Combining System Introspection with User-Provided Description to Support Configuration and Understanding of Pervasive systems

Chris Greenhalgh[1], Kevin Glover[1], Jan Humble[2], Jamie Robinson[3], Steve Wilson[3], Jeremy Frey[3], Kevin Page[4], David De Roure[4]
[1]*School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK*
{cmg,ktg}@cs.nott.ac.uk
[2]*Progress Software, 200 Rustat House, Clifton Road CB1 7EG Cambridge, UK*
jan.c.humble@gmail.com
[3]*School of Chemistry, University of Southampton, Southampton SO17 1BJ, UK*
{j.m.robinson, sw1703, j.g.frey}@soton.ac.uk
[4]*School of ECS, University of Southampton, Southampton SO17 1BJ, UK*
{krp,dder}@ecs.soton.ac.uk

Abstract

Pervasive computing systems such as smart spaces typically combine multiple embedded and/or mobile sensing, computing and interaction devices. A variety of distributed computing approaches are used to integrate these devices to support coordinated applications. This paper describes how simple user descriptions of (primarily) physical aspects of such a system can be combined with information from system introspection to make the system and its log recordings more understandable to potential users, as well as supporting easier configuration and monitoring, and allowing the expression of certain kinds of system behaviour that are otherwise hard to achieve.

Keywords: Pervasive Computing, Semantic Web.

1. Introduction

The field of pervasive computing is founded on the proliferation of computing devices, including mobile and personal devices (mobile phones, PDAs, laptops, etc.), "traditional" desktop and server computers, digitally augmented objects (from key rings to cars) and diverse devices such as sensors and displays embedded within buildings and other settings. Mark Weiser [1] articulated one influential vision of how such devices might work together in a ubiquitous computing environment.

This paper describes how simple user descriptions of (primarily) physical aspects of such a system can be combined with information from system introspection to make the system and its data logs more understandable to potential users, as well as supporting easier configuration and monitoring. We achieve this using Semantic Web techniques, an established approach [2] but applied here in a new way.

Many computational and informational applications of computers are essentially independent of the specific hardware they employ or its situation. For example, information published through the World Wide Web or large numerical simulations on supercomputers can often be thought of just as 'bytes' or 'flops', essentially independent of their physical manifestation (give or take performance, latency, instruction set or access control). However many pervasive computing applications are fundamentally dependent on the particular hardware used and its situation. For example, a sensor network [3] consists of a coordinated array of sensor devices which are each intended to sense something in particular, such as the temperature or illumination, of the specific physical setting in which they are placed. To run the same sensing 'task' on devices somewhere else would be nonsensical.

However, when viewed from 'inside' a computational device, i.e. from within the digital or informational domain (such as software running on one of these sensors), the 'view' of the physical setting is profoundly limited. In this example, the 'temperature at the eastern shore' is likely to be a single fixed point number, being captured from an unremarkable I/O interface on one particular sensor node.

A variety of networking and distributed computing approaches are used to support pervasive computing systems and applications, and these can effectively overcome many of the challenges that arise in such systems. For example, the software running on the sensor network nodes may establish and maintain a spanning wireless network and coordinate data collection, querying and archiving across the whole sensor network, automatically reconfiguring the system if a sensor node fails [4].

However, we argue that there are many things that the supporting software will never be able to do without information from 'outside' the pervasive computing system, most commonly from the system's (human) designers or users. In this example, only the person wiring up the sensor node knows which A/D input is connected to the temperature sensor. Even if a system used 'smart' (self-describing) sensors it would still not know which of two temperature sensors was being used for which particular purpose. Only the person physically placing the sensor node knows exactly where it is. Even if the system included a location system such as GPS it would have limited precision, and would not capture potentially important location-related information such as whether it was in the lea of a nearby rock.

In section 2 we describe an instrumented chemistry lab, and the way that this is represented through the supporting software infrastructure. In section 3 we present the Physical Configuration Manager, the proof of concept application that we have developed using semantic web technologies to combine system introspection with user-provided description. Section 4 shows how this application – and the approach more generally – can support configuration, visualisation and specification of behaviours. In section 5 we consider outstanding issues and reflections. Finally section 6 concludes this paper.

2. Background

The particular and practical example that we consider in this paper is a chemistry lab in which significant elements of an experiment can be automated and left running unattended. However this means that the chemist may not be present in order to observe problems occurring during the experiment (such as a failure of the air conditioning). In other situations (e.g. anomalous results) it may be important to review the circumstances surrounding the preparation for and running of the experiment, with a view to ruling out possible errors such as contamination of samples.

In [5] we describe how we have previously used embedded sensors (temperature, humidity, light, PIR and door switches) and data feeds from other equipment and devices in one chemistry lab to support remote monitoring and historical review of lab conditions during experiments. We are continuing to use the same technologies within another nearby lab, which is the specific example used in this paper.

Consider a single sensor, for example one PIR sensor in a lab. The sensor's output is physically wired to one particular input on a data capture board, which in turn is connected to a particular computer. Software running on this computer, given the appropriate drivers, can interface to the data capture board and check the current state of each input, typically receiving a number which corresponds to the input voltage. One of the main concerns of this paper is: what does this number mean and how does a user know this?

The prototype presented in this paper is built on top of the EQUATOR Component Toolkit (ECT) [6], which is a middleware for prototyping (primarily) smart spaces and augmented artifacts. The primary building block in ECT is a software component, of which there are several kinds including hardware interface components, behaviour specification components (such as scripts) and user interface components (such as Processing applets). ECT provides a framework in which such components can be created, managed and inter-connected across a distributed set of computers. It also provides a set of user applications or tools for doing this.

In the case of the instrumented chemistry lab, data from the in situ sensors is initially exposed as messages distributed over a MQTT (Message Queue Telemetry Transport) protocol connection to an IBM Message Queue (MQ)-based middleware as described in [5], rather than via ECT. However a general purpose MQTT Bridge component in ECT allows this live data to be exposed within ECT in a similar way (in this case the software configuration requires network details of the MQ server to be used, and the message topics to subscribe to).

Whether connected directly or via some other middleware, the data from the PIR sensor (for example) can be viewed and used within the ECT-based system, e.g. to trigger other software components (such as graphical interfaces), and can be recorded using ECT's general logging facilities.

However looking exclusively at the views available from within the software – from within the digital domain – it is not at all clear what that value represents, i.e. that '1' actually means 'movement has been detected in a certain portion of chemistry lab 5017'. This issue may manifest as the system is deployed. It is even more likely to become problematic as time passes and other people get involved, both with the running system, and with data logged or recorded from the system: what does the data mean?

To expose the input from a PIR sensor in ECT a compatible hardware interface component must be created on the machine that is connected to the data capture board, and this component must be configured to communicate with the board (e.g. specifying a COM port or other hardware identifier). The view of the hardware interface component from within ECT will then reflect the current state of each analogue input, including the PIR sensor, as a set of component property values.

3. The Physical Configuration Manager

The Physical Configuration Manager (PCM) (see figure 1) is an alternative user interface for ECT which

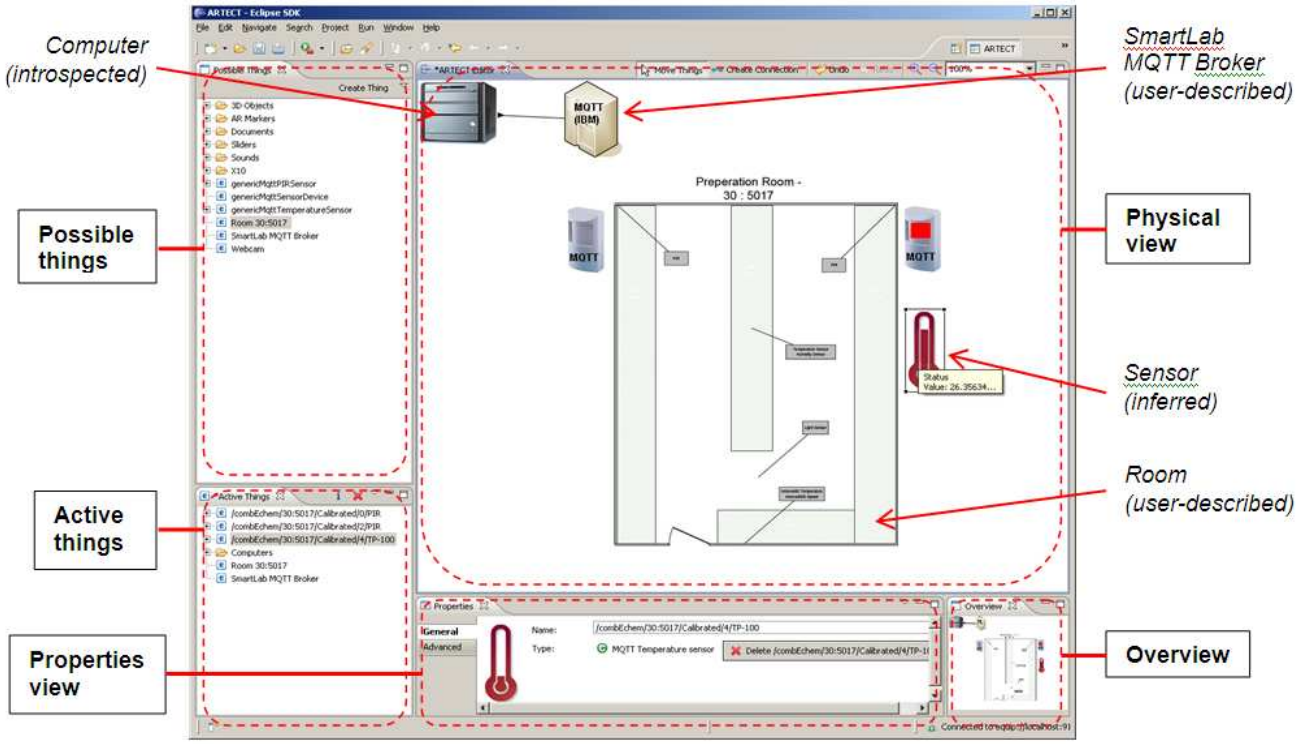

Figure 1. Screen image of the Physical Configuration Manager showing a view of a chemistry laboratory.

combines information derived from introspection of the running (software) system with user-provided descriptions of the hardware in use and the physical context. Work on the PCM was initially motivated by the conjecture that pervasive systems such as this might be easier to understand by having first-class representations of the physical and hardware elements of the system as well as the digital and software elements. Although this prototype is built specifically as a client for ECT, the same principles could be applied to other middleware for pervasive computing.

This section describes the architecture and general operation of the PCM; the following section gives examples of its use and capabilities. The PCM is implemented in Java using the Eclipse Rich Client Platform (RCP), i.e. it is a "heavyweight" desktop application. Its architecture is as shown in figure 2 and described below. It is implemented using the JENA open source RDF framework for Java.

The internal world model is defined using a relatively simple OWL ontology and maintained in a JENA RDF model (figure 2, centre). The graphical user interface consists of a number of Eclipse/RCP views which display different subsets and representations of this common model, e.g. a "physical" view (figure 1, top right), a software component view, a properties view (figure 1, bottom centre). Changes in the JENA model trigger events which cause the views to be updated (a typical model-view-controller approach).

The system connects to a running ECT system (or can start a new one), and continuously introspects the running system to find: all computers that are part of the system (such as that presented by the computer icon in figure 1), all software components that are currently running, all software components that can be created and all links currently established between software components. This introspection is a standard capability of ECT. This information is mapped to the ontology and published and maintained in the common model, which therefore is kept up to date with the state of the software system.

Through a simple drag and drop interface the user can identify physical 'things' that are currently to be considered as part of the system (figure 1, left, top & bottom, and the 'user-described' items in the main view). These can include: sensors such as the PIR sensor, devices such as the data capture board, locations such as the lab or other significant entities. Internally, these are cloned from the 'possible things' RDF model to the common model.

Linked to the common model is a forward-chaining rule engine (the standard RETE engine provided in JENA). As the model changes – due to user action or changes in the software system – these rules can fire to add or remove inferences to or from the current model.

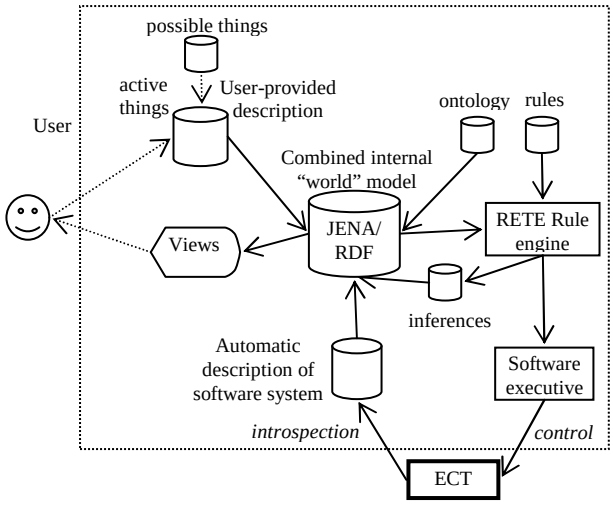

Figure 2. Internal architecture of the Physical Configuration Manager.

The rules include both standard entailments (more or less those of RDF Schema) and rules specific to the PCM. As statements are added to and removed from the common model the user interface views will update or animate accordingly. In addition, the presence of certain kinds of statements will cause the software executive (figure 2, bottom right) to make changes to the running ECT system and components, for example creating new software components, configuring them or connecting them together.

4. Examples of use

We will demonstrate how the Physical Configuration Manager (PCM) can provide help with: configuration and initial deployment of a system; visualising and understanding a system; and realizing higher-level behaviours. The configuration example shows how user-described things could cause software components to be created and configured. The visualisation example shows how the presence of software component(s) can imply the existence of physical things, and also be used to infer and show the state of physical things.

4.1. Configuration

In order to interface to the chemistry lab sensors from ECT the user must create the right software component to act as an MQTT bridge, and then configure it with the correct IP address, port number and topics of interest. Previously in ECT this would have been done using the graphical software component view. The PCM, through system introspection, has the same knowledge of ECT software components, and can render a similar software component view and editor.

However, the PCM also allows non-ECT elements of the system to be described and worked with. In this case, one of the authors has used the ontology to describe a "SmartLab MQTT Broker" physical thing. This RDF is initially loaded into the "possible things" model, and is visible in the "possible things" view (figure 1, top left). This description includes various information about this physical thing, in particular the kind of ECT software component that can interface to it, and the configuration information required in this particular case (IP address, port, topic). Now to connect to this particular broker the end-user: drags it from the "possible things" view to the main view (or to the "active things" view); and visually connects it to the ECT host computer in the main view (drawing a line between them, visible in the top-left corner of the main view in figure 1).

The standard PCM ontology and rule set support the idea that ECT software components can be 'proxies' for physical (external) things. Given the above description, the creation of the visual connection between the "SmartLab MQTT Broker" physical thing and the ECT host computer creates an ect:Association in the common model which causes rules to fire which: create a new MQTT Bridge software component (via the creation of a 'ect:createComponentRequest' statement, seen by the software executive), set the configServerUrl, configTopics and configured properties on the new component, and establish in the world model a 'proxy' relationship between the user-described "SmartLab MQTT Broker" physical thing and the corresponding MQTT Bridge software component.

4.2. Visualisation

To continue with this example, the MQTT Bridge software component connects to the external MQ broker which is distributing the sensor messages from the lab. As new messages are received it dynamically creates new MqttTopic software components within the ECT system to represent each message type (in this case this corresponds to each sensor in the lab). These are observed by the PCM as it monitors the ECT system, and are directly reflected in the software component view. Each MqttTopic (like the underlying message) has a 'topic' property (e.g. "/combeChem/30:5017/Calibrated/4/TP-1" – one of the temperature sensors in the lab), a 'value' property (e.g. "26.36") and a date/time when this value was measured.

The configuration example showed how user-described things could cause software components to be created and configured; in this case the presence of software component(s) implies the existence of physical things, in particular a temperature sensor. A custom rule causes a "genericMqttTemperatureSensor" from the possible things view to be created for each topic that matches the pattern ".*/TP-.*" (which is the common practice adopted in the lab to assigning topics to temperature sensor values). Note that this uses a custom

rule engine function “CreateResource”, which in the case copies the RDF subgraph describing an MQTT temperature sensor. The iconic representation of this inferred temperature sensor is visible towards the right of figure 1.

Another set of rules identifies relationships between described things and software components irrespective of whether user-described things cause the creation of the software components, or vice versa, or each is independently created. This correspondence is generally modelled by the software entity (component or property) having a “ect:hasProxy” relationship to the described (physical) thing. In this example the “SmartLab MQTT Broker” thing in the physical view is the proxy for the MQTT Bridge component, the temperature sensor is the proxy for the appropriate MqttTopic component, and in addition the gauge of the temperature sensor representation is the proxy for the MqttTopic’s ‘value’ property.

In this situation the rules infer that the property’s value (in this case, the temperature) should also be associated with the described thing (in this case, the temperature sensor’s gauge). This associated value can be used to animate the visual display in various ways, for example in figure 1 the temperature sensor icon’s central column moves up and down as the value changes (like a mercury thermometer) while the PIR sensor’s ‘activity’ indicator (a red square on the icon in figure 1) is visible only when the sensor is reporting activity. For positional sensors, the position of icons in the physical view can be modified as the sensed positions change.

Figure 1 also shows a plan of the lab space. This is purely a user-described thing – there is no explicit manifestation within ECT of that location. But within the context of the PCM physical view it provides a common reference frame for the placement of the physical devices. The various devices can also be dragged ‘into’ the lab location in that view and this locatedness is explicitly represented and exploitable within the common model.

Note that this visualisation or representation of the combined physical-software system can be useful in a number of different situations and contexts. During the initial deployment and configuration of the system it provides a common representation and at-a-glance indication of the system state. While the system is in ongoing use this view can provide a direct and hopefully intuitive view of the state of the system. It also encompasses a lot of the information that is needed when updating, maintaining or trouble-shooting the system (especially if the original hardware interfacing is also done with ECT and the PCM). Finally, it can be used to re-establish the context and meaning of historical data, such as that captured using ECT’s standard logging facilities (which can be used to record all of the activity with the ECT system and re-play or review it).

4.3. Behaviour

ECT (without the PCM) can be used to create a range of interactive system behaviours. For example, the various PIR sensor values from one room could be connected to a scripting component which combines their values to give a whole-room estimate of activity. The output from this could be connected to a relay output to switch a light or other indicator in another office to indicate (in)activity in the lab.

However, the PCM enhances this in two ways. First, given appropriate rules and descriptions the user can work in terms of the items in the physical view, rather than the generally intangible software components and properties of the ECT system itself. In another context we have been exploring the use of the PCM by museum curators to prototype interactive Augmented Reality installations, and in this application the main method of configuring the installation is by selectively linking the various described entities to imply specific relationships and interactions [7].

Second, some forms of behaviour are relatively easy to express in terms of RDF rules (in the PCM) but very hard to express through the data-flow transformations between component property values which are normally used in ECT. For example, a ECT Phidget RFID reader component publishes in ECT the IDs of the RFID tags that are currently in range of the reader. In the PCM the user-description can specify that a particular RFID tag is actually attached to a particular chemical sample. A straightforward rule can then infer that if (a) a certain RFID reader is reading a certain ID and (b) it is known (user-described) that that ID is associated with a certain sample and (c) it is known (user described) that the RFID reader is in a certain lab, then that sample must be in that lab at that time (and the visualisation will be updated accordingly). This is essentially the strategy used in many semantically oriented pervasive computing systems (e.g. [8]).

5. Discussion and future work

Having described the Physical Configuration Manager and given examples of its utility we now consider a number of other significant issues and aspects, including areas for further development.

5.1. Initial experiences

As reported in [7], initial experiences with museum curators and the Physical Configuration Manager have been generally very positive. They found the tool easy to use and easy to understand, and were able to quite rapidly create and evolve a range of interactive systems (in that case combining sensor and video inputs and audio and 3D graphical outputs). This lends support to the hypothesis that the “physical” perspective supported

by the PCM is accessible and comprehensible to users with no knowledge or particular understanding of the underlying software component infrastructure or concepts.

The main limitation that they encountered was in the range of behaviours and interactions that they could specify, which is essentially the range of behaviours specified in the rule set they were working with: without being reasonably expert in RDF and inference they were not in a position to "open the box" and extend the capabilities of the system themselves (this is also considered in section 5.3).

5.2. Performance and scale

Applying the PCM to the chemistry lab as presented here has been reasonably straightforward. However we have encountered some problems of scalability: when first connecting to the MQTT broker, if all of the available topics are received (several dozen) the PCM runs out of memory in the process of creating the various physical representations of the inferred sensors. More generally, the current version has a single physical view area (which can be zoomed in and out), and consequently has limited support for working with large systems or visualisations (e.g. no multiple pages, no nested pages). The ECT system itself, which underlies the PCM, is also limited in the scale at which it can sensibly be used, specifically across a handful of machines within a single organization (with a reliable network).

Part of the vision which lies behind the work on the instrumented chemistry lab in particular is the idea of publication at source [9], which implies that data from the lab – including relevant environmental data such as that considered here – should be reachable as part of the provenance trial of any academic publication which ultimately reports on the work. The work presented here contributes to this vision in that the PCM visualisation and the user-provided and inferred description which underlie it should make such data understandable to a reviewer or researcher wishing to explore a trail of data provenance. However neither ECT nor the PCM currently have any direct link to the kind of large-scale archival framework that this implies.

A light-weight starting point might simply be to deposit a saved (RDF-XML) copy of the common "world" model along with the corresponding ECT log in whatever archival framework is being used. In addition, where the system is actively evolving, the PCM metadata can also be woven into the time-based ECT log as it is generated. The Digital Replay System (DRS) includes basic support for replaying ECT logs, which can then be re-viewed using the PCM.

5.3. Extending and customising

The current version of the Physical Configuration Manager incorporates the ontology (OWL in RDF-XML format), possible thing descriptions (RDF-XML files) and both the standard and component and application-specific rules (in the textual format of the JENA rule engine).

Adding support for a new purely physical thing (e.g. a new room of interest) currently requires hand-authoring of appropriate RDF thing description(s). Support for another software component and/or hardware device also requires hand-authoring of appropriate RDF descriptions of the device, with annotations that describe its relationship to the corresponding software component(s) (e.g. configuration property settings). To create new descriptions automatically from running software components (as in 4.2) also requires the specification of the additional rules to do this. Specifying new rule-behaviours obviously requires the specification of such rules, but may also require supporting additions to the ontology (if the behaviour depends on concepts or properties that are not already modeled). Adding a new kind of animation of visualisation requires extensions to the Java implementation of the appropriate view elements within the Eclipse/RCP implementation of the PCM.

All of these tasks require skills and experience well beyond that required to simply use the Physical Configuration Manager (e.g. ability to write specific RDF-XML). To some extent this issue is ameliorated because much of this work need only be done once, e.g. when a new software component is written. This can then be included with the standard distribution (or distributed through ECT with the software components themselves, as is currently the case for the user documentation of the software components).

There are other examples, however, where an end-user might reasonably want to make additions to the system, for example specifying new specific instances of existing classes (another room, another MMQT broker, another RFID-tagged sample holder). This implies that the PCM user interface should also allow some limited and specific additions – mainly to the set of possible things – to be done simply and graphically by non-expert users.

5.4. Related work

Our work extends a body of work in Semantic Web and pervasive computing [1,8,10] by providing a case study of introspection, configuration and understanding. A clear parallel can be seen between this and the semantic annotation of Web Services. However the kinds of configuration and behaviour choices being made in a system such as the one described are not

necessarily a good fit for a semantic service model. For example, the challenge is not generally to find a "semantically compatible" service (e.g. [11, 12]), but to work in a coordinated way with the particular physical and digital entities that are at hand. Also, the kinds of visualisations and interactions that are seen in the PCM (e.g. plans and maps) fit well with pervasive computing systems (especially smart spaces and location-based systems) but do not apply as naturally or universally to the more abstract data and process flows found in general Web Services.

In terms of systems intended to support pervasive computing applications the Physical Configuration Manager provides a concrete answer to the often overlooked question of where semantically rich descriptions actually come from in the first place. It also demonstrates one way in which semantics can be brought into play in such a system which is more general in scope than an "add-on" for service discovery, but which does not necessarily require universal support for semantics. While there are many similarities of implementation technology with [8], the emphasis there is on inferring and abstracting context information from "lower level" information in order to create applications based on the derived higher-level context. The PCM shows how the semantic level can usefully link directly back to the "lower level" devices and components, and also how the developer and user can (and probably should) be involved and supported in this process.

6. Conclusions

In this paper we have demonstrated how simple user descriptions of (primarily) physical aspects of a pervasive computing system (in particular an instrumented chemistry lab) can be combined with information from software system introspection to make the system more understandable to potential users, as well as supporting easier configuration and monitoring, and allowing the expression of certain kinds of system behaviour that are otherwise hard to achieve.

The Physical Configuration Manager is a usable proof of concept realization of this proposition, which employs semantic web technologies to combine and reason across user-provided descriptions and run-time system information. The same philosophy and approach could be applied to other systems and middleware.

ECT and the Physical Configuration Manager are available under the BSD open source licence from SourceForge (CVS branch 'semanticmedia').

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EP/C010078/1) and the European Union's 6th Framework Programme (IST-2004-004150).

References

[1] Weiser, M.: The Computer for the Twenty-First Century. Scientific American 94-104, 1991.

[2] Lassila, O., and M. Adler: Semantic Gadgets: Ubiquitous Computing Meets the Semantic Web. In D. Fensel et al.: Spinning the Semantic Web: 363-376, 2003.

[3] Szewczyk, R., Osterweil, E., Polastrem, J., Hamilton, M., Mainwaring, A.M., and Estrin, D.: Habitat monitoring with sensor networks. Commun. ACM 47(6) 34-40, 2004.

[4] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K.: System architecture directions for networked sensors. In Proceedings of the ninth international conference on Architectural support for programming languages and operating systems, pp. 93-104. ACM Press, 2000.

[5] Robinson, J. M., Frey, J. G., Stanford-Clark, A. J., Reynolds, A. D., and Bedi, B. V.: Sensor Networks and Grid Middleware for Laboratory Monitoring. In Proceedings of the First International Conference on E-Science and Grid Computing, pp. 562-569. IEEE Computer Society, 2005.

[6] Greenhalgh, C.: A Toolkit to Support Rapid Construction of Ubicomp Environments. In: UbiSys 2004: System Support for Ubiquitous Computing Workshop at the Sixth Annual Conference on Ubiquitous Computing , 2004.

[7] Koleva, B., Egglestone, S.R., Glover, K., Hampshire, A., Greenhalgh, C., Dade-Robertson, M.: Creating hybrid artefacts using an abstracted user-oriented representation. Submitted to Interacting with Computers, special issue on Physicality and Interaction.

[8] Wang, X., Dong, J.S., Chin, C.Y., Hettiarachchi, S.R., Zhang, D.: Semantic Space: An Infrastructure for Smart Spaces. IEEE Pervasive Computing 3(3) 32-39, 2004.

[9] Frey, J.G., De Roure, D., Carr, L.: Publication At Source: Scientific Communication from a Publication Web to a Data Grid. In: EuroWeb, Oxford. 2002.

[10] Ryusuke Masuoka, Yannis Labrou, Bijan Parsia, Evren Sirin, Ontology-Enabled Pervasive Computing Applications, IEEE Intelligent Systems, vol. 18, no. 5, pp. 68-72, Sept/Oct, 2003.

[11] Jaeger, M., Rojec-Goldmann, G., Liebetruth, C., Mühl, G., Geihs, K.: Ranked Matching for Service Descriptions Using OWL-S. in: Kommunikation in Verteilten Systemen (KiVS), pp. 91-102, 2005.

[12] Bandara, A., Payne, T., De Roure, D., Lewis, T.: A Semantic Framework for Priority-Based Service Matching in Pervasive Environments. In: On the Move to Meaningful Internet Systems: OTM 2007 Workshops, pp. 783-793, 2007.