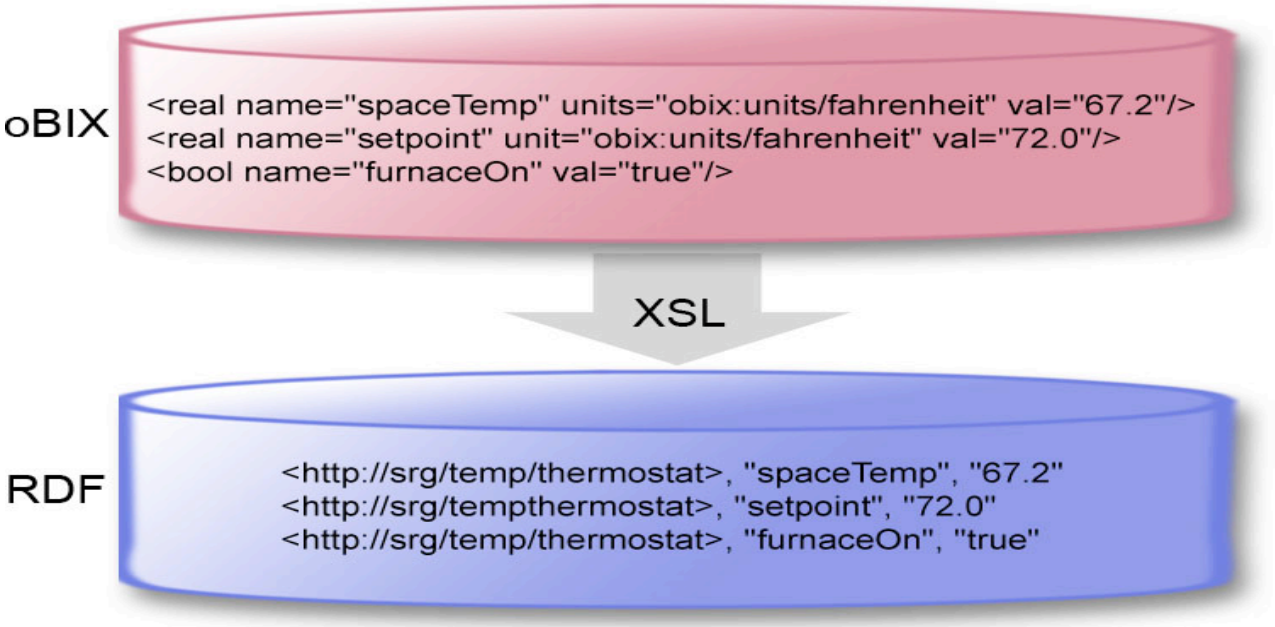

Figure 2: Diagrammatic representation of the transformation from oBIX to RDF

3.3. Programming

In addition to our work on web-sensors we are developing a number of virtual sensors. Rather than using physical or web data, virtual sensors operate on Construct's data store, using RDF querying languages (such as SPARQL or RDQL) to query for data of interest. When such data is available, it is processed by the sensors and the resultant values are added to the data store. One possible use for this technique is data-aggregation. For example, a virtual sensor can be used to compute an average temperature from the forecasts of several weather web sensors. Applications query data from Construct in the same way as virtual sensors.

We take the development of virtual sensors a step further by developing an inference framework that integrates with Construct's data store. Such a framework makes it possible for the system to discover regular patterns in behaviour and to use these to determine the context that underlies a sequence of actions. An illustrative example of this in the context of an assisted living smart home application would be if an alarm sensor were put in place to ensure that infirm users made safe night-time bathroom trips. If a person takes an inordinate amount of time to return from a bathroom trip it may be necessary to trigger an alarm.

4. Example Scenario

In this section we explore a simple smart homes scenario in order to demonstrate how a more semantics-driven approach to integration simplifies application development. Our aim is to show how the use of context fusion and symmetric information representation can provide a significantly enriched environment for pervasive applications over and above traditional sensor-driven systems.

4.1. The scenario

Consider a smart home that has an autonomous heating control system and a personalised alarm system. In winter, especially if the house is left unoccupied for any length of time, steps need to be taken to ensure that the

temperature does not fall below freezing (or else the water pipes may burst). Normally, this would be achieved using thermostat-controlled central heating. However, this home uses storage heating. Storage heaters use cheap electricity (usually at night-time) to store heat in ceramic bricks and release it during the day when electricity is expensive. It is more economical than a typical heating system if it stores enough heat each night to heat the house during the day.

The typical solution is for a user to manually turn the heater up in cold weather and down in warmer weather. Our proposal is to use web-published weather data to regulate the thermostat in the storage heater. We use a web-sensor to retrieve the weather forecasts and use these predictions to adjust the thermostat to reflect the expected temperature and store an appropriate amount of heat. This makes the regulation of the thermostat autonomous. If the weather forecast was incorrect, the thermostat will ensure that the temperature does not fall below freezing by turning the heaters on whenever necessary, which will of course necessitate the use of more expensive electricity.

This solution is enough to ensure that the house temperature does not drop too low. But what if the owner wishes to leave the house for a long period of time or invite a guest to stay in their absence? We have developed web-sensors that extract data from published calendar files. By incorporating information about the expected occupancy of the house and altering the thermostat temperature accordingly we improve the efficiency of the heating system.

4.2. The principles of a solution

Construct provides a solution to the problems of sensor heterogeneity, device volatility, data propagation, and application integration within smart-environments. Its use of the RDF standard for modelling context provides a common level of abstraction with which to represent information generated within a smart-space. This allows all data to be managed, manipulated, and disseminated independently of the technologies used to acquire it. In this way, Construct is able to deal with the heterogeneous sensors and devices that are typical in a smart-environment.

The arrangement of devices in a smart space may change frequently as devices turn on and off, users change location, and new data sources are integrated. By using established discovery protocols (such as Zeroconf [3]) to detect running instances of Construct, and a decentralised protocol for data propagation, Construct is robust to such changes. As the infrastructure is fully decentralised, the failure or removal of one deployment has minimal impact on the smart-space. This allows Construct to evolve in size gracefully, without the need for centralised administration. New devices need only detect a running instance of Construct to join the smart-space. In the smart home application described in the previous section, a new thermometer could be connected to the network, and Construct's discovery protocols would ensure that data from that thermometer was available to whatever applications are consuming temperature data.

5. Conclusions and ongoing work

Existing home automation systems exhibit a tight coupling between applications and the sensor networks that they utilise. As it is desirable to support both the integration of such systems and the sharing of sensed data across multiple applications, there is a clear need for middleware level support.

We introduce a sensor-fusion based middleware for smart-homes called Construct. Construct treats all components of a smart environment as sensors or actuators and maps data from these components into a unified format. This provides us with a common level of abstraction which simplifies the process of manipulating and querying for data.

Construct extends the traditional model of home automation systems, which focus on information sensed within the home to include information sensed from external digital media. As increasing amounts of information is published in digital formats and made available, additional value-added features can easily be integrated with assisted-living applications.

We describe an assisted-living scenario that demonstrates how Construct fuses sensed data and facilitates its consumption. Our next step is to continue building up a repository of physical and virtual sensors for use in the smart-home environment. We are complementing this work by the development of an inference framework that performs aggregation of data and deals with inconsistencies that may arise when the same type of data is produced from different sensors (such as when location sensors provide data that says a person is in two places at once) [6,7]. We are also developing a set of heuristics for evaluating the performance of Construct.

Construct is currently under development and will be released as open source software in the fourth quarter of 2006. Several sensors, demonstrator applications and ontologies describing context data will be part of this

release as well as a prototype of the smart home application described in this paper.

Acknowledgements

This work was partially supported by the grants "Secure and Predictable Pervasive Computing" from Science Foundation Ireland and "A Platform for User-Centred Evaluation of Context-Aware Adaptive Services" from Enterprise Ireland.

References

[1] Coronis systems, wavenis technology http://www.coronis-systems.com/descriptif.php?id=descr_tech
[2] The Smart-Its project. http://www.smart-its.org/.
[3] Zeroconf. http://www.zeroconf.org/.
[4] S. Bushby and H. Newman. The BACnet communication protocol for building automation systems. ASHRAE Journal, 33(4):14–21, April 1991.
[5] A. K. Clear, S. Knox, J. Ye, L. Coyle, S. Dobson, and P. Nixon. Integrating multiple contexts and ontologies in a pervasive computing framework. In Contexts and Ontologies: Theory, Practice and Applications, Riva Del Garda, Italy, August 2006.
[6] L. Coyle, S. Neely, P. Nixon, and A. Quigley. Sensor aggregation and integration in healthcare location based services. In 1st Workshop on Location Based Services for Health Care, November 2006. To Appear.
[7] S. Dobson, L. Coyle, and P. Nixon. Hybridising events and knowledge as a basis for building autonomic systems. In Journal of Trusted and Autonomic Computing, Sept. 2006. To appear.
[8] B. Frank. oBIX specification. Working Draft 0.8, December 2005.
[9] J. Kephart and D. Chess. The vision of autonomic computing. IEEE Computer, 36(1):41–52, January 2003.
[10] B. McBride. Jena: Implementing the RDF model and syntax specification. In Proceedings of the 2nd International Workshop on the Semantic Web, Hong Kong, May 2001.
[11] S. Rhee, D. Seetharam, S. Liu, N. Wang, and J. Xiao. i-beans: An ultra-low power wireless sensor network. In Interactive Poster in the Fifth International Conference on Ubiquitous Computing (UBICOMP), 2003.
[12] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. Lecture Notes in Computer Science, 2218:329–350, 2001.
[13] G. Stevenson, L. Coyle, S. Neely, S. Dobson, and P. Nixon. Construct — a decentralised context infrastructure for ubiquitous computing environments. In IT&T Annual Conference, Cork Institute of Technology, Ireland, 2005.
[14] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer- To-Peer Lookup Service for Internet Applications. In Proceedings of the 2001 ACM SIGCOMM Conference, pages 149–160, 2001.
[15] S. Voulgaris, M. Jelasity, and M. van Steen. A Robust and Scalable Peer-to-Peer Gossiping Protocol. In The Second International Workshop on Agents and Peer-to-Peer Computing (AP2PC), 2003.
[16] W3C. Resource Description Framework (RDF). http://www.w3.org/RDF/.